Automatic Tongue Contour Segmentation using Deep Learning

by

Shuangyue Wen

A thesis submitted in partial fulfillment of the requirements for the
Master of Applied Science in Electrical and Computer Engineering

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

Abstract

Ultrasound is one of the primary technologies used for clinical purposes. Ultrasound systems have favorable real-time capabilities, are fast and relatively inexpensive, portable and non-invasive. Recent interest in using ultrasound imaging for tongue motion has various applications in linguistic study, speech therapy as well as in foreign language education, where visual-feedback of tongue motion complements conventional audio feedback.

Ultrasound images are known to be difficult to recognize. The anatomical structure in them, the rapidity of tongue movements, also missing segments in some frames and the limited frame rate of ultrasound systems have made automatic tongue contour extraction and tracking very challenging and especially hard for real-time applications. Traditional image processing-based approaches have many practical limitations in terms of automation, speed, and accuracy.

Recent progress in deep convolutional neural networks has been successfully exploited in a variety of computer vision problems such as detection, classification, and segmentation. In the past few years, deep belief networks for tongue segmentation and convolutional neural networks for the classification of tongue motion have been proposed. However, none of these claim fully-automatic or real-time performance. U-Net is one of the most popular deep learning algorithms for image segmentation, and it is composed of several convolutions and deconvolution layers.

In this thesis, we proposed a fully automatic system to extract tongue dorsum from ultrasound videos in real-time using a simplified version of U-Net, which we call sU-Net. Two databases from different machines were collected, and different training schemes were applied for testing the learning capability of the model. Our experiment on ultrasound video data demonstrates that the proposed method is very competitive compared with other methods in terms of performance and accuracy.

Acknowledgments

First of all, I would like to express my sincerest gratitude to Prof. WonSook Lee for providing a precious research chance to work with state-of-the-art techniques for tackling real-life problems. And the patient guidance as well as the valuable technical help during my study, are crucial for enabling this research. Her encouragement and precious help are the keys to the success of my research.

I am very grateful to Dr. Hilmi Dajani for being a thoughtful and caring professor, an elaborate and knowledgeable lecturer. He is one of the first people that led me into the area of machine learning and taught me to way to do proper research.

And I would like to thank Ph.D. Candidate Mr. Hamed Mozaffari, from whom I received huge help. Also, for the thorough teaching within and without my research area, and a vigorous collaboration which speeds up my productivity much. He is not only a forerunner in the research area who's willing to share but also a friend that you share joy and sorrow with. I would also like to thank my lab mates in CG++ group, Wei Ruan, Nan Wang, and Kang Wang, who spent with me my best times in Uottawa. Junyao Zhu, Yixiang Li, Xunzhe Wen, Kai Yang, my best buddies that carried me through my hardest times. I'm so lucky to meet all of you guys in my life.

Last but not least, my greatest appreciation to my family: my father, mother, and brother, who unconditionally support and love me, encourage me all the time, without you, my study would be impossible. Thank you all!

Table of Contents

Glossary of terms

USI: Ultrasound Imaging

UTI: Ultrasound Tongue Imaging

PC: Principle Component

MLP: Multi-Layer Perceptron

CNN: Convolutional Neural Network

FCN: Fully Convolutional Network

RNN: Recurrent Neural Network

GD: Gradient Descent

SGD: Stochastic Gradient Descent

MSD: Mean Sum of Distance

List of Figures

List of Tables

1 Introduction

The tongue is a highly mobile, deformable and precise organ, routinely deploying, in extremely rapid succession, various and subtly different movements manifesting great combinatorial flexibility. Visualization of tongue gestures during these various dynamic processes is the first step in the treatment of speech sound disorders [1]. Ultrasound imaging of tongues has been applied not only to tongue disorder treatment [2] but also to such areas as language training [3] and silent speech interface [4], enabling speech communication with non-audible signals [5]. When learning to perform some complicated activities or the subject is impaired to perform such activity accurately, the visual feedback will be crucial in this case, for the subject to regulate their behavior to achieve better performance for this specific activity. Current research using Ultrasound imaging as feedback mainly aimed to help one acquire a second language [6] [7], as well as part of therapy in treating speech-related diseases [8][9]. The importance of visual feedback is significant for humans to regulate their behavior [10] [11], which is also known as 'motor learning principle' [9]. With proper visual feedback, the human subject can learn the changing patterns and postures with higher efficiency, which could be a painstaking process that takes up a great amount of time without such feedback. It also grants the teacher a direct way of evaluating students [10], enabling a better understanding of how the subject performed.

For phonetic and speech training, a mid-sagittal view of Ultrasound imaging is widely adopted, as it displays relative backness, height and the slope of various regions of the tongue [8]. A human subject makes a speech consisting of specific phonemes. Then an Ultrasound video showing tongue movement corresponding to those phonemes is brought to the human subject. With such a scheme, it reported improved performance for treating speech sound disorder [8] [9], and also aiding speech production and perception [6] [7].

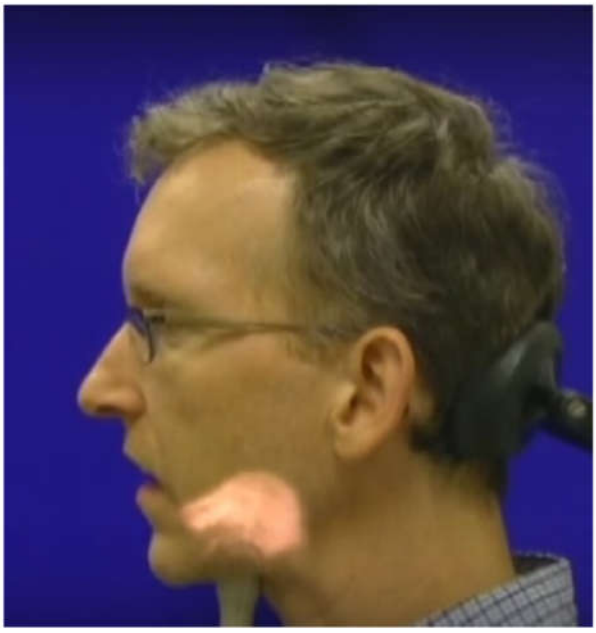

Figure 1-1 Ultrasound overlay video frame [7], in which such overlay is produced manually using photo editing tools. And the goal of our research is to achieve similar performance automatically

Ultrasound imaging is endowed with a reasonably rapid frame rate (typically 60Hz to 120Hz), allowing researchers using a conventional ultrasound machine to capture subtle and swift movements of the tongue during speech production [12], [13] in a two-dimensional (2D) view (midsagittal with/without coronal). An example of capturing a midsagittal view of a tongue is shown in (a). A 3D version with a costlier 3D ultrasound transducer has a slower capture process [14].

In order to quantify and analyze tongue shape, the tongue dorsum is usually defined as the brightest and longest continuous region across these images and segmented in a sequence of images. A variety of techniques based on a range of algorithms have been proposed for tongue contour extraction and tracking, including active contour models [2], [15], [16], and machine learning-based methods [17]– [19].

Although many methods for tongue contour segmentation, extraction, and tracking in an image and/or video frames have achieved acceptable and reproducible results, real-time and fully automatic performance has seldom been reported so far. The difficulty is due to the inherent characteristics of ultrasound imaging: low signal to noise ratio, high speckle noise corruption, image artifacts, and others [18]. Manual labeling is usually required both for initialization and for

cases when the tongue is not clearly detectable or when consecutive frames are very different due to rapid motion [20].

1.1 Objectives

The objectives of this thesis include:

1. Design and implementation of a real-time fully-automatic tongue segmentation and tracking algorithm. Since most of the published tongue trackers hardly achieve the capabilities of running in real-time or require manual initialization, these are huge drawbacks since both of them will narrow the application of the algorithms, and trained personnel for manual initialization only will be a disaster for commercialization. Thus, a new segmentation and tracking algorithm combining accuracy and running efficiency as well as the automation is in need.
2. A new method is proposed for real-time fully-automatic tracking that gives a pixel-wise label for tongue based on a shallow (comparatively) convolutional neural network. The shallow network design reduces runtime cost with the accuracy of features extracted not diminished much.
3. After the training of the neural network architecture to segment the classifier responsible for the tongue, the postprocessing is applied to extract the tongue skeleton information, which gives the possibility to study the shape and curvature characteristic of the tongue.

We used python and Keras library for the implementation of the deep learning algorithms and the OpenCV library [28] for image processing implementation.

1.2 Contributions

For now, the state-of-the-art methods have many limitations, for example, all the traditional active contour based model or other similar methods are semi-automatic, and the accuracy is limited. The details will be given in chapter 2.3. The newly emerged machine learning models are

developing fast for their adaptation and high performance, deep learning-based methods, are especially favorable among them.

Deep learning techniques are promising as solutions for this challenging task and have indeed been explored over the last few years for ultrasound tongue imaging. Encoder-decoder architectures were tried for segmentation in [21]– [23]. Tongue contours were extracted automatically using deep belief networks (DBNs) and deep auto-encoders [24], which are probabilistic generative models comprised of several Boltzmann machines (RBNs) stacked on top of each other. More recently a convolutional neural network (CNN) was used for tongue gesture target classification [25].

