Cilia Segmentation in Medical Videos with Fourier Convolutional Neural Network

by

Weiwen Xu

(Under the Direction of Shannon Quinn)

Abstract

The study of cilia motion analysis is crucial because cilia are found on almost all vertebrate cells. Abnormal function of cilia can manifest as a variety of symptoms. Traditional study of cilia beat pattern and beat frequency for cilia motion analysis rely on manual labeling for locating cilia. However, manual segmentation of cilia is time consuming and difficult even for a professional. The time and effort needed for constructing cilia segmentation makes the study of cilia motion analysis extremely hard. Therefore, an automatic model for producing cilia segmentation is highly needed. In this work, Fourier methods are introduced to traditional convolution neural network for cilia segmentation utilizing the distinct structure of Cilia. Although the result is not as satisfactory as hoped, we manage to establish the theory support for our model, provide thorough analysis and hypotheses for the potential problem based on our results. The possibility of utilizing frequency spectrum in the application of image segmentation in deep learning model is well explored in this work and thorough discussion and hypotheses for future work is discussed as well.

Index words: Image Segmentation, Fourier Transform, Signal Proccessing, Deep Learning, Convoluational Neural Network

Cilia Segmentation in Medical Videos with Fourier Convolutional Neural Network

by

Weiwen Xu

B.S.(Hons), University of Liverpool, 2017

B.S., Xi'an Jiaotong Liverpool University, 2017

A Thesis Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

2019

Cilia Segmentation in Medical Videos with Fourier Convolutional Neural Network

by

Weiwen Xu

Major Professor: Shannon Quinn

Committee: Yi Hong
Frederick Maier

Electronic Version Approved:

Ron Walcott
Interim Dean of the Graduate School
The University of Georgia
December 2019

Dedication

Dear grandma and grandpa.

Acknowledgments

I would like to thank my major advisor Dr. Shannon Quinn, for the encouragement, patience, and advice he has provided. I would like to thank Dr. Yi Hong, for the help for providing advice and being on my committee and reviewing this thesis. I would like to thank Dr. Frederick Maier for help when needed as well as being on my committee and reviewing this thesis.

I would like to thank many of my dear lab mates including Bahaa who kindly went through my theory with me, Sonia who is always kind and supportive, and my dear friends including Isela who is always there to discuss with me when I needed help, and many others.

I would like to thank my family for all of their support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis proposes a new deep learning method that combines Fourier transform traditionally used in signal processing and deep learning model for modeling the cilia representation in frequency domain in order to help with the segmentation of cilia area in medical videos. Chapter two gives some previous work related to image segmentation and spectral analysis for dynamic texture. Chapter three introduces the data process, the design of basic Fourier blocks and two Fourier models using those Fourier blocks that solves the segmentation problem from different perspectives. Chapter four gives introduction of the dataset we use, demonstrates the results we have for both of the model, and raises hypotheses for the potential problems. The remainder of this chapter gives brief introduction to cilia biology, cilia motion analysis and provides the motivation of this work.

1.1 Introduction to Cilia

The cilium is a organelle with a fibrillar substructure [1, 2] that protrudes from surface of lots of cells [3]. An example of cilia is in Figure 1.1. There are two types of cilia: motile cilia and primary or non-motile cilia [2]. Although imaged to be 'static', primary cilia are also dynamic and it helps to sense extracellular signals from the environment [4]. However, we are not working with primary cilia in this thesis. What we are dealing with in this paper is motile cilia which normally project on surface of a cell in large numbers and beat together in coordinated waves [2]. Motile cilia functions in cell motility or movement of extracellular fluids [3]. Cilia mentioned below in the rest of the thesis all refer to motile cilia.

Cilium beat at a certain tempo known as the cilia beat frequency (CBF). Cilia beating

pattern is asymmetric and contains two distinctive strokes: effective stroke and recovery strokes. Effective strokes are faster and stronger than recovery stroke as it produces a strong force to propel mucus forward while recovery stroke is for cilia to return to the original position in the underlying periciliary fluid against the flow [2]. Such beat pattern produces net fluid flow in the direction of effective stroke. There are around 200 to 300 cilia on a ciliated epithelial cell and those cilia must beat in same direction and in coordinated waves to ensure proper fluid propulsion [2].

1.2 Cilia Motion Analysis

Ciliopathies include a group of disorders associated with genetic mutations resulting in either abnormal formation or dysfunction of cilia. Cilia are found on almost all vertebrate cells. Abnormal function of cilia can manifest as a variety of features that include characteristically retinal degeneration, renal disease, cerebral anomalies as well as congenital fibrocystic diseases of the liver, diabetes, obesity and skeletal dysplasias [5].

Primary ciliary dyskinesia (PCD) is a rare genetic disorder of dysfunctional respiratory cilia. High-speed video-microscopy analysis (HVMA) of cilia beat pattern (CBP) and cilia beat frequency has been recommended as a first-line diagnostic test along with transmission electron microscopy (TEM) in the study and diagnose of primary ciliary dyskinesia. Further methods for primary ciliary dyskinesia diagnosis include immunofluorescence (IF) microscopy, genetics and measurement of nasal nitric oxide (nNO) production [6].

Among videomicroscopy analysis methods, a lot of different models have been applied to cilia motion analysis. S. Quinn et. al. uses differential image velocity invariants to classify cilia motion [7]. In [8], it proposed a end-to-end pipeline with convolutional LSTM model that provides automated analysis for cilia motion. Other attempts include analysis of cilia movement on kymograph extracted from a sequence of digital images [2]. However, all these models suffer from the difficulty of localizing cilia areas. It is mostly done by manual labeling based on one frame of each video producing one static mask. Convolutional neural network

for cilia segmentation in [8] uses static mask image as ground truth for each video during training process.

However, this is problematic. Some of the videos are low quality with the problem of camera drifting and being out-of-focus. Meanwhile, some cells move along with cilia. These facts might result in incorrect ground truth masks which are problematic to the training process. In addition, the natural characteristic of cilia being small relative to the size of frames makes it hard to detect for non-experts let alone the problem of time-consuming even for professionals.

1.3 Motivation

Manual labeling for cilia is, first of all, very difficult. The nature of cilia being small in size makes it hard to be detected even by a professional. The poor video quality problems like camera drifting, being out-of-focus increases the difficulty. Secondly, manual labeling is extremely time consuming. It has been the most time consuming part for the cilia motion analysis study. Since locating cilia is the first step towards the study of cilia motion analysis, developing an automatic model for producing the cilia segmentation is highly needed. Last but not least, using static masks for training the model for cilia segmentation in videos is problematic. The cilia beat motion indicates that the area of cilia changes for each frame. Therefore, the ground truth segmentation mask ideally should be slightly different for each frame. However, most of all the models use only on static mask for each video. Additionally, some part of the cell beat together with cilia. Using one static mask as ground truth for the entire video can accidentally include part of the cell as being cilia resulting in incorrect ground truth.

The fact that cilia has periodic beat pattern indicates pixels inside cilia area has distinct 1D signal along the time axis meaning they have rich temporal information. Such distinct signal should result in high peeks in frequency spectrum obtained by fast Fourier transform. Similarly, the distinct hair-like structure indicates distinct 2D signal of cilia, meaning cilia

has rich spatial information. This should also result in high peeks in 2D frequency spectrum from fast Fourier transform. Fourier transform transforming signals from spectral domain to frequency domain is useful when the interested object has some distinction in the frequency spectrum. Since both the temporal and spatial of cilia have distinct frequency in frequency spectrum. Working in frequency domain with Fourier transform seems to be promising. J. Zhang et. al. embedded Fourier analysis into recurrent neural network creating Fourier recurrent unit [9]. It also proves that Fourier basis is more powerful than polynomials for approximating functions. However, J. Zhang et. al. only explores temporal information for one dimensional signals [9] while for the segmentation of cilia area in videos should consider both spatial and temporal information because cilia motion could be subtle for cilia with defect. Therefore, a model that simultaneously learns spatial and temporal feature is needed in the application of cilia segmentation. Luckily, with 3D Fourier transform, the model can simultaneously learn both spatial and temporal information easily.

1.4 Contributions

Our contributions in this thesis can be summarized as follows:

1. We introduce a new convolutional and pooling operation that allows convolutional neural network to learn features entirely in frequency domain

2. We transform the octave convolution block to help with the memory issue in the proposed Fourier convolution and pooling operation.

3. We propose a revised W-Net with the application of proposed convolutional and pooling operation, and further reduce the number of parameters with octave convolution to reduce redundancy in such neural network.

4. We propose a revised U-Net for partial reconstruction of the original video with cilia area only with the application of Fourier convolution, pooling and octave operation.

5. We give hypothesis for the potential problems shown in the results and analyze the design of the methods in different perspectives.

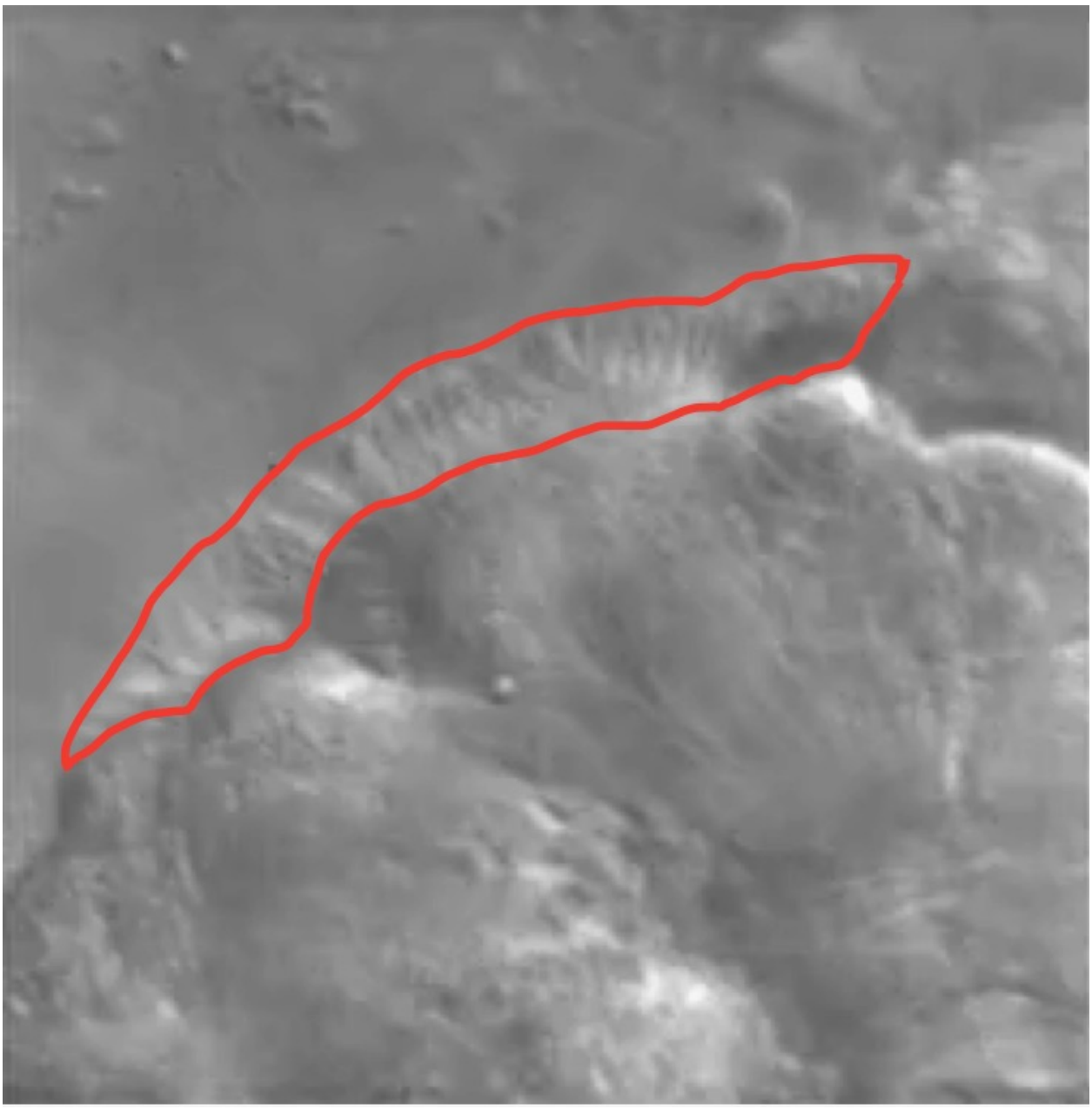

Figure 1.1: Example of cilia. The area highlighted by red lines is cilia.

Chapter 2

Related Work

In this chapter, a range of related work has been conducted for the problem of image segmentation and spectral analysis will be introduced. This will include the related deep learning approach for image segmentation. The introduction of spectral analysis for dynamic texture will focus on the application of Fourier transform for such topic as well as its limitations.