Deep convolutional networks are widely used for classification in general or medical imaging, and very modern architectures such as "fully convolutional networks" (FCN) [26] and "U-net" [27] enable image segmentation.

Our own work on ultrasound tongue contour extraction and tracking were motivated mainly by U-net. We modified the original U-net architecture to work with small amounts of training data while extracting tongue contours and tracking them in real-time without manual intervention. We call it "simplified U-net" (sU-net). Our experiments demonstrate superior performance and accuracy compared to previous methods of ultrasound tongue imaging. To the best of our knowledge, there have as yet been no previous attempts to apply U-Net architecture for ultrasound tongue image sequences. And we can claim our proposed sU-net is one of the best models for tongue extraction in terms of accuracy and time-efficiency.

1.3 Thesis Structure

The thesis is organized as follows:

- Chapter 2 presents the background of ultrasound imaging principles, including the formulation of the images, the characteristics of the ultrasound image, as well as acquiring and interpretation of the ultrasound tongue image.
- Chapter 3 reviews the tongue segmentation or contour extraction from ultrasound images. The methods from the classic active contour models to the most state-of-the-art

machine learning models will be put into a survey, their features, pros, and cons will be compared.

- Chapter 4 starts the discussion of the system implementation. First, the overview of the whole system will be given, followed by the technical details of all sub-sections of the system, including the details for CNN which composed the core part of this research, as well as the implementation platform and tool. Also, the necessary post-processing techniques are introduced.
- Chapter 5 focuses on implementing the contour extraction system enabled by a machine learning model. The acquirement of two different databases and their partition for training are reviewed. Necessary steps for implementing& tuning the network are also presented into details. Finally, the training process of the model is illustrated at the end of the chapter.
- Chapter 6 showed the training result and the evaluation of it. The post-processing results, as well as the evaluation metric, are listed first. This is followed by a detailed comparison of the performance. In the evaluation, the performance is compared with other state-of-the-art methods in terms of accuracy, run-time efficiency, and generality.
- Chapter 7 summarizes all the work done, drawing the conclusion to what has been achieved in the research, and also reviews the system performance. The limitations are discussed, and what can be done to improve the system are also suggested.

2 Background of Ultrasound Tongue Imaging

Before introducing the state-of-the-art methods and our tongue extraction method, it's essential to give a background review of ultrasound image modality: for the B-mode image generation and characteristics, and as well as the tongue structure in it. They are essential aspects in analyzing the ultrasound images. In this chapter, firstly the characteristics and formulation of ultrasound images will be given, followed by the noise feature and priors of B-mode ultrasound. Then, the tongue structure in the image will be discussed, including the capturing process and structural analysis of tongue, as well as the importance of tongue analysis for linguistic and medical research. Finally, a summary would be given in the end.

Ultrasound is described as a certain kind of sound waves that the frequency of ultrasound is higher than the capabilities of human hearing. From the definition, it's easily seen that the ultrasound is of no differences in its physical properties when compared with regular audible sound waves. Although the hearing capability varies from person to person, a typical range of the ultrasound frequencies is from 20 kHz up to several gigahertz. Ultrasound is widely used in a variety of fields. The ultrasonic devices can detect objects as well as measuring distances. Its nondestructive characteristics made it possible for testing products and structures in a friendly manner; it's also used for detecting invisible flaws. Because it's nondestructive property and easiness to apply, it becomes one of the most used devices of imaging the organs and tissue for medical and clinical use, also known as the sonography. Through ultrasound images, we can see into the human body, because of the different characteristics of muscle fiber, fat, fluid and air, the ultrasound images can have abundant information and often one or more trained personnel are required for interpretation of such images, for it's challenging for people to recognize ultrasound images, even harder than MRI or CT.

For a reason discussed above, the segmentation becomes a necessary step in medical imaging, not only can it help people understand the images, but for trained personnel, for it could also obtain qualitative measurements such as the location of monitored organ as well as for quantitative measurements such as area, volume or the analysis of dynamic behavior of anatomical structures over time [29]. Another reason ultrasound images play a crucial role is that they can be produced at video-rate and therefore allow a real-time analysis of moving structures, which is the main

challenge of monitoring human tongue, for the human tongue is such a highly deformable organ. Nevertheless, the acquisition of these images is non-invasive, cheap, and does not impose ionizing radiations while other medical imaging techniques may do. On the other hand, the automatic segmentation of anatomical structures in ultrasound imagery is a real challenge due to acoustic interferences (speckle noise) and artifacts which are inherent in these images.

Here we discuss the ultrasound image segmentation methods, in a broad sense, focusing on techniques developed for medicine. First, the basic characteristics and principle of ultrasound sonography are presented. After that, the ultrasound tongue monitoring will be emphasized, both for its process and application. The third section explains the ultrasonic image segmentation evaluation standards. And lastly, the fourth section will talk about the related works in ultrasound tongue segmentation but will mainly be focused on the traditional methods such as the active contour model. The state-of-the-art methods will be discussed in Chapter 3.

2.1 Ultrasound: Principle and B-mode Imaging

Ultrasound waves are produced by a very small but rapid push-pull action of a probe (transducer) held against a material (medium) such as tissue [30]. Vibrations that are of frequencies less than about twenty-thousand vibration cycles/s are referred to as acoustic wave, which would be audible to human ears, above this is where the term ultrasonic would be employed. For clinical ultrasound, vibrations in the range of 20 000 to 50 000 000 cycles/s are used [29]. The vibration is usually generated either in a continuous wave fashion or in pulses, but most of the modern ultrasound devices use the pulse-wave ultrasound for better sensing quality and accuracy, e.g., the B-mode ultrasound is pulse-wave ultrasound which is favored by hospital and clinics, another kind of the ultrasound is called the continuous wave ultrasound, It is not considered in the thesis as it is less popular and non-relevant to the research topic. Figure 2-1 shows the particle distribution and the pressure wave description of the generated pulses of the ultrasound wave. The upper figure shows the particle distribution along the central axis while the lower figure showed the waveform description of the ultrasound pressure fluctuations.

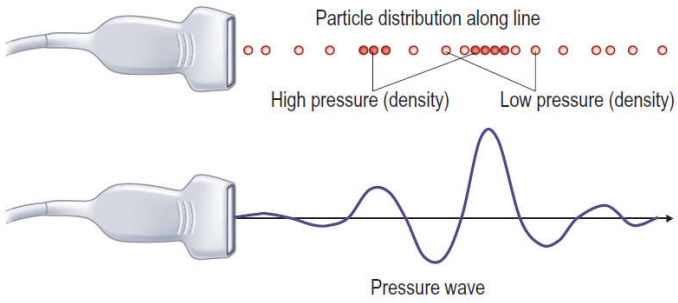

Figure 2-1. The Probe and generated pulses of ultrasound wave during operation [30].

The B-mode images are the most widely used modality, and it's the raw signal adopted for this research topic. In the next, the emphasis would be on how B-mode ultrasound image is obtained. Here is a general description of B-mode ultrasound image formation (a detailed description would be given in the next paragraph): In a standard ultrasound system there are three basic types of data available for analysis: radiofrequency (RF) signals, envelope-detected signals, and B-mode images. A transmit/receive ultrasound transducer receives multiple analog radio-frequency (RF) signals which are converted to digital RF signals and beam formed into a single RF signal. The RF signal is then filtered, and envelope detection is performed to give an envelope detected signal [29]. Finally, the envelope-detected signal undertakes signal processing, for example, log-compression, and some necessary post-processing techniques are required to form a grayscale image. For generating the B-mode image, the interpolation and other image processing techniques would be applied.

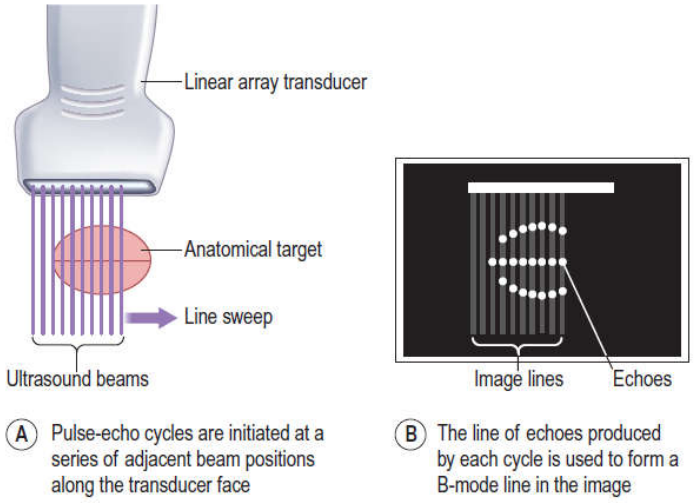

Figure 2-2. The principle of image composition by ultrasound probe [31]

For a more detailed description on sensing and processing for obtaining the B-mode image, the process goes like this: at first the transducer or what is commonly known as the probe, producing a pack of pulse of ultrasound wave, each pack consists of a few cycles of ultrasound waves at the selected frequencies for different penetrating depth (described in the previous section). The pulse would propagate through the human tissues along a beam-shaped corridor at the speed of sound *c* for the particular tissue. Upon reaching a reflecting interface, an echo is produced, which travels back to the transducer, also at speed *c* [31].

The process and propagation of ultrasound would continue so the pulse would reflect at the deep surfaces, generating more echoes at various depth, then go back to the probe, each pulse in a given pack would return in a different time due to the distance they traveled. The total time after transmission of the pulse for each echo to arrive back at the transducer is called the 'go/return' time, which is simply twice the depth (go and return) divided by the speed c, i.e., *2d/c*. As described in [30], in the soft tissues of the human body, the speeds of sound will be within the range of 5% of the average value of sound speed, which is 1540 *m/s*.

The image processor sets this value as default in order to calculate how deep the origin of each echo-producing feature is, Such computation is made from the time of go/return of the submitted signal. In human tissues, there is an almost continuous series of interfaces, and when the ultrasound beam scatters along those interfaces, a continuous series of echoes would be generated, following the pulse transmission. The echoes will form a line, and this line is used in producing the B-mode line in the image, the image brightness along the B-mode line denotes the with the strength of the echo, and the distance down the line displays the arrival time of the echo. Once echoes reach the maximum depth and get received after returning, another identical pulse-echo cycle as the previous one is initiated, the beam is emitted to the adjacent position with regards to the first one, when this process continues, the continuous B-mode line will be generated in the image.

This process is well illustrated in Figure 2-2. *Figure 1-1*Figure 2-2 (A) shows how the signal is sampled by the transducer: the ultrasound beams would go through a line sweeping stage and form many B-model lines. The two-dimensional B-mode image is composed of this individual, adjacent lines. When this process repeats along the entire length of the transducer face, a large number of adjacent image lines (typically 256 [30]) will be formed, which form a 2D B-mode image. Once the last line of the return signal for the image has been formed, this repetitive process will initiate

to form a continuous series of frames. The ultrasound images are displayed as they are formed, resulting in a real-time display of moving images as shown in Figure 2-2 (B). In the simplest image forming process, each line is formed by a single pulse-echo cycle as illustrated in Figure 2-2 (B). The travel time of the ultrasound pulses and echoes are finite; its go/return time from the reflecting target draws a line to the fundamental limitation of ultrasound imaging systems. This time to form each image line is the 'go/return' time to the maximum depth. Large imaging depths result in long line times. When coupled with many image lines, this leads to a long frame time and low frame rate. A high frame rate will be needed to image a target which is moving rapidly, e.g., the heart, it may also be necessary to restrict the image depth and the width of the image. Figure 2-3 below showed the architecture of the B-mode imaging processor, the formulation of a B-mode ultrasound image will go through stages including beam formulation via a transducer, the digitize in the receiving stage and storage of memory. After the formulation of the beam is completed the image could also be extracted and would be put into the cine memory, the final output is made after the post-processing stage.

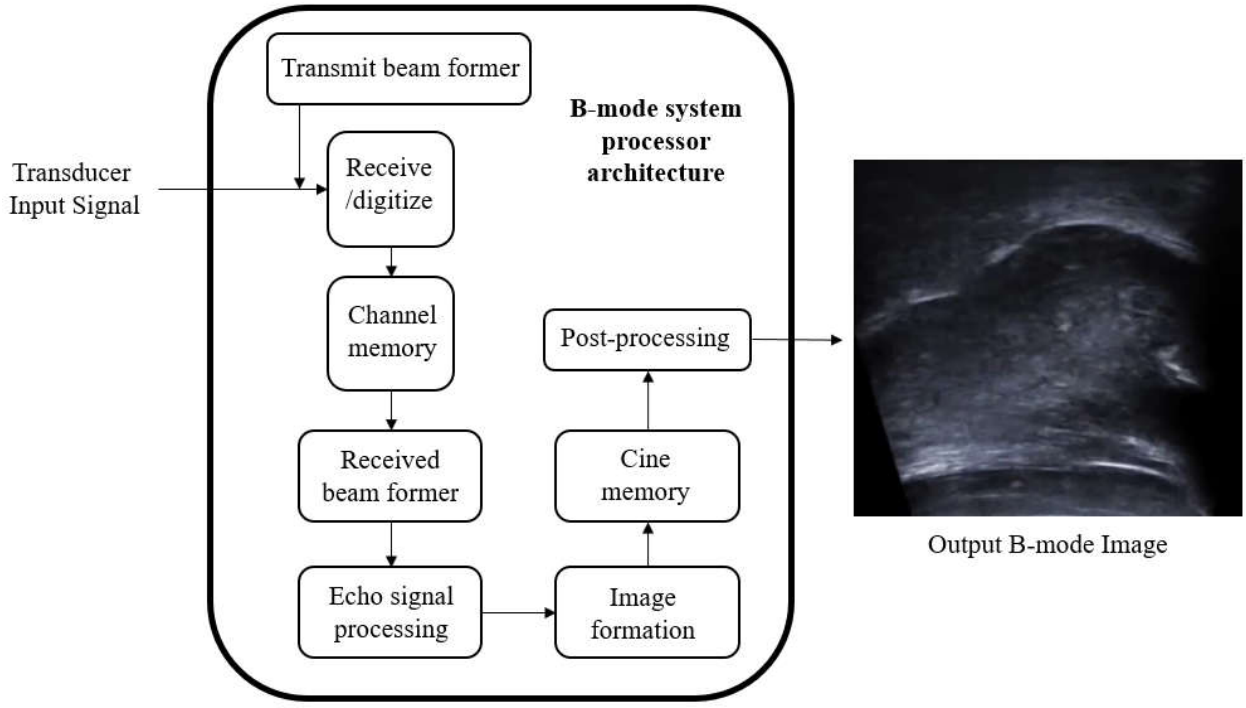

Figure 2-3 The block diagram of B-mode ultrasound device (re-made from [31])

2.2 Characteristics of Ultrasound Image

Various advantages of using medical ultrasound have led to increased use of ultrasound in not only its traditional area of application, such as diagnostics and clinical research but also in emerging areas such as image-guided interventions and therapy. Thanks to the recent advances in transducer design, many aspects of the transducer have greatly improved since then, for example, the resolution, both spatial as well as temporal. The digital sampling system makes the imaging more efficient. The portability and cost, make it easier to commercialize, etc., which means that the quality of information from an ultrasound device has significantly improved [39]. Thus, the emergence of interest has been growing since then, that tried to understand how to implement one of the most classic image processing tasks, the image segmentation, to ultrasound images. As many types of research in this field showed, while in other clinical imaging areas (e.g., X-ray computed tomography (CT) and magnetic resonance imaging), the application of general image processing methods may suffice, when it comes to the ultrasound data, due to some more challenging problems introduced previously, it's common that the customized methods, which tried to model the certain physics in tongue shape and ultrasound image pattern and gain a more successful result.

But the ultrasound image segmentation remained a challenging task, it's obvious that the content of image is in a bad quality compared to other image modalities, the contour is usually not connected, and the distribution of the brightness or grayscale would not share a straight-forward pattern, many sophisticated algorithms and models had been proposed for solving the problems, e.g. [2] [18] [40], a systematic review would be given in the next section.

In this section, the inherent feature of the ultrasound and the challenging part for segmentation would be discussed, for it directly impact the performance. There are characteristic artifacts which make the segmentation task complicated such as attenuation, speckle, shadows, and signal dropout; due to the orientation dependence of acquisition that can result in missing boundaries [41]. The contrast is also very low, and many organs described in the previous sections would appear or disappear during an ultrasound examination, which would surely increase the difficulty of segmentation task and make the system less robust.

2.2.1 Speckle Noise of Ultrasound Images

Ultrasound images have characteristic granular noise patterns; it's called the speckle noise. It's an inherent feature for coherent imaging such as radar imaging, optical coherence tomography images, as well as clinical ultrasound imaging. Many efforts had been put into the analysis of this effect, which is highly related to the signal, both the radar and ultrasound community had it as a major subject of investigation. While we say the speckle "noise," it's intuitively non-desirable, and texture appearance of the observed speckle does not correspond to the underlying tissue structure. However, the local brightness of the speckle pattern does reflect the local echogenicity of the underlying scatterers [41].

For this reason, speckle noise, in general, could be undesirable and ought to be reduced for improving image quality, but in fact, the speckle noise is only a qualitative description of the ultrasound image, some of it is the results of the reflection of some small tissue structures and thus carry information. Thus, from the perspective of segmentation, it's both acceptable to choose to remove it or utilize it for the information it contains. The received echo signals are obtained by a coherent summation of echo signals from many ultrasound scatterers [41]. Speckle has a random and deterministic nature as it is formed from backscattered echoes of randomly or coherently distributed scatterers in the tissue [42] [43]. It has been shown that the statistical properties of the received signal and thus of the echo envelope, depending on the density and the spatial distribution of the scatters [42]. Several distribution families have been proposed in the literature. For the special case of a large number of randomly located scatterers, the statistics of the envelope signal shows a Rayleigh distribution [43]. The speckle would be called "fully developed" under such condition. Both Rice distribution and K-distribution had been put into modeling such special scattering conditions, whereas the Rice distribution aims to compensate the coherent component of scatters caused by the regular and consistent part within the tissue [42] [44]. And the K-distribution tried to mitigate the low effective scatter density [45]

Unluckily, both distributions failed to model the scatter characteristics accurately. Effort on generalizing these distributions via mathematical models [45] for getting a better result is made. Although the different scattering conditions could be coped well with some of the new models, the computational complexity is always beyond expectation even when the comparatively simple models [46] are proposed. And it is worth mentioning that in clinical ultrasound imaging systems,

a log-compression is performed to control the dynamic range of the image, which could introduce new noises.