2.1 Image Segmentation

Image segmentation has been one of the most important topics in the image processing field of study. Image processing aims at cluster pixels into image regions. Early stage studies include thresholding based or edge based segmentation like watershed algorithm. Other multivariable clustering methods like K-means and expectation maximization can also be used as unsupervised image segmentation algorithm. With the development of deep neural network, performance of neural network models on image segmentation is largely improved from the traditional methods. Convolutional neural network is often used in such topics and there are couple of classic convolutional neural networks in the application of image segmentation like U-Net that works on supervised image segmentation and has achieved significant improvement in IoU ("intersection over union") result from its prior sliding window convolutional neural network on the ISBI cell tracking challenge 2015 [10]. Other convolutional neural network like W-Net aims at providing unsupervised learning for image segmentation purpose with the improved n-cut loss that optimizes cut for different groups using graph theory.

U-Net introduced in [10] contains two main paths. The first path is a contracting part

which follows the typical architecture of a convolutional network. It consists of four stages until a bottleneck is reached. In each stage, two 3×3 convolutions are applied first with each followed by a ReLu unit and a 2×2 max pooling operation with stride 2 for downsampling. The contracting path is therefore also called downsampling path. Each stage in downsampling path compresses the feature map to half while doubling the number of channels. When a bottleneck is reached two 3×3 convolutions are applied while keeping number of channels the same. The second path is an expansive path also known as upsampling path. Similar to contracting path, it also has four stages. Each stage consists of a upsamping of the feature map from previous stage followed by a 2×2 convolutions that halves the number of channels. The resulting feature map is concatenated with the correspondingly cropped feature map from the downsampling path. Following the concatenation, two 3×3 convolutions are applied with each followed by a ReLu activation function. In order to produce the segmentation map, a 1×1 convolution is used to map the resulting 64-component feature vector from the upsampling path to a desired number of classes. The loss function is to compute the cross entropy between the final output feature map and the corresponding ground truth mask.

X. Xia and B. Kulis proposes an Autoencoder styled network called W-Net for unsupervised image segmentation including an encoder followed by a decoder and the segmentation result is in the middle [11]. Both encoder and decoder have similar structure as U-Net [10] described above. The last convolutional layer of the encoder is a 1×1 convolution which maps the feature vector to a desired number of classes denoted as K, followed by a softmax layers which re-scales them to range $(0, 1)$. The decoder takes the output of encoder as its input. The final convolution layer of the decoder is a 1×1 convolution to map feature vectos back to its original input. The loss function of W-Net, apart from the normal reconstruction loss that is common to Autoencoder, consists of two parts. The reconstruction loss is computed by mean square error of output feature map and input image. The other part it soft N-cut

loss. The soft N-cut loss is a soft version of N-cut loss introduced by [12] since the original N-cut loss is non-differentiable.

2.2 Spectral Analysis for Dynamic Texture

Static texture in images is described as repeated pattern that exhibits some extend of variability in their appearance [13]. While dynamic texture, also known as temporal texture, is to extend the static texture to the spatio-temporal domain. Temporal variations like motion and deformation are introduced in dynamic texture. Dynamic textures are defined as sequences of images of moving scenes that show certain stationarity properties across time [14].

The feature extraction in the study of texture analysis is essential. A large variety of methods have been developed and can be applied to extract meaningful information from the raw pixel values of the images. Signal processing approaches like filter banks, wavelets and Fourier transform fall in the category called spectral analysis. These methods analyse frequency or spatial-frequency content of textures in spatial domain like steerable filters, in frequency domain like Fourier transform or in both of the two domains like Gabor filters and wavelet transforms.

2.2.1 Fourier Transforms and Convolution Theorem

In mathematics, a basis is a finite or infinite set $B = \{\vec{b_i}\}_{i \in I}$ of base vectors $\vec{b_i}$ that spans the whole space and is linearly independent. Therefore the basis must be an orthogonal set. And in vector space theory, any vector $\vec{v}$ can be expressed as a linear summation of a finite basis elements.

In Fourier theory, it makes use of such theory and creates a Fourier basis that contains sinusoids of any frequency. Therefore, in Fourier theory, any periodic functions (with period T) can be expressed by sum of those Fourier basis and this is called Fourier series or Fourier expansion.

$$
\begin{aligned}
f(t) &= a_0 + \sum_{n=1}^{\infty} a_n \cos\left(\frac{2\pi nt}{T}\right) + \sum_{m=1}^{\infty} b_n \sin\left(\frac{2\pi mt}{T}\right) \\
&= \sum_{n=0}^{\infty} a_n \cos\left(\frac{2\pi nt}{T}\right) + \sum_{m=1}^{\infty} b_n \sin\left(\frac{2\pi mt}{T}\right) \\
a_0 &= \frac{1}{T}\int_0^T f(t)dt \\
a_n &= \frac{2}{T}\int_0^T f(t)\cos\left(\frac{2\pi nt}{T}\right)dt,\ n \geq 1 \\
b_m &= \frac{2}{T}\int_0^T f(t)\cos\left(\frac{2\pi mt}{T}\right)dt,\ m \geq 1
\end{aligned}
$$

Utilizing Euler's equation $e^{it} = \cos t + i \sin t$, we can know from the above Fourier series equation that a_n are actually the real coefficients and b_m are the imaginary coefficients. With the help of the same equation, we can express sinusoids as $\cos t = \frac{e^{it}+e^{-it}}{2}$ and $\sin t = \frac{e^{it}-e^{-it}}{2i}$. Therefore, we can rewrite the previous Fourier series with complex numbers.

$$
\begin{aligned}
f(t) &= \sum_{n=-\infty}^{\infty} U_n e^{i\frac{2\pi nt}{T}} \\
U_n &= \frac{1}{T}\int_0^T f(t)e^{-i\frac{2\pi nt}{T}}dt
\end{aligned}
$$

However, Fourier series alone does not extend Fourier theory beyond periodic functions. Fourier transform is the extension of the idea of Fourier series to non-periodic functions. Letting the period expand from T to infinite expands the definition from periodic functions to aperiodic functions and also fundamentally changes the nature of transformation from time domain to frequency domain. This is called Fourier transform. The notation for Fourier transform of function $f(t)$ is $\mathscr{F}\{f(t)\}$. The result of Fourier transform of function $f(t)$ is a function of frequency f therefore we can also represent it as $F(f)$. $F(f)$ is often called the spectrum of $f(t)$. There is also inverse Fourier transform which transforms result of Fourier

transform $F(f)$ back to its original function $f(t)$, denoted as $\mathscr{F}^{-1}\{F(t)\}$. $f(t)$ and $F(f)$ are also called a Fourier pair because they are distinct representation of the same underlying identity.

$$\mathscr{F}\{f(t)\} = F(f) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi ft}dt$$
$$\mathscr{F}^{-1}\{F(f)\} = f(t) = \int_{-\infty}^{\infty} F(f)e^{i2\pi ft}df$$

Fourier transform is defined on continuous signal or continuous function. In the application of image processing, discrete Fourier transform (DFT) which deals with a finite discrete-time signal is often used. Discrete Fourier transform is the sampled Fourier transform. Instead of containing all frequencies, it contains only a set of frequencies. The number of frequencies is the same of the size of image. Since gray-scale images are 2-D functions $f(x, y)$ where x and y are coordinates of pixel location and each $f(x, y)$ is its corresponding pixel intensity value, 2D discrete Fourier transform is what has been used in the application of image processing. For a square image of size $N \times N$, the 2D discrete Fourier transform and 2D inverse discrete Fourier transform are defined as:

$$F(m, n) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y)e^{-i2\pi(\frac{xm+yn}{N})}$$
$$f(x, y) = \frac{1}{N^2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} F(m, n)e^{i2\pi(\frac{mx+ny}{N})}$$

The complexity of computing the DFT is $O(N^2)$ as each pixel point is calculated twice. Fast Fourier transform is an algorithm that is introduced for computing DFT with complexity of $O(N \log N)$. There are many variants of fast Fourier transform algorithm. The most classic one from [15] states that adding certain data sequence values after they have

been multiplied by the same factors of fixed complex constants during the evaluation of different DFT transform coefficients causes redundancies. And the efficiency of traditional DFT is improved by re-ordering the data sequence and/or transform sequence to eliminate such redundancies. We can also extend DFT to 3 dimensions by doing the same procedure to the three axes.

Convolution theorem states that the Fourier transform of a convolution of two signals is the point-wise product of their Fourier transforms. It is equivalent to say that convolution in time domain is point-wise multiplication in the frequency domain. Therefore, in image processing, if we define image as a 2-dimensional function f and define filter as g, we have

$$\mathscr{F}\{f * g\} = \mathscr{F}\{f\}\mathscr{F}\{g\}$$

To note that in convolution theorem, there is a constrain on the two signals that both of them should be in the same length. Therefore, compared to convolution applied in normal convolutional neural network, the definition of filter signal g is to pad the kernel used to convolve the feature map with with value 0 to the same size as the feature map as shown in Figure 2.1.

2.2.2 Drawbacks of Fourier Transform

The main drawback of Fourier transform in the application of image analysis is that Fourier transform does not describe local features. A narrow frequency band-pass represents a large spatial region. A small change in the frequency domain affects large area in the spatial domain. However, for image segmentation and analysis of cilia texture, describing local variations of the cilia texture is crucial. Ideally, texture operator of both spatial and frequency domain should be localised. However, the localisation of a texture operator in both spatial and frequency domain is limited due to Heisenberg's uncertainty. The Heisenberg's uncertainty states that the localisation in both frequency and spatial domain is limited by a lower bound on their product:

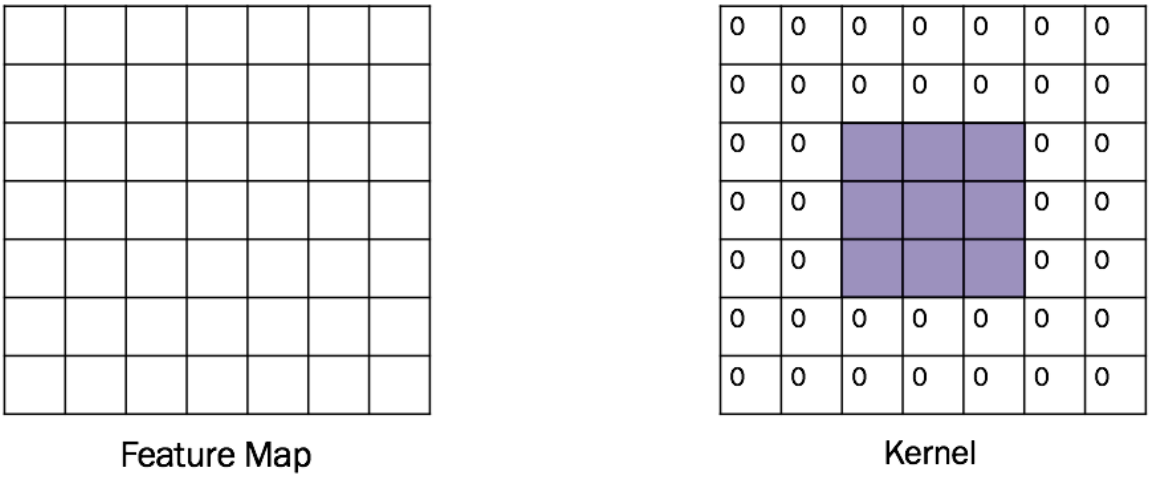

Figure 2.1: Left: an example of feature map in traditional convolutional neural network. Right: middle purple part (3 * 3) is an example of 3×3 convolution kernel. In order to satisfy the convolution theorem, it should be padded to the same size as the feature map in the left with value 0.

$$\Delta x \Delta y \Delta f_x \Delta f_y \geq (\frac{h}{4\pi})^2$$

where Δ refers to the uncertainty in the variable and h is Planck's constant.

Chapter 3

Network Architecture and Pipeline

In this chapter, detailed network architectures and overall data processing pipeline are included. General data preparation is discussed first. Small redesigned reusable component blocks are proposed, followed by discussion of two models for the segmentation of cilia. Starting with pre-processing process, the proposed Fourier convolution and pooling method is introduced afterwards. Thirdly, octave convolution block is discussed. Following the octave convolution block, the first model for unsupervised pixel-wise classification is introduced along with the W-Net architecture that this first model is based on. Lastly, the second model for supervised partial video reconstruction is presented. Introduction of both of the model also includes their respective additional touch to the general data preparation method, post-processing method.

3.1 Data Pre-processing

Because our videos are all in difference sizes. These videos are all resized to same size for each frame and with same length, meaning all the videos are resized and truncated in number of frames to (250, 250, 250). It makes sure the standard size for future process.

To prepare the data for the Fourier convolution neural network, Fourier transform should be applied to map original videos to the frequency domain to to make the data mathematically suitable for the network to learn features in frequency domain. However, mapping

original video data to frequency domain with Fourier transform alone is not enough. Additionally, certain processing should also be applied to address the problem of localisation of Fourier transform.