2.2.2 Priors in Ultrasound Image

• Gray level distribution: As noted in the previous section, the generalized Rayleigh distribution-based model of speckle has proved a reasonable choice. Such Rayleigh distribution was used in the contour extraction based approaches of, e.g. [47] [48], and in the segmentation based algorithm in, e.g. [49] [50]. Other gray level distribution models have also been studied and used in the ultrasound segmentation literature: for instance, the Gaussian [51], exponential [52], Gamma [53], and Beta distributions [54].

• Intensity gradient: The motivation for using intensity gradient as a feature comes from the computer vision literature where based on a photometric model, high-intensity gradients or equivalently intensity step changes/discontinuities in intensity are frequently associated with edges of objects. In ultrasound segmentation, it is, therefore, appropriate to use intensity gradient as a segmentation constraint if the goal is to find acoustic (impedance) discontinuities. Strictly speaking, one is also assuming that there is an approximately constant intensity on either side of the boundary (i.e., speckle has a low amplitude). It has proved a popular constraint with or without prior speckle reduction [41]. To reduce the effect of speckle before gradient estimation, median filtering [55], morphological smoothing [56] have been used, and others like speckle reduction, thresholding based adaptive smoothing and prefiltering techniques [41] have also been used.

• Phase: The local phase provides an alternative way to characterize structure in an image which has been used for example in [57]. Measuring local phase, or rather phase congruency over spatial scales, provides a way to characterize different intensity features in terms of the shape of the intensity profile rather than the intensity derivative magnitude; for example, a step edge has a phase value of 0 or $\frac{\pi}{2}$, an intensity ridge has a phase of π. Thus, the phase has been suggested as a more robust feature for acoustic boundary detection [41]. Speckle also has a phase signature so appropriate spatial scales have to be selected.

• Similarity measures: Within the current discussion of B-mode ultrasound segmentation, similarity measures have been used to provide a displacement estimate as a constraint in spatial-temporal analysis rather than a standalone segmentation feature (e.g., used in image registration).

The literature on ultrasound-ultrasound similarity measures is very small. The sum-of-squares difference (SSD) is used [41].

• Texture measures: Although ultrasound texture patterns are intrinsically dependent on the imaging system, they characterize the microstructure of the tissues being imaged. The distribution (coherent and/or random) of scatterers and their relative sizes to the wavelength of the incident ultrasound pulse, produces different texture patterns. As noted in the section on speckle noise, ultrasound speckle statistics/measures have been successfully utilized to help with medical image processing [41]. General statistical texture analysis methods have been used with some success in this field. One potential advantage of texture analysis is that most texture measures are based on statistical patterns of the intensities and thus independent of the physics of the imaging system [58]. Thus, in segmentation, where the aim is often to characterize the imaged object rather than necessarily characterize its true physical properties.

2.3 Tongue in Ultrasound Image

2.3.1 Acquiring the Ultrasound Tongue Images

The anatomy structure of the neck-lower chin area is complex; however, a sound knowledge of normal anatomy is not essential to perform ultrasound diagnose and signal to capture of the tongue. When setting the frequency of the probe, according to the clinical ultrasound operating guide [37], it's suggested that a typically 7.5–17 MHz would be suitable for examining, depending on the size of the patient's submandibular region. A 9 MHz probe would be suitable for most adult, a 12 MHz probe for a child's submandibular region or for a slim adult, whereas a 5 MHz probe might be required for deeper structures in a large neck [37].

The ultrasound gel is crucial for sonography because the ultrasound wave has such high frequencies, it would attenuate quickly in the air, making the energy damping too severe that the sensor could not detect strong enough signal for interpretation of the structure within the submandibular region. After implementation of gel onto the probe, a valid exam could be undertaken. The best position for the examination is with the patient's neck extended, and some additional force ought to be applied to make a solid contact of probe and submandibular, with the chin in the midline. It is helpful to have a routine technique for scanning the neck comprehensively.

[37]. The whole experiment for one subject should take around 10 minutes, to allow the subject to find the best location and adjust the image to best quality, then pronounce the given set of words. The experiment set is illustrated in Figure 2-4. The device used to capture the signal is from ULTRASONIX, and during the experiment, the subjects are asked to pronounce some certain words, and the device would record the video showing the tongue movement in association with the word he/she pronounced.

Figure 2-4 The experiment scene of gathering ultrasound tongue images

2.3.2 Structure of Tongue in the Ultrasound Images

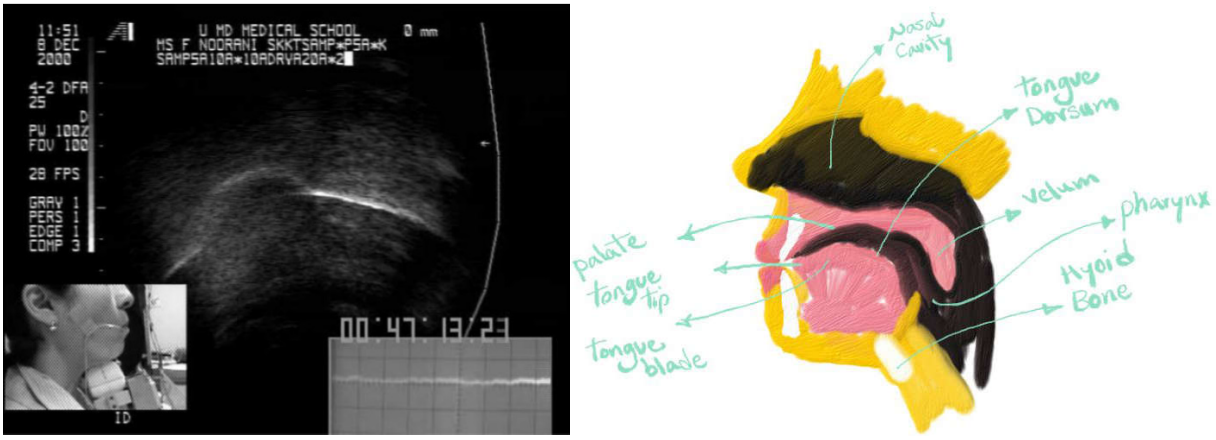

Figure 2-5 The typical ultrasound machine output (left) and the possible structure for the tongue area [38]

When the orientation of Image changes, the received image also changes, there are many possible orientations showing inside the submandibular area. But the most used orientation in the field of research is the sagittal view of the tongue. Figure 2-5 contains an ultrasound image acquisition scenario (left). The image on the left showed a typical setting of ultrasound tongue examination. The bright line is the tongue contour. Because that the Water bolus in front of mouth has air above it. The posterior tongue is visible. [20] The bottom-left corner of the left image showed the probe setting under the submandibular. A schematic of the tissue which could be seen with ultrasound is shown on the right in Figure 2-5. The figure on the right is the Longitudinal panoramic sonogram of the submental region, demonstrating the structures those are possible to be seen in an ultrasound examination. The ultrasound beam is directed upward into the tongue. Unlike an X-ray, which produces a 2D view of the 3D head, the ultrasound image depicts a thin slice of tissue (1.9mm) [20].

In the ultrasound tongue image, the tissue-air interface would reflect almost all the power of the sound wave. That's to say, the deeper structure beyond the tissue-air surface would not be imaged, and a black area would show because that interface would give a strong reflection. As a result, there would be no ultrasound energy propagating to the palate, for this reason, the palate wouldn't be able to be seen. A similar effect is caused in the sagittal image by the mandible and hyoid bones, which would refract the sound waves before they reach the tongue surface. [20] The resultant image is called an ''acoustic shadow'' due to the characteristics of the bone density. It would usually happen at both edges of the image (because the mandible and the hyoid bones are close to the root of the tongue, and below the tip of the tongue when the tongue is rested), it can thus obscure some import parts of the tongue or obscure the motion pattern of the tongue. A common practice would be to push the transducer further up, press tightly against the jaw. It would usually make it clear for the tongue tip to show up in the image, although it is tempting, unfortunately the downside would be a disastrous deformation of the posterior tongue, if a high accuracy reconstruction of the tongue model or a precise quantitative analysis is required, the created errors in the tongue shape will bring severe interference.

It's worth mentioning that in addition to the tongue, other structures are sometimes visible in the image, these structures are of less importance, but as they will introduce artifacts in images, a description of them will be listed below as:

Vellum. This is the imaging of the floor of the nasal cavity when a velar sound is made. In detail, the lingua-velar contact would allow a pack of air to pass into the velum and reflect back from the air above it [20].