3.1.1 Video Normalizing and Patching

The first step in pre-proceeing should be normalizing the data. In order to make sure such process will not affect the data in frequency domain, a simple method is used. Each video is subtracted by its mean and divided by its deviation.

In the previous chapter, we have discussed the drawback of Fourier transform, that it is not able to describe localized features in spatial domain since a narrow frequency band-pass represents a large spatial domain. Yet, in dynamic texture analysis and image segmentation, we need operator to catch localised features to distinguish different features between interested object/texture and background area. Ideally, such operator should be localized both in frequency and spatial domain but the localisation of operator is limited by a lower bound on their product by Heisenberg's uncertainty.

Patches are made from each video to extract both spatial and frequency information by computing Fourier transform on those local neighbourhoods. Size of patches on each frame should neither be too big nor too small. A big patch cannot catch the localised information and a small patch can lose important frequency information. Therefore, the choice of patch size should be decided with causality. For each frame of the video, the spatial information requires small sized patches so that local spatial information could be caught which is helpful to the model to distinguish interested object area or not. Because of the natural repeated beating pattern of cilia, frequency in time domain of each pixel within cilia area shows strong characteristic compared to non-cilia area and this characteristic is consistent across time domain. Such strong characteristic is also helpful for classification of each pixel. Therefore, unlike for spatial information where feature is not consistent because of different objects presented in the image, we can have much longer patch size to catch rich temporal

information. Therefore, the patch size is 25×25 on each frame and 100 in time. We used `tf.contrib.signal.frame(plane, length, step)` to make the patches where plane is the data to be patched, length is the size of the output patch, step is the length to move the window for each patching. This function can only frame along one axis at a time so for our 3D data, we need to apply it 3 times. The patches in spatial is non-overlapping but it is overlapped by 50 steps along time axis. Therefore, to conclude, for one video, we can produce 100 volume patches with size $(100, 25, 25)$ spatially and each with 4 patches in time. We can see the 4 volumes in time as 4 channels. Therefore, we can get volume data for each video in $100 \times 100 \times 25 \times 25 \times 4$ dimension where $(100, 25, 25)$ is the size of feature map (also meaning the size of the volume data), 100 as the "batch" number and 4 is the number of channels. This 100 "batches" will be the input of our network and we will only input this 100 patches for one video for each batch. The meaning of batch here is a bit different from other the ones we use for training other networks. This will be explain in later section where we introduce the detailed structure of our model.

One of our goal is to learn spatial and temporal information. This can be realized by trying to understand the distinct frequency representation of cilia area in 3 dimension. This is why we made the patches of videos into three dimensional volumes.

In the actual implementation, FFT is computed with `tf.spectral.fft3d`. However, zero frequency component, which is also called DC component in signal processing, is at top left corner. In order to prepare for easier process for later stages, the result is shifted using `np.fft.fftshift` by half of the size in both the directions so that we have a symmetric frequency matrix. To conclude, after shifting, the zero frequency is in the center; frequencies close to center is called low frequencies and frequencies that is around the edges are called high frequencies.

3.1.2 Fourier Transform and Complex Numbers

After normalizing and preparing patches to extract local information for each video, the video patches should then be mapped onto frequency domain to allow future computation in the Fourier convolutional neural network. Fast Fourier transform (FFT) method is used to create the frequency representation of the video patches.

Resulting matrices of Fast Fourier transform all contain complex numbers. According to convolution theorem, both video data and kernel should be transformed with FFT and then be multiplied together. Ideally, we should have the transformed video data and directly initialize kernels with complex number which represent the FFT result of the "kernels" of the normal convolutional neural network. However, Tensorflow does not support kernel initialization with complex numbers and backwards signals during back propagation cannot be computed for complex numbers. Therefore, we need to seek for other methods.

Complex numbers are in the form of $a + bi$ where $i = \sqrt{-1}$. Both a and b are real numbers. a is called the real part and b is called the imaginary part. Complex numbers can be thought of as vectors in the complex plane with basis vectors $(1, 0)$ and $(0, i)$. The length of a complex number is called magnitude where $|a + bi| = \sqrt{a^2 + b^2}$ and the angle from the real-number axis is called phase where $\phi(a + bi) = \tan^{-1}(\frac{b}{a})$. The property of complex includes the fact that when you multiple two complex numbers, their magnitudes multiply as well: $|z_1 z_2| = |z_1||z_2|$.

Utilizing such property, we can tackle the problem of complex number in Tensorflow. The magnitudes of FFT results of the video patches are calculated for each video. Meanwhile, kernels could be initialized with real numbers as usual which represent the magnitude of FFT result of the "kernels" of the normal convolutional neural network. The multiplication required by convolution theorem can be applied by multiplication of the two magnitude matrices. The result means the magnitude of FFT result of resulting convoluted matrix. Writing this to notations for clarification:

$$
\begin{aligned}
\text{Normal convolution: } Fm(t) &= Fm(t-1) * Kernel(t) \\
\text{Fourier convolution: } |\mathscr{F}\{Fm(t)\}| &= |\mathscr{F}\{Fm(t-1) * Kernel(t)\}| \\
&= |\mathscr{F}\{Fm(t-1)\}\mathscr{F}\{Kernel(t)\}| \\
&= |\mathscr{F}\{Fm(t-1)\}||\mathscr{F}\{Kernel(t)\}| \\
\text{(Equivalent to) } Fm_{Fourier}(t) &= Fm_{Fourier}(t-1)Kernel_{Fourier}(t)
\end{aligned}
$$

Such process has another meaning at the same time. Auto-correlation is the correction of a function with itself: $f(t) * f(-t)$. Power spectrum of a signal is the Fouier transform of its auto-correlation functions.

$$
\begin{aligned}
P(s) &= \mathscr{F}\{f(t) * f(-t)\} \\
&= F(s)F^*(s) \\
&= |F(s)|^2
\end{aligned}
$$

The power spectrum is useful for detecting periodic patterns/texture in the image. In our case, because of the hair-like appearance of cilia in the image, patches of cilia area have strong periodic pattern. Similarly, the repeated beating pattern of cilia makes the pixel values along time axis for each pixel location show strong periodic pattern as well. As the power spectrum is the squared magnitude of the Fourier transform of the function. The aforementioned process can also be seen as calculating square root of power spectrum of the resulting convoluted function.

3.2 Fourier convolution Block and Octave Block

This section will discuss about the proposed 3D Fourier convolution and transpose 3D Fourier convolution, as well as the Fourier octave convolution that aims at reducing redundancy for the Fourier convolution operator. The octave block used is introduced in [16] and we are utilizing it to work with our 3D Fourier convolution. As mentioned in previous section, this is designed to learn three dimensional frequency which means that we are trying to do equivalent operation in frequency domain as spatial 3D convolution operation. We denote feature map as F with size (M, N, D) and denote kernel as K with the same size (M, N, D).

3.2.1 3D Fourier convolution and pooling operator

With the convolution theorem, Fourier convolution operator could simply be multiplying feature map with kernel of the same size as feature map. The result of multiplication has the same size as feature map and kernel which is (M, N, D).

However, since we are dealing with image data, we have an additional axis which is the channel size. Therefore, the true size of feature map would be $(M, N, D, inChan)$ where $inChan$ stands channel size. However, this point-wise multiplication operation alone does not support the change in the number of channel, which is crucial to any convolutional neural network. In the traditional convolution operation in 2D CNN, the change of channel size can be explained by an example. For example, if we are talking about using a 3×3 Conv kernel on a feature map of size $(M, N, inChan)$, the size of kernel is actually $3 \times 3 \times inChan$ where the 3×3 convolution operation results of $inChan$ numbers of channels are summed up to squeeze the channel size of resulting feature map to one. Then, by using $outChan$ number of such kernels in the same way, we can increase or decrease the channel size of the feature map. In our fourier convolution, we can use the same strategy by using kernel of same size to do the point-wise multiplication, summing all $inChan$ channels to squeeze them down to one,

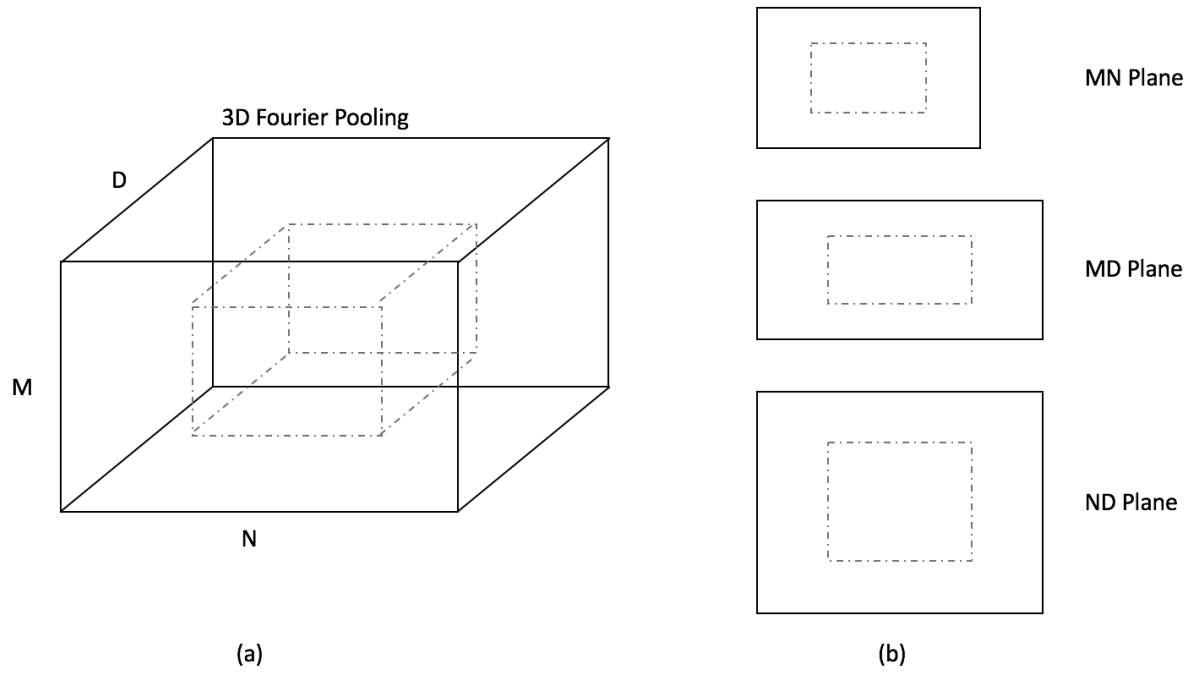

Figure 3.1: 3D Fourier Pooling. (a) Overview of 3D Fourier Pooling; (b) View from three orthogonal planes. Equal portion of both sides in each axis are truncated. This results in a smaller volume inside the original feature map containing all the low frequencies. The size of result feature map from pooling can be controlled by the pooling ratio.

and manipulate the channel size by applying the same operation $outChan$ times. Whether or not the summing operation affects the feature map the same way as it does for normal convolution is still debatable and will be discussed in later sections. We can also change to use a simple max pooling or average pooling along the channel axis to replace the addition.

The transpose 3D Fourier convolution can be realized by padding around the edge of feature map with value 0 to the desired size first and then applying 3D Fourier convolution to the padded feature map.

Pooling operation normally is used to reduce the size of feature map to reduce memory used for computation and maintain the important and valuable information in the feature map at the same time. In frequency domain, the image data is distributed differently com-

pared to spatial domain. We can still reduce the data size but retain more information. The low frequency data closer to center is the valuable information that we want to retain. Therefore, we just need to truncate the outer layer of the matrices. Figure 3.1 illustrates the 3D Fourier pooling operation. Therefore, given feature map of $M \times N \times D$ dimension, the Fourier pooling operation that reduces the size of feature map to an arbitrary ratio `p` is computed as follows:

$$m_{min} = (0.5 - \frac{p}{2}) \times M,\ m_{max} = (0.5 + \frac{p}{2}) \times M$$
$$n_{min} = (0.5 - \frac{p}{2}) \times N,\ n_{max} = (0.5 + \frac{p}{2}) \times N$$
$$d_{min} = (0.5 - \frac{p}{2}) \times D,\ d_{max} = (0.5 + \frac{p}{2}) \times D$$

We denote the Fourier pooling operation as `fourierPool(X,p)` where p is the variable for ratio of output size to the input size and X is the input tensor.

3.2.2 Fourier Octave Block

The size of feature map can be reduced by Fourier pooling operator. However, since Fourier convolution requires the kernel to have the same size as feature map, there is still redundancy in kernel compared to spatial convolution operator. Here is the comparison of number of parameters of one U-Net path for traditional convolution method and Fourier convolution method in Table 3.1. In [16], it introduces a method called octave convolution that reduces such redundancy. As an application of octave convolution in our proposed model, the octave convolution discussed here could be slightly different from the original one.