Palate. Normally the palate is not able to be seen because when the tongue is rested, there would be a hollow area between tongue surface and palate, and the sound is not able to travel through it. But it's valuable to track the palate and insert it on the image as a reference structure. [20] The reason is that for an accurate measurement, the head should be the reference instead of the jaw, and palate have a fixed spatial relationship with the head. The tongue would approximate the palate during a swallow in one or a group of consecutive frames. Therefore, the palate would be able to be seen in those frames. It's worth noticing that the water bolus (items being swallowed) can introduce the artifact.

The hyoid bone. Like the jaw, the hyoid bone would reflect the ultrasound wave and result in a black region in the deeper structure. Very rarely the reflection of the ultrasound of hyoid bone would form a brightness area in the ultrasound image. But even when visible, the hyoid shape cannot be demarcated [20].

Tongue extrema. Pharynx and tip. The two reasons that the tongue tip or root can't be seen clearly is that the sound reflection and refraction. Ideally, the ultrasound wave travels unimpededly through various human tissues and organs until the tissue-air interface in which almost all the energy would be absorbed. For this reason, the when the strong reflection occurs it is rare that structures beyond it could be imaged. Moreover, if an earlier interface causes considerable sound reflection, the tongue surface itself may not be imaged well [20]. The region under the tip of the tongue, there would be a shadow cast by the jaw and the hollow region with air, they may obscure some of the tongue tips unless the tongue tip is physically pressed against the bottom of the mouth. For the back part of the tongue, the hyoid bone would obscure part of the posterior tongue surface (described in the previous paragraph). Typically, measurements of the tongue contour are best stopped at the hyoid shadow since it is difficult to distinguish the tongue root from thyro-hyoid activity [20].

2.3.3 Applications of Ultrasound Tongue

The study in this thesis research aims to help these pronunciation learning problems by providing visual feedback of the tongue surface in Ultrasound. In this process, visualization of both learners' tongue movements, as well as a native speaker's, can be brought to a learner. Then, the learner can see how his/her tongue differs from the native speaker's. Thus this visual feedback helps his/her to modify the tongue movement and speed up the process of learning or rehabilitation.

Ultrasound imaging of the tongue is attractive to researchers because that the analysis of the shapes and moving patterns of the human tongue during speech is important when modeling the vocal tract. For tongue articulation modeling task, ultrasound has started to be used for visualization [33], and even tongue dorsum tracking [34]. Ultrasound modality could image tongue motion at a reasonably rapid frame rate (60 Hz). There are also other reasons for the ultrasound is one of the most prevailing imaging modality thanks to the, e.g., its affordable cost, non-invasive characteristic, and its portability. For this reason, ultrasound becomes the most suited modality for the research topic. Here in this section, some popular domain of application for ultrasound tongue imaging would be discussed.

- ***Linguistics***

One natural form of communication is speech, but wrong articulation in pronunciation causes misunderstanding. The speaking capability is actually the work of a very complex and indigenous system, in which a collaboration of several different organs would take place, and as a result, the pronunciation of a certain word or sentence is made possible. To pronounce a complete word or sentence, a complicated combination of phonemes needs to be articulated correctly in order to do so. And also, the making of sounds varies depending on tongue position and shape which is not visible from outside.

Figure 2-6 illustrates the approximate position of the regions to which tongue can reach when speaking vowels whereas the vertical and horizontal coordinates of the tongue differ during the speech. For the case of consonants, tongue's shape is even more complicated than that of vowels. The linguistic study aims to understand human speech, and to model the movement pattern of the vocal tract is a vital step for doing so. Thus, ultrasound tongue imaging had become an important tool for linguistics.

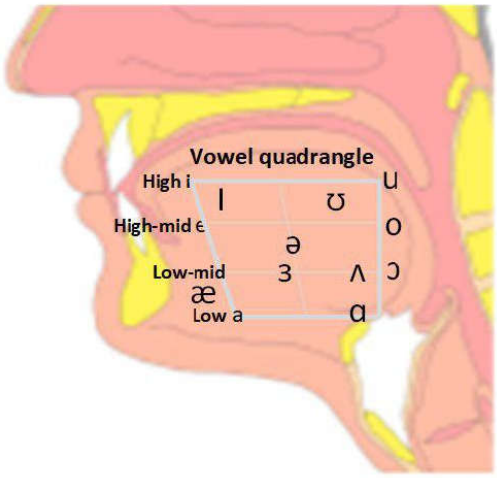

Figure 2-6 The approximate position of the tongue when producing a vowel.

It's worth mentioning that Prof. Jeffrey Mielke from uOttawa had built a linguistic database which contained several thousand phonological patterns in 549 languages and multimodal phonetic data on the production of 130 cross-linguistically frequent consonants and vowels. He had been using such data to conduct research on the topic of automatic transcription and acoustic analysis of two conversational speech corpora and articulatory and acoustic analysis of laboratory speech, using ultrasound imaging. And many similar databases had been built that contained UST, MRI sequences of specific phonemes important for linguistic study. Including the seeing-speech dataset which is used for this research.

- ***Language Education***

As we discussed, the production of speech is a complicated process. There are many factors hindering people from speaking correctly; the failure of correct speaking mostly happens when people are learning to speak a second language, or they have speech disorders (which would be discussed in the following section). Many applications based on USI are made for language education. Research on the relationship between visualization and education can be found in [35]. Survey on possible applications of utilizing Ultrasound in new language learning is found in [7]. Similar studies can also be found in [8] [9] [36], in which the ultrasound-guided learning of correcting accented speaking as well as help overcome some certain speech disorders are emphasized.

- ***Speech Therapy***

The USI is also widely adopted in speech therapy. There are many kinds of disease that would hinder people from speaking correctly. Normally a language learner learns to pronounce a new word just by hearing. The deaf or partially deaf people would also be speech-impaired because of lacking such feedback. Researches have revealed that visual feedback techniques such as Ultrasound Imaging can help people to learn to speak with better efficiency. Moreover, these technologies have assisted speech therapist for the rehabilitation of many speech disorders. For people that have some certain kind of speech disorders [32], a visualization of tongue motion could help them understand better how they should manipulate the way they speak. In literature, researchers had found that for both adolescents and adults, the help of ultrasound tongue image helped rehabilitation for one of severe hearing impairment, the residual speech impairment [32].

- ***Medical Treatment***

The most traditional and original use of ultrasound is for medical use. The ultrasound signal could travel through the human body and reflect mostly on the air-water surface as discussed in the previous sections. Thus, by imaging the tongue region using ultrasound, one can be diagnosed if he or she had tongue disease including severe ones like cancer or minor ones like blisters.

2.4 Summary

In this chapter, the principle and image generation of B-mode ultrasound is reviewed, the characteristics of the ultrasound image, including their image priors. e.g., Gray level distribution, speckle noise properties, texture measure, etc. The ultrasound images are quite different from normal images from cameras from which the information is preserved well, and the determination or interpretation of them is easy and intuitive. While for ultrasound images, such a task is difficult. Understanding the characteristics of ultrasound images is important for the implementation of the segmentation task.

And also, the importance of segmentation of the ultrasound image had been discussed in this chapter; it can be concluded as 1. helping new practitioners to understand the image content of ultrasound applications better-using visualization. 2. enabling the possible quantitative analysis such as volume computation and automatic shape extraction and diagnose [34] [59]. The

acquirement and interpretation of USI are also discussed, including the possible structures that can be seen in ultrasound images. And in the next chapters, the discussion will be focused on the development of representative methods in ultrasound image segmentation, especially for tongue ultrasound images.

3 Literature Review of Ultrasound Tongue Image Processing

The importance of segmentation of the ultrasound image had been discussed in the previous chapter. The segmentation of organs and other substructures in medical images allows quantitative analysis of clinical parameters related to volume and shape, as, for example, in heart-lung diagnose or brain analysis. Furthermore, it is often an important first step in computer-aided detection pipelines. The task of segmentation is typically defined as identifying the set of voxels which make up either the contour or the interior of the object (s) of interest.

In this chapter, the studies done in this area would be reviewed, the methodologies including the traditional models which use the statistical approaches, the ones trying to build physical models of tongue, and the state-of-the-art machine learning approaches that provide a new aspect in solving the segmentation problem, would be input into discussion and comparison.

In the ultrasound tongue image interpretation, often the classical method is to quantify tongue shape by extracting the contours of the tongue surface, using different kinds of methods, such as, the active contour model [60] [2], image segmentation [18], and a machine-learning based method [21] [61]. However, the signal-to-noise ratio of the ultrasound images is relatively low, and the speckle noise degrades the images by mosaicing continuous structures. Moreover, subjects' ultrasound tongues will vary in different image quality, and the contours in some frames are not always visible during the recording of the ultrasound videos. Out-of-plane motion and speckle noise provoked decorrelation to make this problem even harder. While sustainable effort has been devoted to applying the contour tracking, for a long duration of tracking, it still poses a great challenge due to the aforementioned difficulties, and manual labeling is frequently needed for most of the traditional algorithms to improve the performance of contour extraction. In this section, a systematic review of the traditional methods including the image segmentation and active contour model, as well as the machine learning, especially the deep learning-based methods are to be given. The review would focus on not only the methodologies themselves, but also the pros and cons, and the performance.