The Fourier octave convolution aims at effectively processing the low and high frequency in their corresponding tensor but also allow efficient communication between the two frequency components. Suppose we have a feature map F with size (M, N, D, C) where M, N, D are height, width and depth of the volume (time axis in our case) and C is the number of

U-Net Layers	Normal Convolution	Vanilla Fourier Convolution
Down sampling 1-1	3*3*1*64	100*25*25*4*64
Down sampling 1-2	3*3*64*64	100*25*25*64*64
Down sampling 2-1	3*3*64*128	50*12*12*64*128
Down sampling 2-2	3*3*128*128	50*12*12*128*128
Down sampling 3-1	3*3*128*256	25*6*6*128*256
Down sampling 3-2	3*3*256*256	25*6*6*256*256
Down sampling 4-1	3*3*256*512	12*3*3*256*512
Down sampling 4-2	3*3*512*512	12*3*3*512*512
Bottleneck 1	3*3*512*1024	6*1*1*512*1024
Bottleneck 2	3*3*1024*1024	6*1*1*1024*1024
Total	~18.84m	~589.33m

Table 3.1: Comparison of number of parameters of one U-Net downsampling path for traditional convolution method and Fourier convolution method.

feature channels. The first step is to factorize the feature map into high and low frequency component. We denote $F = \{F^H, F^L\}$ as the factorized frequency feature tensors. Such factorization is only used in the first layer where low frequency group takes αC number of channels and a Fourier convolution is applied followed by activation function and 3D Fourier Pooling to get the most inner 0.5 low frequency information. Meanwhile, high frequency is the remaining $(1 - \alpha)C$ channels and a 3D Fourier convolution is applied followed by activation function ReLU.

The octave operation is given by $F^H = F^{H\to H} + F^{L\to H}$ and $F^L = F^{L\to L} + F^{H\to L}$ respectively, where $F^{A\to B}$ is denoted as update from group A to group B. In [16], it describes $F^{H\to H}$ and $F^{L\to L}$ as intra-frequency information update and $F^{H\to L}$ and $F^{L\to H}$ as inter-frequency information update.

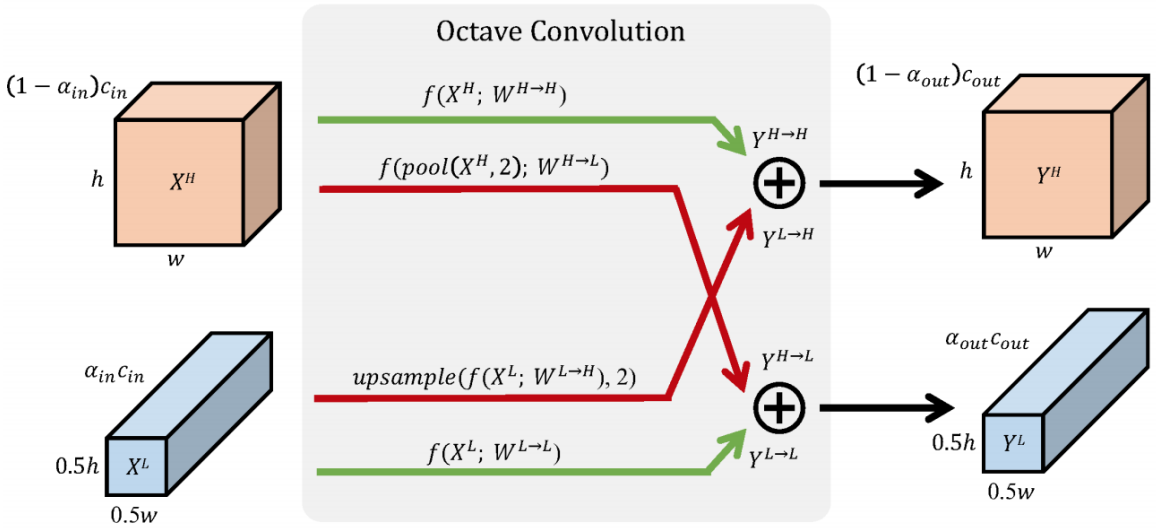

Figure 3.2: Octave Convolution in [16]. Our octave convolution is different in the convolution function $f(X;W)$. In the original octave convolution, it refers to the normal spatial convolution operation. Here we refer to 3D Fourier convolution.

Kernel should therefore be factorized in the same way. We denote factorized kernel $K = \{K^H, K^L\}$. K^H is responsible for operation with F^H and K^L is responsible for operation with F^L. It can be further divided into intra- and inter-frequency part for each component where $K^H = \{K^{H\to H}, K^{L\to H}\}$ and $K^L = \{K^{L\to L}, K^{H\to L}\}$.

Figure 3.2 show the whole octave convolution update process. We used the proposed 3D Fourier convolution instead spatial convolution for $F^{A\to B}$. Applying Fourier convolution operation within Octave convolution making it Fourier octave convolution, the computation is done as follows:

$$
\begin{aligned}
F^H &= F^{H\to H} + F^{L\to H} \\
&= K^{H\to H}F^H + upsample(F^L K^{L\to H}, 2) \\
F^L &= F^{L\to L} + F^{H\to L} \\
&= K^{L\to L}F^L + fourierPool(F^L, 0.5)K^{H\to L}
\end{aligned}
$$

`upsample(X,2)` is transpose 3D Fourier convolution of the input tensor X that expands the size of tensor to twice as its original size.

Table 3.2 shows the significant contribution octave block has to reducing the number of parameters.

U-Net Layers	Vanilla Fourier Convolution	Octave Fourier Convolution ($\alpha_{in} = 0.8$, $\alpha_{out} = 0.8$)
Down sampling 1-1	100*25*25*4*64	100*25*25*1*13 + 50*12*12*3*51
Down sampling 1-2	100*25*25*64*64	100*25*25*13*13 + 50*12*12*51*51
Down sampling 2-1	50*12*12*64*128	50*12*12*13*26 + 25*6*6*51*102
Down sampling 2-2	50*12*12*128*128	50*12*12*26*26 + 25*6*6*102*102
Down sampling 3-1	25*6*6*128*256	25*6*6*26*52 + 12*3*3*102*204
Down sampling 3-2	25*6*6*256*256	25*6*6*52*52 + 12*3*3*204*204
Down sampling 4-1	12*3*3*256*512	12*3*3*52*103 + 6*1*1*204*409 (bottleneck 1)
Down sampling 4-2	12*3*3*512*512	12*3*3*103*103 + 6*1*1*409*409 (bottleneck 2)
Bottleneck 1	6*1*1*512*1024	
Bottleneck 2	6*1*1*1024*1024	
Total	~589.33m	~66.17m

Table 3.2: Comparison of number of parameters of one U-Net downsampling path for vanilla Fourier convolution method and octave Fourier convolution method. Because of the design of octave operation, the octave Fourier convolution reaches bottleneck one stage earlier than the vanilla Fourier convolution.

3.3 W-Net and Revised Fourier W-Net for Segmentation - First Model

3.3.1 W-Net

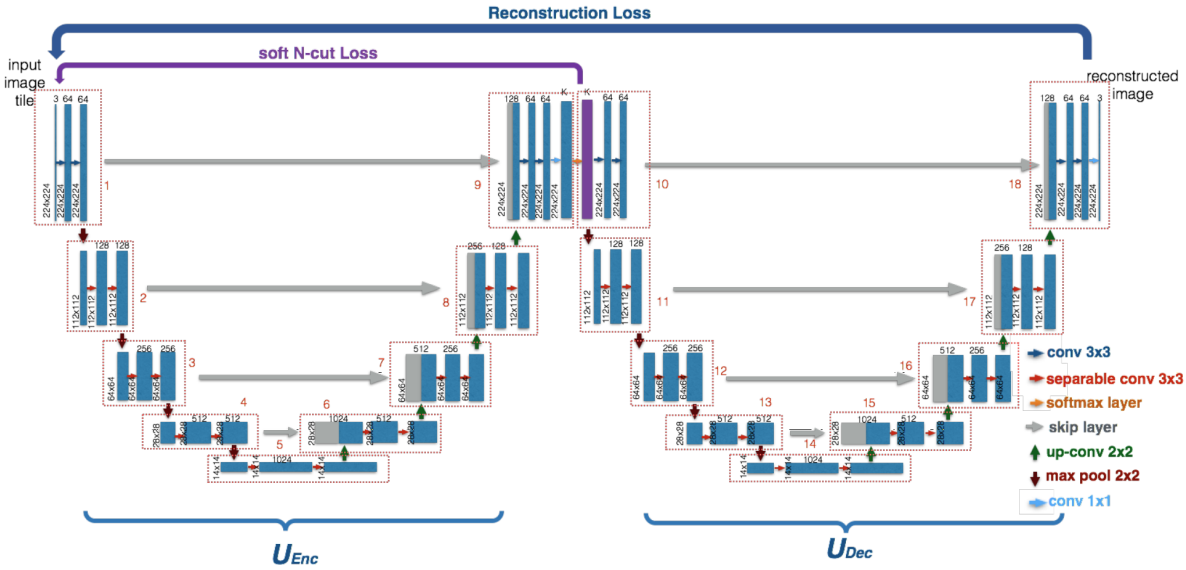

Figure 3.3: W-Net architecture [11]. It is constructed by two U-Net forming an autoencoder model. It is used for unsupervised image segmentation problems.

W-Net introduced in [11] has an overall autoencoder with two fully convolutional network architectures tied together. Both of the fully convolutional network have similar structure as U-Net introduced in [10]. The encoder network encodes the input image into a k-way soft segmentation where k is the pre-defined number of classes. The decoder network takes in the segmentation layer and reverse the process back to a reconstructed image. The reconstruction error between reconstructed image and input image is minimized. Also, a soft version of normalized cut loss function on the encoder network is also jointly minimized with the reconstruction error. Also, some post processing methods are used on the output segmentation result. This will be discussed in next section. In this section, we only focus on the detailed network structure and its revised version with aforementioned proposed methods that allows the network explore features entirely in frequency domain.

The overall network structure can be seen in Figure 3.3 from where we can clearly see an

encoder-decoder structure. Both encoder and decoder network contain two parts which are called down-sampling and up-sampling. There are four stages in down-sampling path until a bottleneck is reached. In each down-sampling stage, two 3×3 convolutions each followed by a ReLU activation function and batch normalization are applied with padding to ensure the size of feature map remain unchanged. The first convolution doubles the channel size while the second convolution does not change the channel size. A max-pooling layer that reduces the feature map to half of its size is then applied and it also links one down-sampling stage with the next one. Each of the feature map produced after completion of a down-sampling stage before pooling layer is saved for later use. When the network reaches the bottleneck after 4 down-sampling stages, two convolutions same as those in the down-sampling stage are applied. The resulting feature map is then up-sampled by a transposed 2D convolution layer to twice of its size while halving number of feature channels. The up-sampling process connects the up-sampling stages. The up-sampled feature map is then concatenated with the feature map of the equivalent stage in down-sampling saved previously. This is also called the skip layer. Same process as in down-sampling stages are then applied.

The final layer of encoder network is a 1×1 convolution follower by a softmax layer. The convolution layer maps each of the 64-dimensional feature vector to a specific number of k classes. Output of softmax layer is the raw segmentation result.

Decoder network is very similar to encoder network. It reads the output segmentation map from the encoder which has size $244 \times 244 \times K$. However, the last layer of decoder network is a 1×1 convolution to map the feature vectors back to a reconstruction of its original input without an activation function.

Loss function of the W-Net consists of two loss. One is the reconstruction loss coming from the autoencoder network which is computed as:

$$
\begin{aligned}
MSE(\boldsymbol{X}, \hat{\boldsymbol{X}}) &= ||\boldsymbol{X} - \hat{\boldsymbol{X}}||_2^2 \\
&= ||\boldsymbol{X} - U_{Dec}(U_{Enc}(\boldsymbol{X}; \boldsymbol{W_{Enc}}); \boldsymbol{W_{Dec}})||_2^2
\end{aligned}
$$

The other one is called soft N-cut loss which is a differentiable version of the normalised cut (N-cut) loss introduced in [17]. Graph theory is applied in N-cut. The normalised cut loss method tries to avoid bias of producing a lot of small partitions by calculating the cut cost as a fraction of the total edge connections to all the nodes in the graph. This can be computed as:

$$
\begin{aligned}
Ncut_K(V) &= \sum_{k=1}^{K} \frac{cut(A_k, V - A_k)}{assoc(A_k, V)} \\
&= \sum_{k=1}^{K} \frac{assoc(A_k, V) - assoc(A_k, A_k)}{assoc(A_k, V)} \\
&= \sum_{k=1}^{K} (1 - \frac{assoc(A_k, A_k)}{assoc(A_k, V)}) \\
&= K - \sum_{k=1}^{K} \frac{assoc(A_k, A_k)}{assoc(A_k, V)} \\
&= K - \sum_{k=1}^{K} \frac{\sum_{u \in A_k, v \in A_k} w(u, v)}{\sum_{u \in A_k, t \in V} w(u, t)}
\end{aligned}
$$

However, the original normalized cut loss is not differentiable and cannot be used to calculate the corresponding gradient during backprogation in the network. Therefore [11] introduces a differentiable version of this N-cut:

$$J_{soft-Ncut}(V,K) = K - \sum_{k=1}^{K} \frac{assoc(A_k, A_k)}{assoc(A_k, V)}$$
$$= K - \sum_{k=1}^{K} \frac{\sum_{u\in V, v\in V} w(u,v)p(u = A_k)p(v = A_k)}{\sum_{u\in A_k, t\in V} w(u,t)p(u = A_k)}$$

where $p(u = A_k)$ means the probability of u belonging to class A_k which is computed by the encoder. Therefore, we can minimize the total cut between groups while maximizing the total cut within groups at the same time by training the encoder to minimize the soft N-cut loss.