In the ultrasound tongue image interpretation, the classical method to quantify tongue shape is to extract the contour, in order to do so, there are several ways. In this section, the methodologies that don't require training or don't have a generative model, are called the traditional method (s), in comparison with the machine learning based methods.

3.1 Snake: Active Contour Models for Deformable Structure Tracking

A widely adopted framework of segmenting the human organs such as the tongue surfaces in the medical image processing active contours or snakes [61], it is the first proposed method that gains success in tackling the highly deformable structures. wherein a discrete set of vertices defining the deformable contour are regulated by an energy minimizing function, expressing that the possible location of the contour is in the vicinity of the image (via external energy functional) as well as constraints on the allowed deformations (via an internal energy functional). Figure 3-1 showed a schematic plot of how active contour model works, first users are required to put a line (green line in the left) near the actually desired contour, this is called the initialization stage. Then after an iterative process, the line will continue to approach the desired contour. The final output is shown on the right. This iterative process will be introduced as follows.

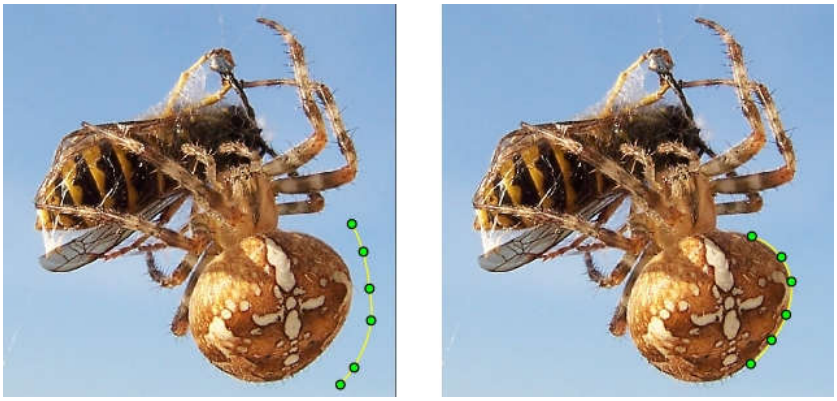

Figure 3-1 The active contour model: a manually initialized line would continue to fit the desired contour by optimization of the specified energy function[62]

More specifically, a snake is an algorithm that minimizes "energy," the specified functions that describe the relationship between two contours, and deformable spline controlled by boundary constraints and image forces that would draw it near to the contour of the object, with "internal forces" that perform as a resistance to the deformation. Snakes could be interpreted as a unique

case of the general technique of energy minimization, that tries to match a deformable model to an image [62]. In 2D, a snake is a curve which can be represented as C (s) = (X (s), Y (s)) where S ε [0, 1]. The curve moves through the image domain to minimize a specified energy function. In traditional snakes, the energy is usually formed by internal forces and external forces is described as

$$E_{snake} = E_{internal} + E_{external} \tag{2.4.1}$$

$E_{internal}$ tends to elastically hold the curve together (elasticity forces) and to keep it from bending too much (bending forces). This energy is defined where Cs and Css represent the first and second derivative, respectively. The snake's tension and rigidity can be controlled by the coefficients α and β.

$$E_{internal} = \frac{1}{2}\int_s \alpha|Cs|^2 ds + \frac{1}{2}\int_s \beta|Css|^2 ds \tag{2.4.2}$$

$E_{internal}$ intends to pull or push the curve towards the edges. Typically, the external forces consist of potential forces. This energy is defined where E_{image} represents the negative gradient of a potential function. This energy is generally the image force where I denotes the image and X = X (x, y) = [x, y]t

$$E_{external} = \int_s E_{image}\big(C(s)\big) ds \tag{2.4.3}$$

$$E_{image}(\mathrm{X}) = -|\nabla I(X)|^2 \tag{2.4.4}$$

Using variational calculus and the Euler-Lagrange the differential equation can be solved. Then, the solution of this force balance represents the final snake position. The differences in the ways the energy function is established will result in different snakes.

$$\alpha\mathrm{Cs} - \beta\mathrm{Css} - \nabla E_{image} = 0 \tag{2.4.5}$$

Although the traditional snakes have found many applications, they are intrinsically weak in four main aspects.

1. They are very sensitive to parameters.
2. They have a small capture range.
3. They have difficulties in progressing onto boundary concavities.

4. The convergence of the algorithm is mostly dependent on the initial position.

3.1.1 Edgetrak Software for Tongue Tracking based on Snake

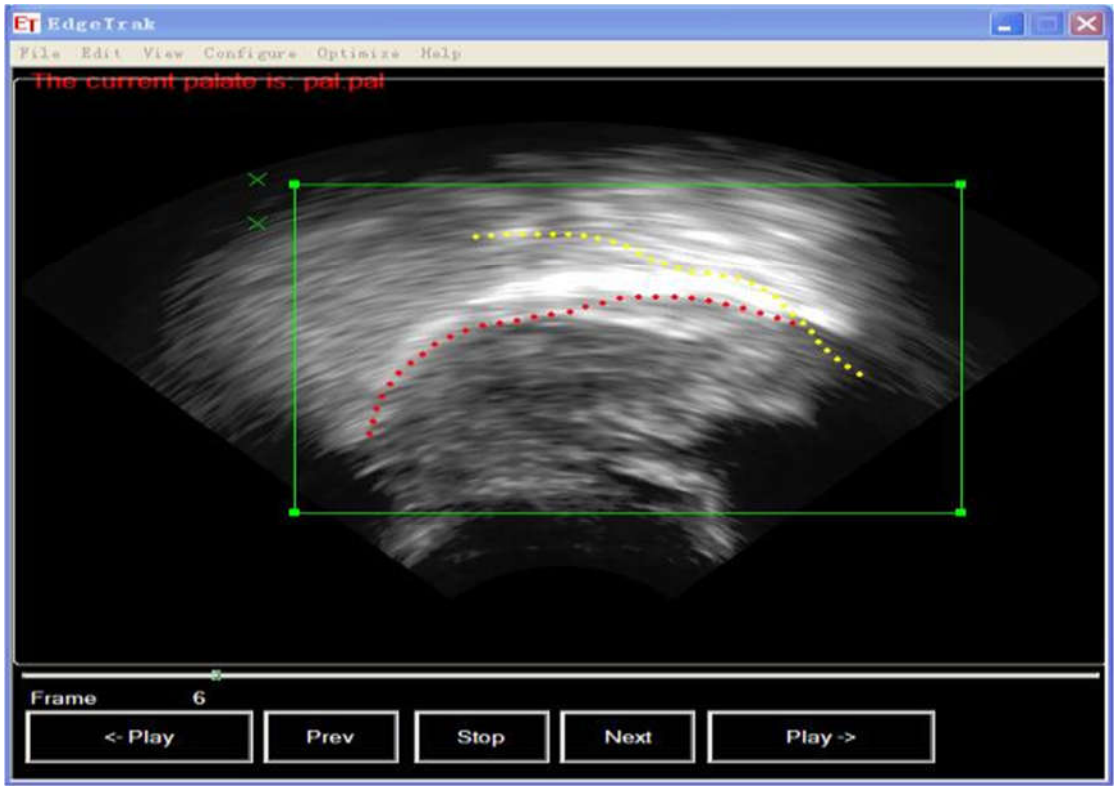

Figure 3-2 Edgetrak software for ultrasound tongue tracking, the method of the Edgetrak is the active contour model, which is used for helping annotation in this research project.

Akgul [16] used a snake formulation for tongue segmentation and tracking in US video sequences and proposed the use of dynamic programming for contour optimization in each frame. The aforementioned methodology [16] was modified in terms of external energy to include a band energy factor that constrains the tongue surface to lie immediately below a bright white band, thereby reducing the possibility of the snake fitting onto speckle noise in the US image. This method is available to the public as the Edgetrak software [60]. Figure 3-2 shows the user interface for Edgetrak software. Edgetrak requires the user to initialize the algorithm by identifying some points near the tongue contour in the first frame and then iteratively minimize the energy functions to optimize the snake for that frame, and this initial guess will be used for the next frame, thereby enforcing some degree of temporal consistency. The yellow line is initiated by a user at a random

place near actual tongue location, and with the Snake process described above, this line will be approximately the actual tongue region, result in this red line in the image.

Although Edgetrak is widely adopted for ultrasound image enhancement and has gained popularity in annotating the ultrasound image, it fails in the presence of rapid tongue curvature increases and occasionally produces tongue shapes that are uncharacteristic of those encountered during the speech. One way to overcome this problem is to enforce global shape constraints on the segmented tongue contour by fitting an active shape or active appearance model [63] [64] of the tongue to the US data, as proposed by [65], in which an active appearance model was built using a training database of segmented X-ray and US images to establish shape constraints respectively and to characterize US image texture in the vicinity of the tongue. Active shape models (ASM) can also be used to constrain and iteratively drive snake optimization [66]. In practice, our experiment with the Edgetrak shows that manual interference was required, when the tongue movement is too rapidly the tracking will fail, and the overall quality of tracking can be improved.