3.3.2 Revised Fourier W-Net for Segmentation

The revised Fourier W-Net has similar structure as the original W-Net with the infusion of Fourier theory and parts designed for segmentation. First of all, the overall structure of the revised Fourier W-Net can be broken down to two main components. They are the 3D Fourier W-Net and 2D segmentation W-Net. To continue with, both of the two main components can be further divided into two paths. Each of the two components has a down-sampling path and a up-sampling path. The two components are assembled as shown in Figure 3.4 where the 2D segmentation W-Net is embedded between the encoder and decoder of the 3D Fourier W-Net with a data transform part as linkage.

Next sections will explain the detailed structure of 3D Fourier encoder and the transformation part that links the 3D Fourier encoder and 2D segmentation encoder. The reason why we need these components will be given and the corresponding loss function will be introduced. Since the 3D Fourier decoder is simply the inverse of its corresponding encoder, structure of the decoder will not be discussed in detail. Same thing for the inverse data transform. We use the original W-Net structure for the inner 2D segmentation network and since we have introduced the original W-Net in detail in previous sections, we will not discuss it either.

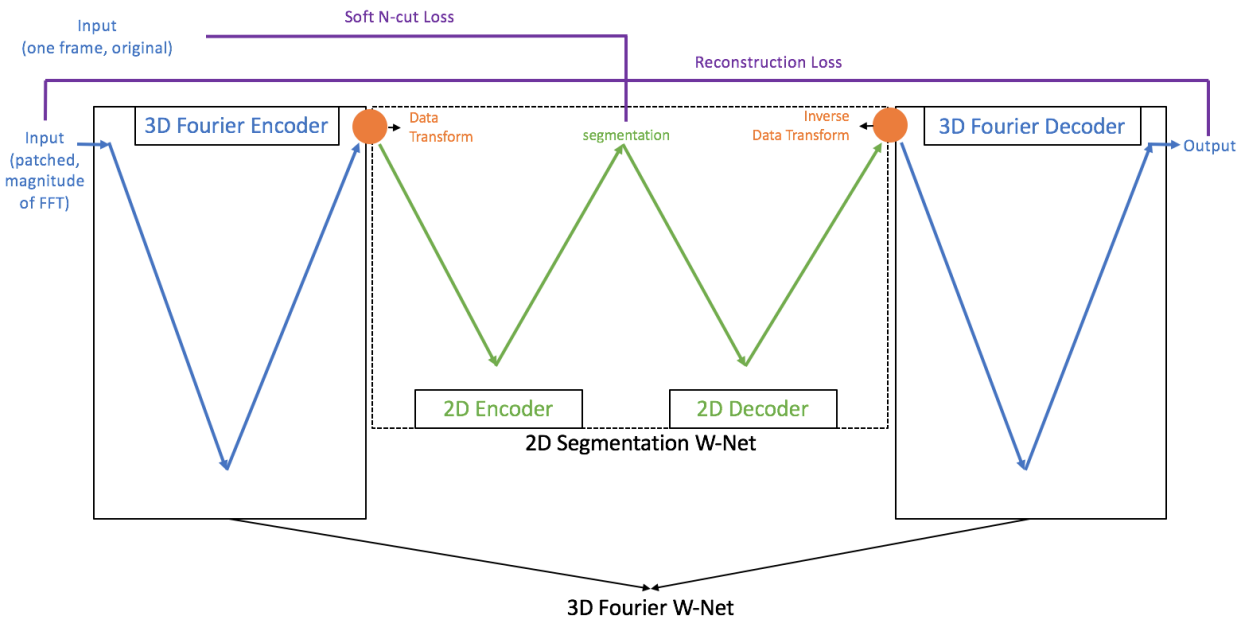

Figure 3.4: The simple illustration of overall structure for the entire revised Fourier W-Net for segmentation with one smaller W-Net embedded inside of another larger W-Net. The blue parts on two sides are 3D Fourier W-Net which works entirely in frequency domain with Fourier operations proposed. 3D Fourier Encoder is used for embedding in 3D frequency domain for cilia. The green part in the middle is a smaller W-Net with traditional convolution operation. It is used to compress embedding information along time axis to produce 2D segmentation.

3D Fourier Encoder

Figure 3.5 shows the detailed structure of the 3D Fourier encoder. The 3D Fourier encoder takes in 4D data with one axis for number of channels. In Figure 3.5, it shows an example of a 3D volume in size (25,25,100) with 4 channels. All the 2D convolutional operations used in original W-Net are replaced by the 3D octave Fourier convolution and the original 2D pooling layers are replaced by the 3D Fourier pooling layers as well.

Divided by the bottleneck block, the left side is the down-sampling sub-component. Similar to the original encoder in W-Net, each stage in down-sampling path consists of one 3D octave Fourier convolution and one 3D Fourier pooling operation besides the first stage. The

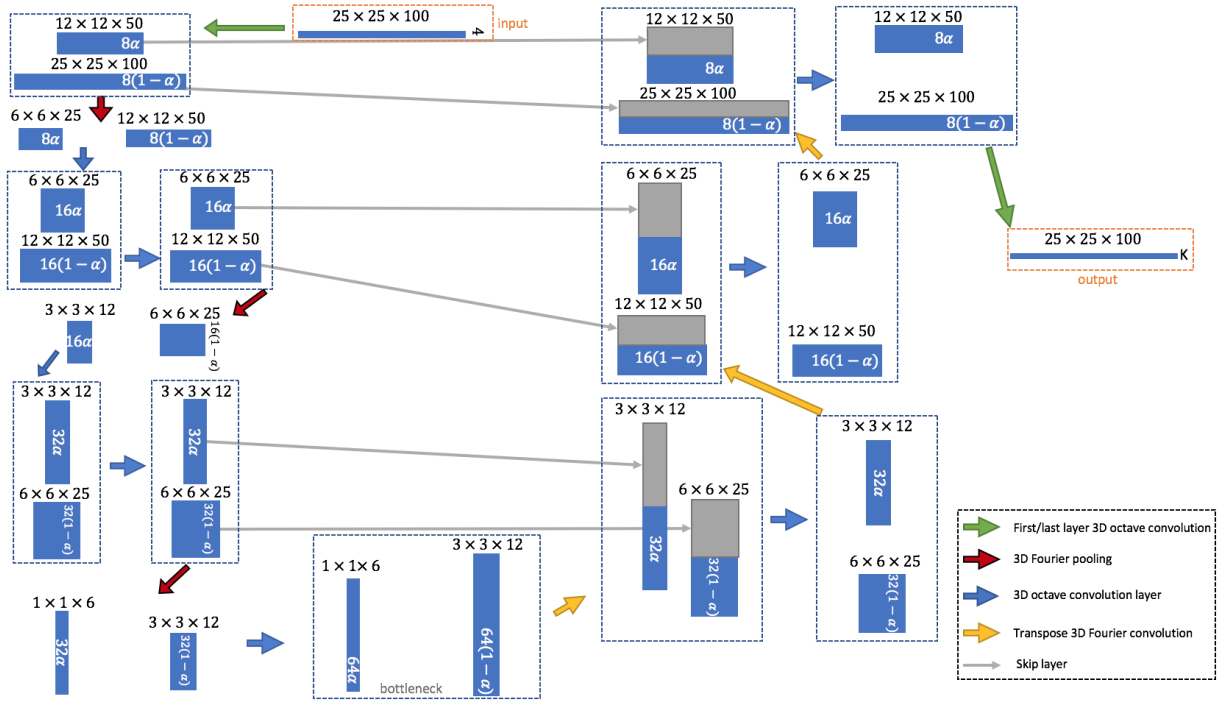

Figure 3.5: 3D Fourier Encoder of Revised W-Net. Arrows indicate an operation. Green arrows are first/last year of octave convolution that breaks full feature map into low and high frequency blocks or combines low and high frequency blocks back to one full feature map. Blue arrows indicate 3D octave Fourier operation. Red arrows indicate 3D Fourier pooling operation. Yellow arrows indicate transpose 3D octave Fourier operation. Gray arrows are skip layers same as those in original W-Net. Each octave block contains two feature maps boxed by dash lines.

transition from one stage to another is by applying one 3D octave Fourier convolution that doubles the number of channels. This continues until we reach the bottleneck. Each set of feature maps obtained right before the pooling layer of each stage is saved. Going from the bottleneck, we have the up-sampling sub-component. In each stage of the up-sampling path, the input feature map from previous stage will be concatenate with the corresponding feature map saved in down-sampling stage, providing extra support for the process of up-sampling. Then one 3D octave Fourier convolution is applied to shrink the number of channels to half. Each stage except the last layer is linked by applying the transpose 3D Fourier convolution

which doubles the size of the feature for all axis other than the channel axis and shrink the channel size by half. Last layer of the up-sampling path combines the high and low frequency components to produce one three dimensional embedding for the original data with K channels. In the actual application, we set $K = 1$ because we only want to get a 3D embedding of the cilia in frequency domain via Fourier infused W-Net.

Data Transform

In the video normalizing and patching section under data pre-processing, we discussed how we prepare our original video data into 100 patches of volume size (100, 25, 25) with channel of 4. It is mentioned that the 100 patches are seen as 100 batches and will be input into our network together at once and we will only input this 100 patches for one video for each batch. The reason why we say the meaning of batch is a bit different is that for the 3D Fourier encoder which produces the 3D embedding for the cilia, we want to see these 100 non-overlapping patches as being independent and identically distributed (i.i.d). This is the characteristic of normal batches. However, going from 3D frequency domain to the inner 2D segmentation network, we want to put these 100 spatial patches back to the original (250, 250) sized frame with time length of 100. So that we can combine all the local spatial information we explored in the 3D Fourier encoder and look at the global spatial picture. In this sense, the 100 patches are not independent and identically distributed (i.i.d) after the transformation.

So the data transform is straightforward. The output from 3D Fourier encoder has size of (100, 100, 25, 25, 1) meaning we have 100 "batches" of (100, 25, 25) volume data with channel size 1. We use `tf.contrib.signal.overlap_and_add(patches, step)` to put back the spatial patches. This function is the inverse of the frame function we used to make the patches and we need to make sure the step is the same size as what we used for patching so that we can get the correct size back and we need to apply it three times as we did for patching. The size of the result feature map should be (1, 100, 250, 250). The time axis now

becomes the channel axis and the 2D segmentation encoder is expect to explore the spatial information for each frame and compress the information along the time axis to produce the correct segmentation map simultaneously.

Loss Function

As is shown in Figure 3.4, we calculate the reconstruction loss and soft n-cut loss as our loss function. However, a little difference from the soft n-cut loss in the original W-Net where they compute the soft N-cut loss between the network input and the segmentation output, we use one frame from the original video unframed to compute the soft N-cut loss against the segmentation result. This is because the soft N-cut loss has assumption that pixels that look similar and are closer to each other are more likely to be in the same class. However, after transforming our data into frequency data with Fast Fourier transform, such assumption does not hold in frequency domain any more. We have to go back to the raw video frame where such assumption holds to calculate the soft N-cut loss. Therefore, our loss function can be written as:

$$
\begin{aligned}
J_{recon} &= MSE(\boldsymbol{X}, \hat{\boldsymbol{X}}) \\
&= ||\boldsymbol{X} - U_{FourierDec}(U_{segNet}(U_{FourierEnc}(\boldsymbol{X}; \boldsymbol{W_{FourierEnc}}); \boldsymbol{W_{segNet}}); \boldsymbol{W_{FourierDec}})||_2^2 \\
J_{total} &= J_{recon} + J_{soft-Ncut}
\end{aligned}
$$

Post-processing

However, the problem of increased variance and large receptive field can lower the localization accuracy despite of the proven success in capturing high level feature representation of the input data for CNNs. Although the soft N-cut loss can help with the localization of object boundaries, in [11], it has been proven that with the applying a fully connected Conditional Random Field (CRF) after last layer of the encoder can largely improve segmentation

with fine-grained boundaries.