3.2 Statistical Model-based tongue contour extraction

Strong temporal consistency constraints for tongue tracking were introduced by [18]. Figure 3-3 showed the schematic plot of the proposed method. A statistical model-based method proposed in their work, a high-order Markov random field optimization is used for tongue contour extraction. Their method implemented in their publicly available software TongueTrack And They modeled the tongue segmentation problem as a global optimization problem over a higher order Markov random field whose vertices represent the tongue contour evolving in 2D+t and whose edges represent adjacency constraints in space (along with the tongue contour) and in time [18]. The tongue contour "nodes" are connected via "edges" within and across ultrasound frames. The tracking boils down to labeling each node with a displacement vector moving the tongue contour to its optimal position in each frame. The model is mathematically elegant and accounts for spatial (i.e., shape) and temporal (i.e., motion) constraints with the more flexible operation . It also allows future frames to be coordinated with the past ones, which may be useful to reduce certain ambiguities. The proposed method does not require any training data. As a result, it doesn't incorporate much contextual knowledge. It is also computationally intensive.

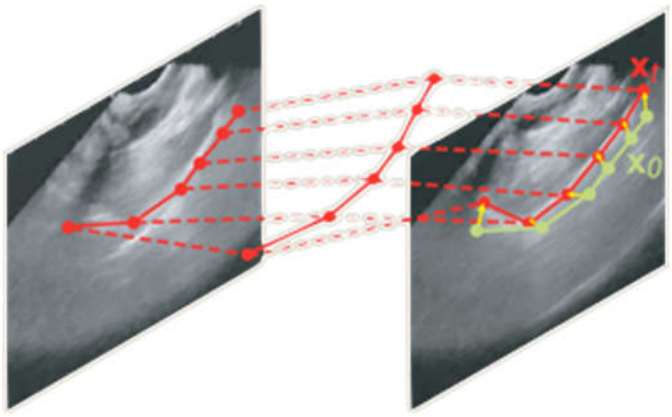

Figure 3-3 TongueTrack software which uses a high-order Markov random field optimization method [18].

3.3 Physical Model-based Tongue Contour Extraction

Instead of relying on heuristics and/or segmented datasets to constrain tongue shape and kinematics, Loosvelt et al. [67] used a biomechanical model as prior information for tongue tracking. Rather than tracking a contour in the image, their method involves fitting the biomechanical model to a set of tracked point features belonging to the entire tongue (not necessarily its surface). Their preliminary results suggest that realistic tongue motion constraints improve tongue tracking when parts of the tongue are invisible in the image.

The proposed biomechanical models consist of computing 3D object deformations using physical laws. In order to have accurate results, the continuum mechanics formulation (conservation laws, matter continuity, etc.) is chosen for the model. Various numerical techniques exist to solve this set of equations. The finite element technique (FEM) is used to compute the equations of continuum mechanics. The discrete geometry for dividing the complex entity into small elements is defined first. Then the problem becomes a mechanical system expressed in ordinary differential equations. Followed by the establishment of the boundary conditions that the system must satisfy. One of the system equations is the constitutive law that depends on the material properties. Finally, the solution of the system is presented in their paper [67]. Their model is shown in Figure 3-4; the left picture shows the manual delineation of the tongue region and the picture on the right is the physical model associated with the delineated region.

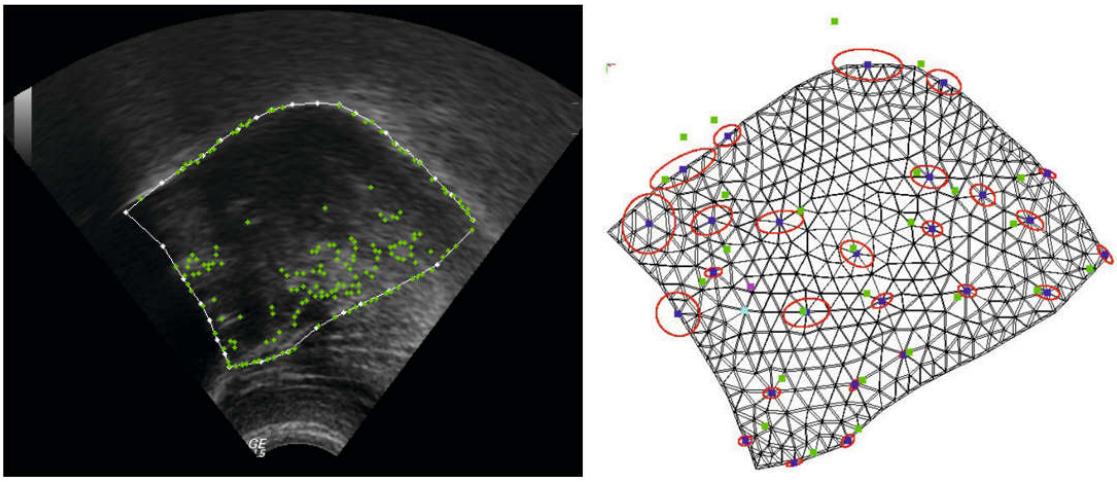

Figure 3-4 The tongue region needs to be fully delineated manually and a physical model formulated by FEM would be fitted to the USI for this method [67]

However, the technique relies on tracking point-based landmarks in US images, a notoriously error-prone task. The manual initialization also put the question into real-time performance. Further validation would be necessary to assess the performance of the method reliably.

3.4 Machine Learning-based Methods

Segmentation is a common task which draws interest in both natural and medical image analysis. It's also one of the most common areas to apply deep learning on, and as such has also seen the widest variety in methodologies, including the development of unique CNN-based segmentation architectures and the wider application of RNNs. To begin this, CNN might be the first structure designed for this task. CNNs can simply be used to classify each pixel in the image individually, by presenting it with patches extracted around the particular pixel. A drawback of this naive 'sliding-window' approach is that the input patches from neighboring pixels have huge overlap and the same convolutions are computed many times. Fortunately, the convolution and dot product are both linear operators, and thus inner products can be written as convolutions and vice versa. By rewriting the fully connected layers as convolutions, the CNN can take input images larger than it was trained on and produce a likelihood map, rather than an output for a single pixel. The resulting 'fully convolutional network' (f-CNN) can then be applied to an entire input image or volume in an efficient fashion. The detailed information of how CNN works and the building block

of CNN will be discussed in section 4.2. Nearly all the state-of-the-art segmentation structure is built upon the CNN/f-CNN. And some classic machine learning based segmentation structures will be reviewed as follows:

3.4.1 FCN

FCN is the abbreviation for the fully convolutional neural network , as noted above, it's the most state-of-art architecture for many computer vision tasks, as a more efficient arrangement of existing CNN structures. Long et al. [68] first introduced the FCN firstly for image segmentation in the year 2015. This network was designed and trained for pixel-wise segmentation. FCN are modified classification CNN. In order to make a network capable of giving segmentation mask pixel-wisely, certain layers have been modified which enable the generation of segmented output maps. The idea of using a modified CNN for pixel-wise segmentation is not new. Matan et al. [69] modified the LeNet network so the strings of digits can be recognized. Other state-of-the-art publications show results on segmentation using CNN structures for making dense pixel-wise predictions.

Detecting the location and size of a lesion in medical imaging is of as great an interest as classification. CNN is usually applied for data classification/recognition producing a single class label output for each image input. Image segmentation differs in that each pixel in an image receives a class label.

The goal of FCNs is to produce "semantic segmentation." Semantic segmentation is to group the pixel in order to give each group a semantic meaning. The output of the segmentation model is of the same size as the original input image with the high order semantic meaning object extracted from the original input image. Such semantic segmentation is made with a stack of convolution layer, which is called by researchers as fully convolutional operation. Figure 3-5 shows such an idea of fully convolution segmentation. After several layers of convolution process, the image will be "shrinking" and at the same time, "deeper," due to the nature of the convolution computation. At last, an output layer would give a prediction of the segmentation mask, in which every pixel is assigned a value representing the group it falls in.