Given the fact that we have an input image I with n pixels and a segmentation task with K classes, in fully connected CRF, a segmentation is modelled as a random field $\boldsymbol{X} = \{X_1, ..., x_N\}$ where each X_i takes one of the class values meaning the label of pixel i. Therefore, solving $argmaxP(X|I)$ gives a segmentation mask X given the input image I. $P(X|I)$ is modelled as a Conditional Random Field as Boltzmann distribution [18]:

$$P(X = \hat{x}|I) \propto exp(-E(\hat{x}|I))$$

Therefore, solving $argmaxP(X|I)$ is equivalent to solving $argminE(\hat{x}|I)$. The energy function is given by [11, 18, 19]:

$$E(\hat{x}|I) = \sum_{i \leq N} {}_u(\hat{x}_i|I) + \sum_{i \neq j \leq N} \psi_p(\hat{x}_i, \hat{x}_j|I)$$

where function $\psi_u(\hat{x}_i|I)$ is called the unary potential and can be calculated as $\psi_u(\hat{x}_i|I) = -\log p(u)$ where $p(u)$ is the label probability calculated by softmax layer in the encoder [18, 19].

The function $\psi_p(\hat{x}_i, \hat{x}_j|I)$ is called pairwise potential which uses weighted sum of two Gaussian kernels to measure the penalties when two pixels are assigned with different labels[18, 19].

$$\begin{aligned}\psi_p(\hat{x}_i, \hat{x}_j|I) &= \mu(x_i, x_j)(w^{(1)}k_\alpha + w^{(2)}k_\beta) \\ &= \mu(x_i, x_j)(w^{(1)}exp(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2}) + w^{(2)}exp(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2}))\end{aligned}$$

$\mu(x_i, x_j)$ is label compatibility function. A widely used compatibility function is the Potts model $\mu(x_i, x_j) = |x_i \neq x_j|$. The first Gaussian kernel k_α is the appearance kernel defined in terms of the color vectors I_i and I_j and positions p_i and p_j with the assumption that nearby pixels with similar color are likely to be in the same class. The second Gaussian kernel k_β is the smoothness constrain that is aimed at removing small isolated regions. Parameters θ_α, θ_β, θ_γ and weights $w^{(1)}$, $w^{(2)}$ are all learnable [18, 19].

3.4 Revised Fourier U-Net for Partial Video Reconstruction - Second Model

In previous section, we discussed the revised Fourier W-Net model with the application of Fourier operation and octave blocks. This in theory allows the model to be able to work entirely in the Fourier frequency spectrum and retrieve rich information encoded in the frequency domain. Since the texture of cilia yields distinctive frequency in 2D spatial domain and the natural beating behavior yields special frequency in time domain, combining spatial information and time information simultaneously and working directly in 3D with the revised Fourier W-Net model should help to classify the distinctive cilia frequency and give us a mask of areas that correspond to cilia.

However, there are a few problems with such design. First of all, the intention of classifying the frequencies that belong to cilia might result in losing the information of shape and location of the cilia area in each frame. Secondly, following the convolutional theorem, we replaced the normal convolutional operation with the point-wise multiplication of the whole feature map and kernel of the same size. Since the data we use is magnitude data which means, ideally, we should be dealing with positive numbers all the way through the entire network. The activation function ReLu should not affect the backward signal a lot. This means that when the back-propagation process starts, the gradient can easily explode or vanish because of the successive multiplication along the way.

Therefore, the partial video reconstruction model is proposed to deal with such problems. The overall concept of this approach is to use a supervised approach to introduce information of shape and location of the cilia and ask the network to reconstruct the corrupted version of the original video given the original video where any area what is not cilia is masked out. We still work entirely in frequency. However, changing the information in the original video will affect vastly on frequency domain and such affect is not some simple masking but change magnitude across the frequency domain. Therefore, we cannot treat this problem as a classification problem but a reconstruction problem to reconstruct the masked video. Since we are working entirely in frequency domain, we need to get real and imaginary parts separately in the output instead of just magnitude.

3.4.1 Data Preparation

In order to adapt to the new model, we need to prepare another version of data. First of all, we need to preserve both the real and imaginary part of frequencies. Remember Heisenbergs uncertainty tells us to make patches out of the original video in order to ensure some degree of localization. Therefore, the same patching approach is applied first and Fourier transform is used to transform the original video patches to frequency domain. However, instead of calculating the magnitude of frequency, we keep real and and imaginary part and store them as our data. To conclude, after reshaping our original videos to 250×250 for each frame, we patch along the first two axis, namely making patches on each frame with no overlapping and then patch along the time axis with 50 steps overlap. Again, we treat the patching along time axis as channels. Therefore, we have patched data of size $(100, 25, 25, 100, 4)$ (i.e. $(numberOfPatches, W, H, D, inChan)$).

For the input data, the raw video is patched with the aforementioned approach. Then Fourier transform is apply and DC components are shifted along D, W, H axis afterwards. At this point, we have data with size $(100, 100, 25, 25, 4)$ and with complex numbers. From here, we can simply extract the real and imaginary part with `.real` and `.imag` to numpy

arrays. Then, we'll have an array of real part of the frequency with size $(100, 100, 25, 25, 4)$ and an array of imaginary part of the frequency with the same size. The array of imaginary part is appended to the array of real part. Therefore, the final data size is $(200, 100, 25, 25, 4)$.

For the ground truth output data that we need for the supervised approach, we need to use the mask data to mask out the non-cilia part in the video. Then the same approach is used to get the final output data size of $(200, 100, 25, 25, 4)$.

In terms of the way used to normalize the data, we choose to normalize the real and imaginary part separately.

3.4.2 Revised Fourier U-Net for Reconstruction

The detailed introduction of the original U-Net network structure will not be discussed here since we have come across the same topic couple of times in the previous sections. Here, we will discuss the minor change in the Fourier convolutional operation that we made to adjust it to work with our new goal and data. Similarly to the previous Fourier convolutional operation, we use convolution theorem to convert the traditional convolution to Fourier convolution, namely, using point-wise multiplication on the frequency domain. This time, however, instead of utilizing how magnitude of complex numbers react to multiplication, we directly look into how the real and imaginary part communicate during multiplication of two complex numbers. For example, assume we have two complex numbers $M = a + bi$ and $N = c + di$, multiplication of the two complex numbers will be:

$$\begin{aligned} MN &= (a + bi)(c + di) \\ &= (ac - bd) + (ad + bc)i \end{aligned}$$

Recall we stored the real and imaginary part of the patches. Therefore, assume the data with size $(200, 100, 25, 25, 4)$ is F, we already know that the first half is real part and

second is imaginary part. Therefore, we can easily restore the original complex frequencies by $F[:100,:,:,:,:] + F[100:,:,:,:,:]i$. Then, in the network, we can initialize two kernels to represent real and imaginary part. Then calculate the real and imaginary part separately like above and again, store the results by appending the imaginary part to the real part, resulting in the output array of same size $(200, 100, 25, 25, 4)$. This way, we avoid the trouble that Tensorflow does not support complex numbers for back-propagation. Note here we excluded the batch axis since it is handled the same way as discussed before and is irrelevant to what we address here.

Chapter 4

Experiments

4.1 Dataset

Our dataset consists of 325 medical videos. Each has different frame size and time length. Only primary cilium is included in our dataset. Our dataset contains several different types of cilia motion with different beat frequency and different beat pattern. This includes cilia with healthy beat pattern as well as unhealthy pattern including wavy or stiff pattern. An example of each beat pattern is provided in Figure 4.1.

Remember our model uses 3D model. The reason why we want to work in the 3D dimension is that we want to work in the spatial and temporal information simultaneously. Figure 4.2 show the three orthogonal panels for cilia with those three beat patterns. It clearly shows the difference of the temporal information between those cilia of different patterns. For the healthy cilia, temporal information in the cilia area shows strong repeated periodic response that is highly coordinate across the whole cilia area. Both the beat frequency in time domain and the texture in spatial domain will both have strong peak in the frequency of their separate dimension. Therefore, this supports our theory that by working directly in 3D domain, the strong frequency that correlates to 1D temporal information and 2D spatial information will be mapped to one 3D frequency. Therefore, by analysing and looking for the strong characters in the 3D frequency domain should be able to help us pin point the areas that belong to cilia. Hence, this should help the network to produce the segmentation mask for cilia area. However, for the wavy cilia, the response in temporal domain is somewhat periodic but not coordinate which causes the chaos in the kymograph images. In terms of the stiff cilia, there is little response in the time domain.

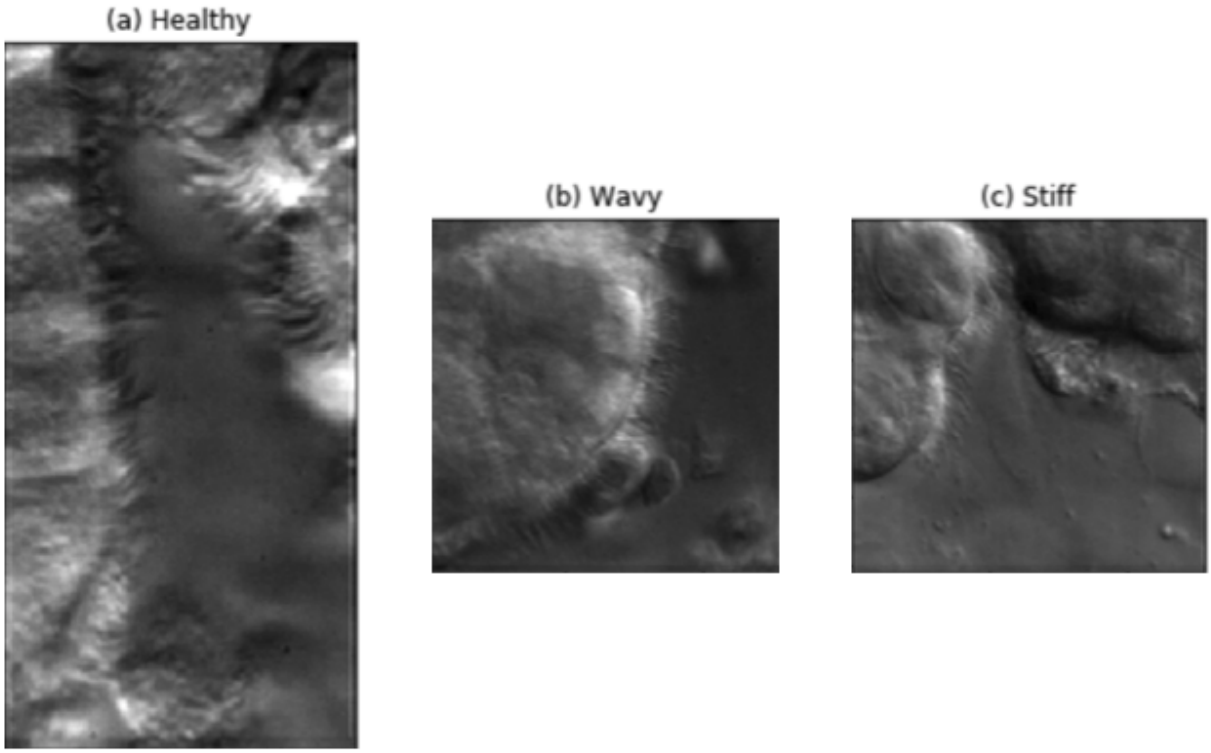

Figure 4.1: Three Types of Cilia Motion

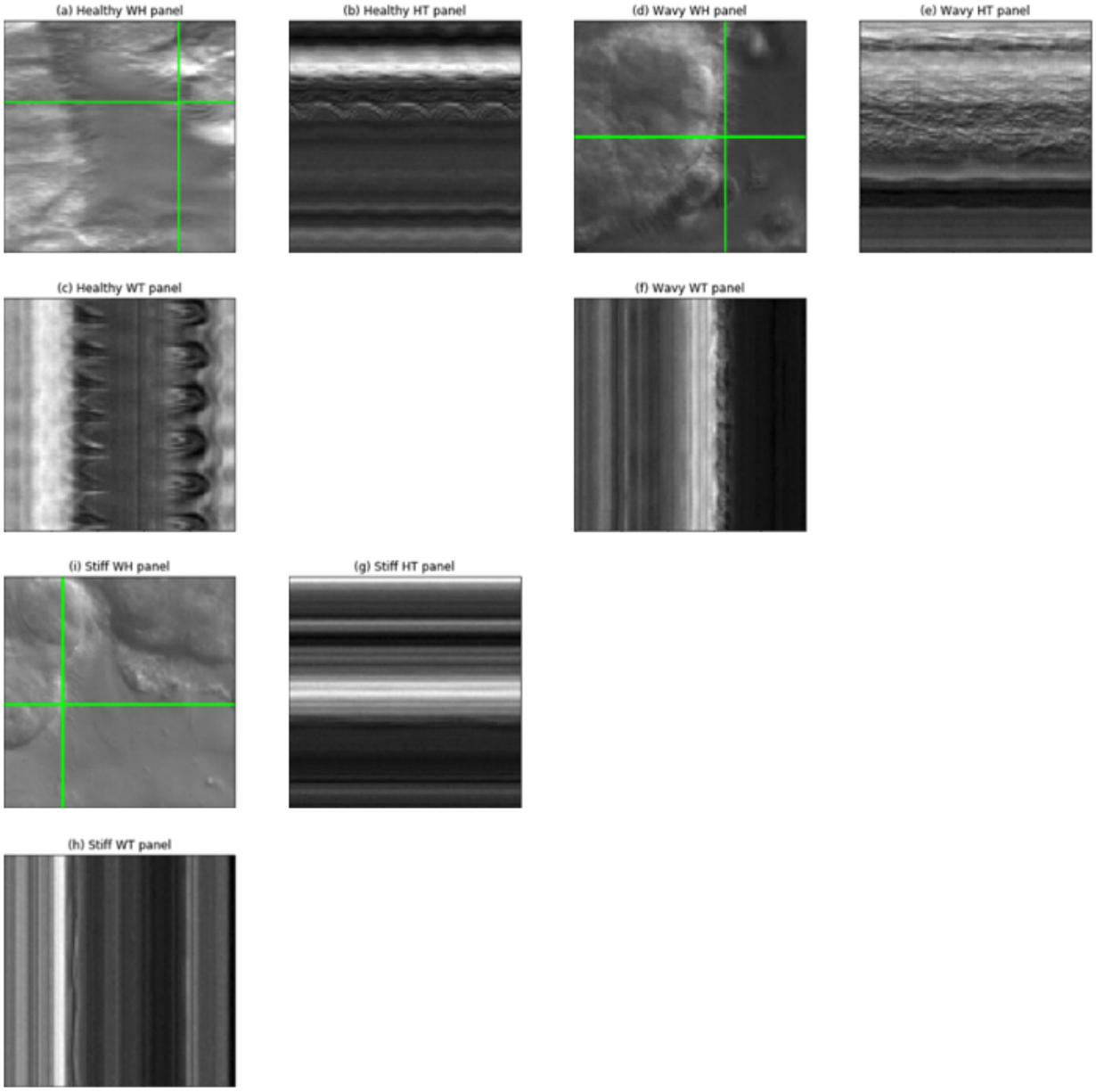

Figure 4.2: Three Orthogonal Panel Slices of Cilia. (a) one frame of healthy cilia example; (b)/(c) corresponding health cilia kymograph; (d) one frame of wavy cilia example; (e)/(f) corresponding wavy cilia kymograph; (i) one frame of stiff cilia example; (g)/(h) corresponding stiff cilia kymograph.

The Fourier frequency data also supports our theory. Figure 4.3 shows an example of three sample points from a video in the dataset with healthy cilia. There are chances that the cell itself will move along with the cilia. Therefore, the change of root part of cilia and the change of the cell in time domain will be similar. Hence, looking at time domain alone cannot solve our problem. In the case when we use 3D Fourier transform, we consider signals both in spatial and temporal domain. Since cross correlation is the way of comparing similarity between signals, Figure 4.4 shows the cross correlation results between the patches that contains those three sample points. Although, as excepted, the difference along slice of time domain is minor, the slice of cross correlation in spatial shows significant difference between root and cell patches.

4.2 Results and Discussion of Problems

The results are not as satisfactory as we hoped for. There are couple of different problems that can lead to such results.

4.2.1 Overfitting/Underfitting

For the pixel-wise classification model (first model), with the standard design discussed in 3.3.2, the training loss goes down nicely as shown in Figure 4.5. However, the testing loss goes to infinity quickly after 2 to 5 epochs. From such behavior, it seems that there is a significant overfitting problem.

This can be explained by the design of the revised Fourier W-Net for segmentation. In our theory, the convolution operation is done by point-wise multiplication. If we denote feature map after convolution operation as Y^n $(n >= 1)$ where n is the number of layer and denote W^n $(n >= 1)$ as the kernel used in each convolution layer. Note that X is the input to the model and $act(.)$ is activation function.

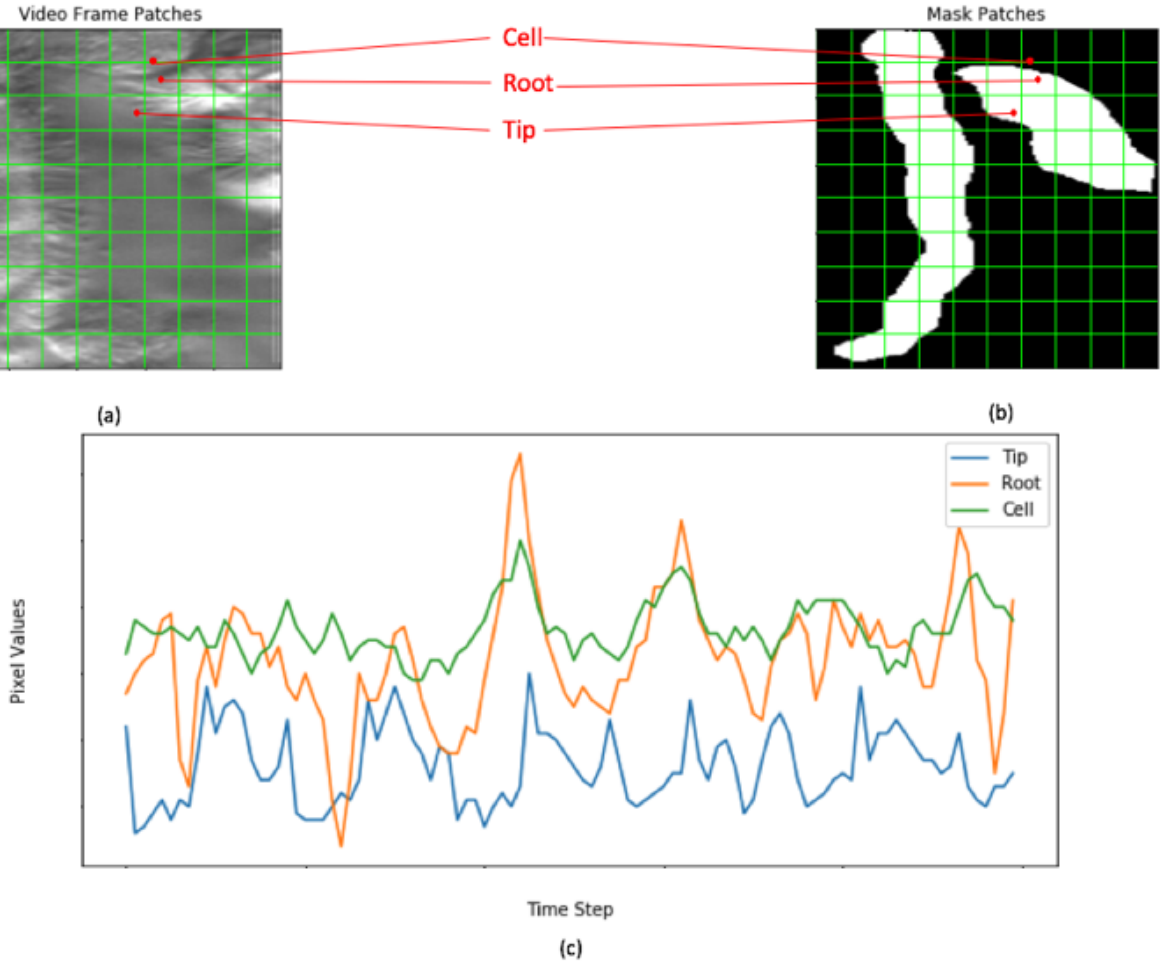

Figure 4.3: An example of why time domain only does not work. (a) One video frame with patch grid and three sample points; (b) Corresponding mask with patch grid and three sample points; (c) Change of pixel value for the three sample points

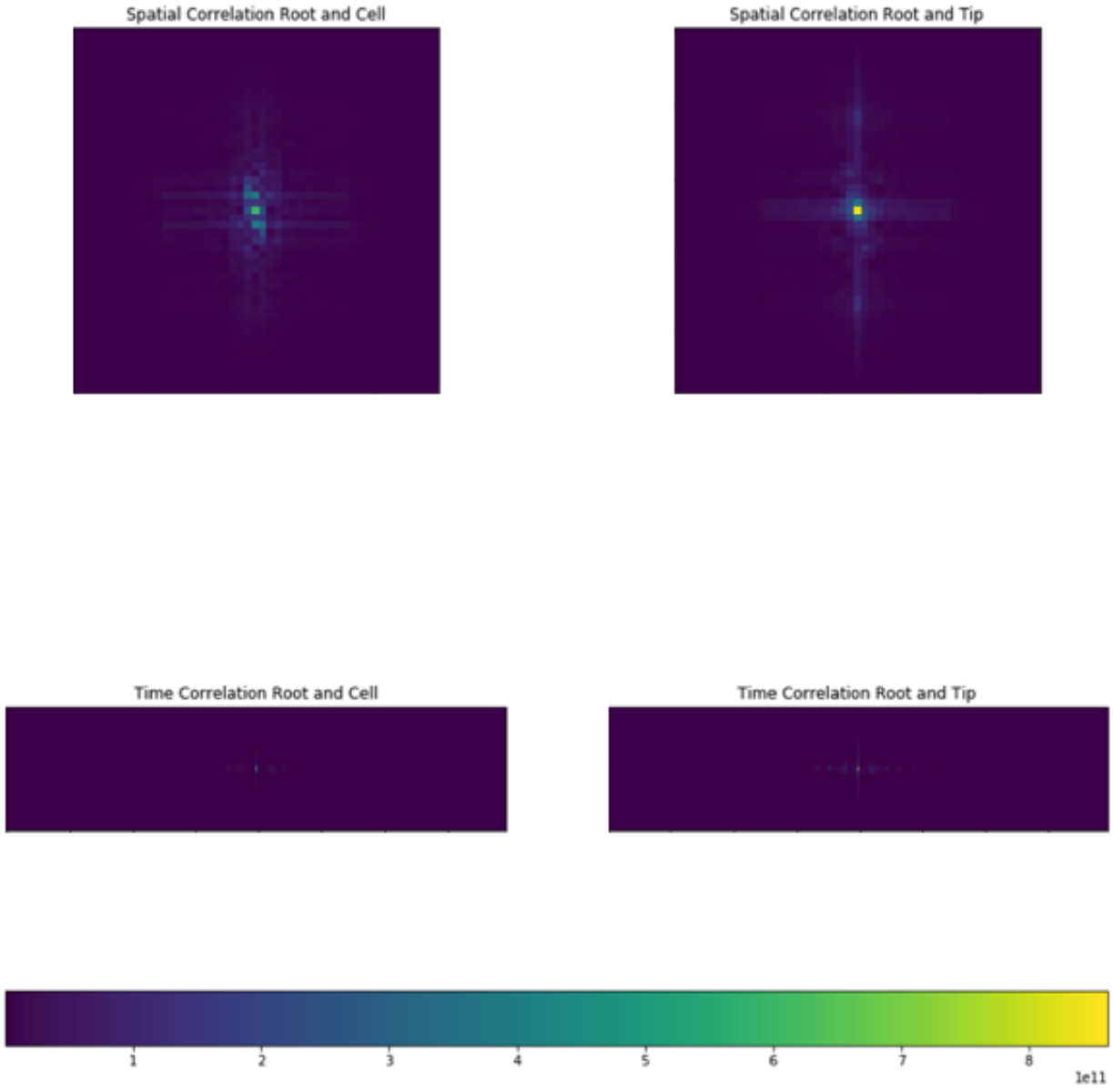

Figure 4.4: Correlation between patches containing the three sample points: An example of why 3D Fourier transform works better

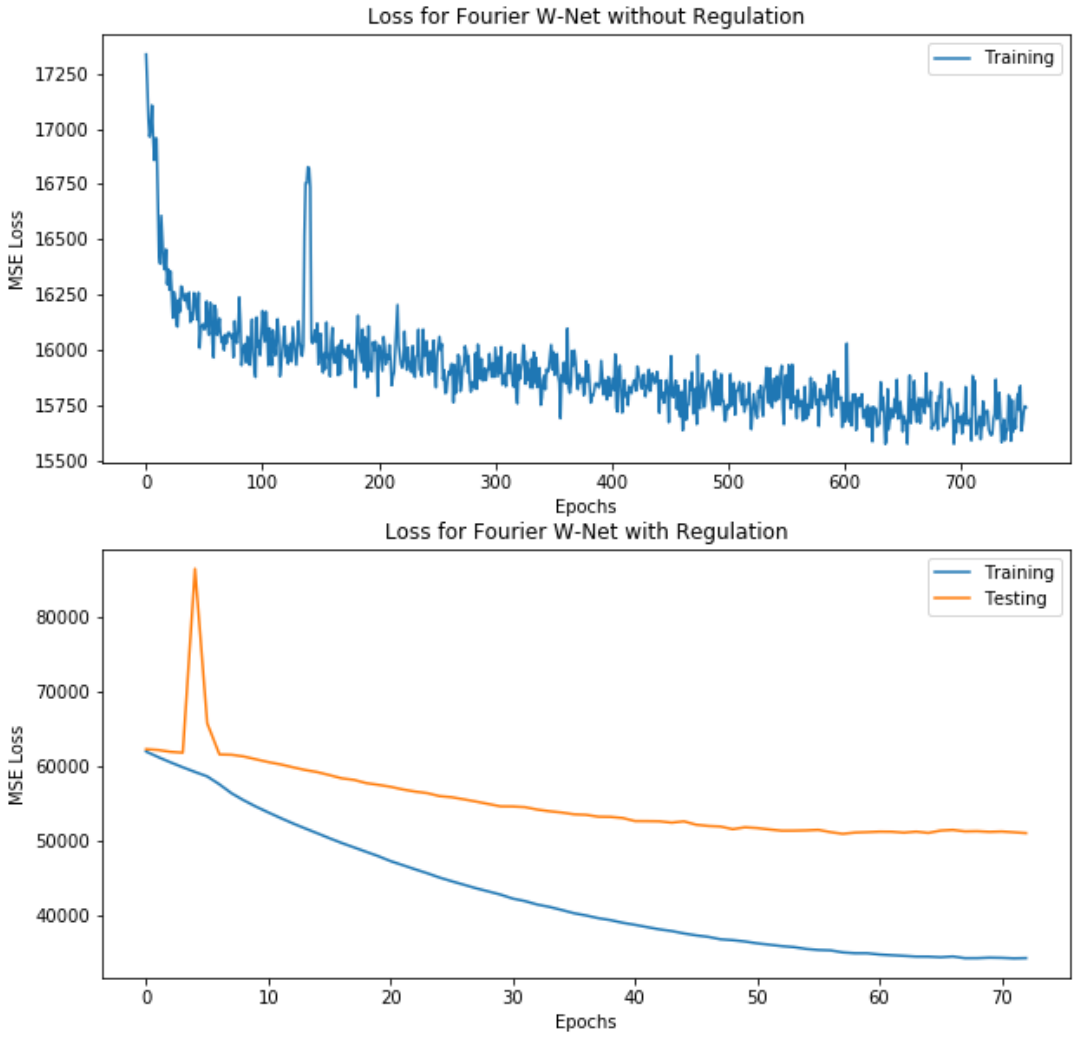

Figure 4.5: MSE Loss of Revised Fourier W-Net for Segmentation. Upper: Without weight regulation; Lower: With weight regulation

$$\hat{Y^1} = act(XW^1)$$
$$\vdots$$
$$\hat{Y^n} = act(Y^{n-1}W^n)$$
$$\text{i.e. } \hat{Y^n} = act(...act(act(XW^1)W^2)...W^n)...)$$

Note that data we use is magnitude which means it is all positive. Also, we use ReLu as activation function. Since ReLu returns exactly the same input when it is positive (i.e. $y = x$ when $x >= 0$), this means $\hat{Y^n} \approx XW^1W^2...W^n$. Therefore, when it comes to back propagation and partial derivative, the successive multiplication of the kernels makes the entire back propagation process unstable. The vanishing and exploding gradient problem will be extreme. Hence the infinity of testing loss quickly after 2 epochs.

$$\frac{\partial \frac{1}{n}\sum_n (Y_i - \hat{Y_i^n})^2}{\partial X} = -\frac{2}{n}\sum_n [(Y_i - \hat{Y_i^n})\frac{\partial \hat{Y_i^n}}{\partial X}]$$

We tried to apply L2 weight regulation after each layer and it solved the infinity testing loss problem as seen in 4.5. However, the change of loss starts to move significantly slowly within 100 epochs. Figure 4.6 shows the results of these two approaches from training stage. However, for overfitting problems, the training results should be extremely good while testing results are poor. In Figure 4.6, however, it seems that the model still works poorly in producing segmentation mask. Therefore, this can very much be a underfitting problem instead.

4.2.2 Patching and Assumptions for Input

Our pre-processing requires to make 3D patches of the videos. Each input to the model is the total 100 patches of a video. However, while inputting these 100 patches to the model as

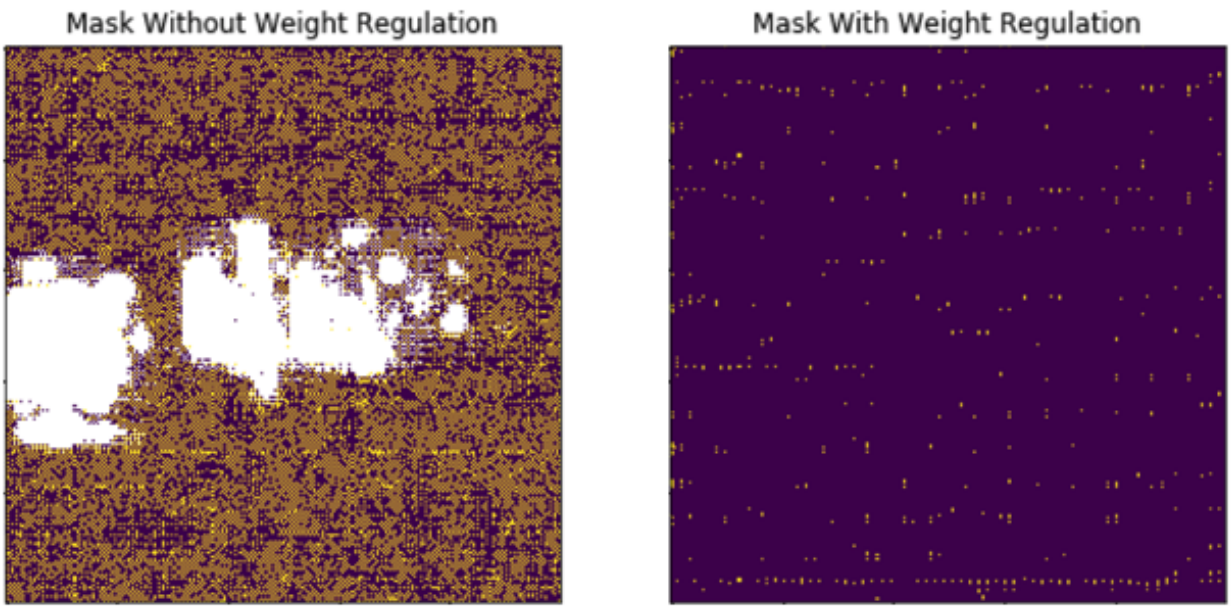

Figure 4.6: Mask result of the classification model with and without weight regulation

a batch, we actually make an assumption that we are treating the individual patches as i.i.d. (i.e. independent and identically distributed). However, this does not hold. The patches with cilia and those without are not identically distributed because such different characteristics inspired our model to begin with. And patches of a video are not independent either since pixels in an object are highly correlate.

However, without patching the video, our model could suffer from the problem in the poor attention for localization. In addition, because of the requirement of point-wise multiplication for Fourier operation, the kernel size is the same as the feature map. This means, without patches, it takes up huge amount of memory. Also, before patching, we first resized all the video frames into same size (250,250). This means the new pixel value has different meanings from the original ones. This can also cause the artifacts in the results. A better approach is to use the same patching methods on the original video completely removing the resize steps.

There is another concern with regard to the patchy behavior in the result from the first model with weight regulation. The segmentation result might not caused by the patching yet the segmentation result might still be the segmentation in frequency domain. Recall the design of the first model, there is no explicit transform from the frequency domain to spatial domain. The data transform block only resized data making time axis into channel axis so that the inner 2D segmentation W-Net can squeeze down the information along time axis. However, this way, even with the power of deep learning, still is not equivalent to inverse Fourier transform which maps from frequency spectrum back to spatial spectrum. Therefore, the result can still be the mask in frequency spectrum. Further investigation into the results is still required to fully understand the problem.

In addition, the video data was treated as a big 3D volume where we apply 3D fast Fourier transform to learn spatial and temporal information simultaneously. However, the time axis can be conceptually different to the two spatial axes. It remains as a question that whether it is appropriate to treat a video as a 3D volume.

4.2.3 Ways of Squeezing along the Channel

In our design, Fourier convolution operation is done the same as normal convolution operation where, after applying one kernel, the entire feature map is added up together along the channel axis to produce a feature map with one channel. The change of channel size is then controlled by the number of kernels used in the process. However, doing the same way may not hold for Fourier convolution. The Fourier convolution is defined by the convolution theorem. However, such addition along the channel in frequency domain is not entirely applicable.

4.2.4 Memory Issue and Overparameterization

Last but not least, one of the biggest issue for Fourier convolution is the huge number of parameters because we need to make kernel the same size as the feature map following the convolution theorem. Comparing to tranditional convolutional neural network which uses much smaller kernel size (i.e. 3×3), Fourier convolution definitely requires much lager memory. Therefore, it also suffers from the problem of Overparameterization which could also be one of the reason for the severe overfitting/underfitting problem

Chapter 5

Conclusion and Future Work

In conclusion, we established a theory for introducing Fourier transform into a deep learning model. We developed basic Fourier convolution operation that applies convolution theorem to allow us work entirely in the frequency domain. We introduced octave convolution to the Fourier convolution to help with the memory issue. In addition, revised Fourier W-Net for segmentation is proposed. This model works in frequency domain and this model is completely unsupervised with the application of n-cut to avoid small clusters. Another revised Fourier U-Net for partial video reconstruction is proposed to deal with some potential problems in the previous model.

There are still a few problems in the theory and models which cause bad performance in the loss as well as the final segmentation from the network. It includes problems in four main perspectives. First of all, the overfitting/underfitting of the model may be the most important problem to look into. Secondly, the assumption on the data might also cause the poor result. The patching method used to reduce the size might lead to the artifacts on the result. In the design of Fourier convolution operation, the way of squeezing down along channel axis can be different. Further proving for how different ways of squeezing influence the parameter training process should be explored in future work. Last but not least, the memory issue caused by the huge number of parameter should be dealt with in the future work.

Future work should start with further analysis of the problem and results. Several problematic behaviors of model should be identified from the segmentation result. Other better algorithm should be developed in the future work to avoid the successive multiplication in

the model. Assumptions on the data should be carefully addressed to fit better in the model. New methods to overcome the problem of artifacts in the result are also desired in the future work.

Reference

[1] L. T. Haimo and J. L. Rosenbaum, "Cilia, flagella, and microtubules," *The Journal of Cell Biology.* 91(3):125-130, 1981.

[2] F. Ijaz and K. Ikegami, "Live cell imaging of dynamic behaviors of motile cilia and primary cilium," *Microscopy (Oxf).* 68(2):99-110, 2019.

[3] L. Vincensini, T. Blisnick and P. Bastin, "The importance of model organisms to study cilia and flagella biology," *Biol Aujourdhui.* 205(1):5-28, 2011.

[4] P. Satir, L. B. Pedersen and S.T. Christensen, "The primary cilia at a glance," *J. Cell Sci.* 123:499-503, 2010.

[5] A. M. Waters and P. L. Beales, "Ciliopathies: an expanding disease spectrum," *Pediatr Nephrol*, vol. 26, no. 7, pp. 10391056, 2011.

[6] J. Raidt et. al., "Ciliary beat pattern and frequency in genetic variants of primary ciliary dyskinesia," *European Respiratory Society*, vol. 44, no. 6, pp. 1579-1588, 2014.

[7] S. Quinn et. al., "Novel use of differential image velocity invariants to categorize ciliary motion defects," *Biomedical Sciences and Engineering Conference (BSEC), IEEE*, 2011.

[8] C. Lu et. al., "Stacked Neural Networks for end-to-end ciliary motion analysis," *arXiv*, arXiv:1803.07534, 2018.

[9] J. Zhang et. al., "Learning long term dependencies via Fourier recurrent units," *International Conference on Machine Learning (ICML)*, arXiv preprint arXiv:1803.06585, 2018.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *MICCAI*, pp. 234-241, 2015.

[11] X. Xia and B.Kulis, "W-Net: A Deep Model for Fully Unsupervised Image Segmentation," *arXiv preprint arXiv:1711.08506*, 2017.

[12] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence* 22(8):888905, 2000.

[13] R. M. Haralick, K. Shanmugam, et al., "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 6, pp. 610621, 1973.

[14] P. Saisan, G. Doretto, Y. Wu and S. Soatto, "Dynamic Texture Recognition," *CVPR*, 2001

[15] D. F. Elliott, "Handbook of Digital Signal Processing," *Academic Press*, Ch 7, pp. 527-631, 1987.

[16] Y. Chen et. al., "Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution," *CoRR*, abs/1904.05049, 2019.

[17] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, No. 8, 2000.

[18] M. T. T. Teichmann and R. Cipolla, "Convolutional CRFs for Semantic Segmentation," *CoRR*, abs/1805.04777, 2018.

[19] P. Krähenbühl and V. Koltun, "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials," *CoRR*, abs/1210.5644, 2012.