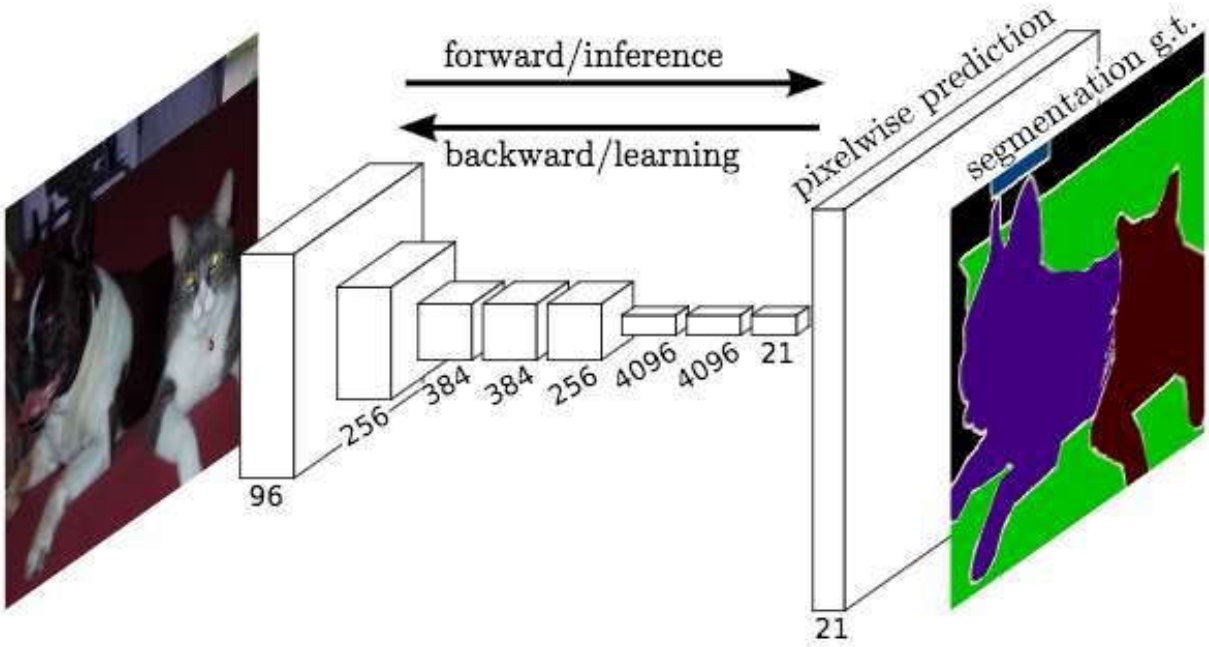

Figure 3-5 Fully-Convolutional Network (FCN) concept plot [70]

The basic idea for a fully convolutional network is that it is "fully convolutional." To specify, all of its layers will be convolutional layers. FCNs don't have any of the fully-connected layers at the end, which are typically used for other CNN structures for classification. Instead, FCNs adopt a convolutional layer at the end as the output layer to classify each pixel in the image.

So, the final output layer will be of the same size (e.g., height and width) as the input image, except for the number of channels because RGB images have 3 channels for representation and segmentation mask often require only one channel (non-overlapping binary segmentation), and number of channels if the number of classes increases . If one is classifying each pixel as one of the different classes, then the final output layer will be height x width x classes numbers. Using a SoftMax probability function, one can find the most likely class for each pixel. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The stride is an important technique for CNN. In the application of convolutional layers, the stride will direct the space between each sample in operation on a pixel grid. For example, if stride is three, there will be three pixels between each subsequent sample.The stride could help to decrease computation complexities, but at the cost of reducing the accuracy, it can be observed that with fewer strides, the segmentation result become better.

Fully convolutional networks are a rich class of models, the modern image classification consists of only a special case. Recognizing this, extending these classification nets to segmentation, and improving the architecture with multi-resolution layer combinations improves the state-of-the-art, while simultaneously simplifying and speeding up learning. [70]

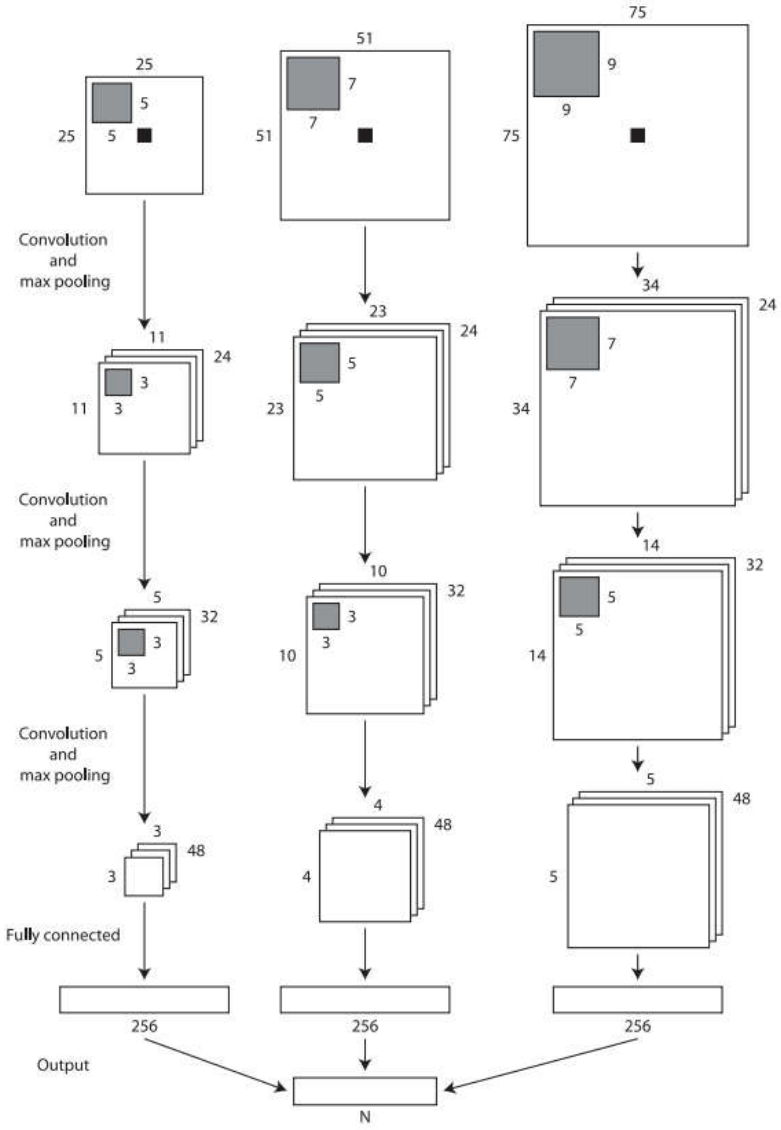

Figure 3-6 The f-CNN structure proposed in [71] for brain image segmentation

The first application of the FCN in medical image processing is with Moeskops et al. [71], who trained a single FCN to segment brain MRI. The structure they proposed is shown in Figure 3-6, which gave us a good intuition for general FCN structure. In their research, it's reported that accurate segmentation results were obtained in images acquired and also with different acquisition

protocols, the result is evaluated on Dice-coefficient metric, a high overall average dice value of 0.735 is reported. However, the disadvantage of their system is that the result is extracted from a limited number of training images. More training images could, therefore, increase performance. In addition, including diverse training data from different data sets in a single training step could generalize the problem and would in this way not require retraining the network. This generalizability over a more diverse dataset might, however, be achieved at the cost of a decreased performance on each of the data sets that are included.

Deep convolutional networks have outperformed the state-of-the-art in many visual recognition tasks. Still, their success was limited due to the size of the available training sets and the size of the considered networks. The breakthrough by Krizhevsky et at. [72] was due to the supervised training of a large network with 8 layers and millions of parameters on the ImageNet dataset with 1 million training images. The typical use of convolutional networks is on classification tasks, where the output to an image is a single class label. However, in many visual tasks, especially in biomedical image processing, the desired output should include localization, i.e., a class label is supposed to be assigned to each pixel. Moreover, thousands of training images are usually beyond reach in biomedical tasks.

The segmentation network for medical image proposed with FCN structure and various skill for location determination was studied. The typical resulting network has two drawbacks. First, when dealing with the big patches, the runtime efficiency degrades. Because the network will run for each patch separately, and due to overlapping patches, there will be a lot of redundancy. Secondly, there is a trade-off between the accuracy of pixel-wise localization of the object and the efficiency in using the context of the input image. Larger patches require more max-pooling layers that would result in a reduction of the localization accuracy while using small patches limits the network to see the bigger context. For improving the performance and bring out the potential of CNN, many complex architectures had been proposed since.

3.4.2 U-net

The most well-known, in medical image analysis, of the novel CNN architectures is U-net [73]. The two main architectural novelties in U-net are the combination of an equal amount of up-sampling and down-sampling layers. Although learned up-sampling layers have been proposed

before, U-net combines them with so-called skip connections between opposing convolution and deconvolution layers. This which concatenate features from the contracting and expanding paths. With modification and extension to this architecture such that it works with very few training images and yields more precise segmentations; The main idea is to supplement a usual contracting network by successive layers, where pooling operators are replaced by up-sampling operators. Hence, these layers will not decrease the resolution of the output. In order to localize the loss of information by Maxpooling layers, high-resolution features from the contracting path are combined with the successive layers. This process is done with the concatenation layer, which combines two layers with the same dimension by randomly choose to sample. The importance of implementing such concatenation layer has been discussed in [74]. Figure 3-7 shows us the complete layout of the U-net structure; each blue box corresponds to a multi-channel feature map resulted in a convolution operation. The number of channels is related to the filter size of the convolution kernel and is denoted on top of the box. The x-y-size is the size of denoting the height and width and is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. [75]

The whole network consists of a contracting path (left side) and an expansive path (right side). It's composed with the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each down sampling step, they double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The necessity of the cropping is due to the loss of border pixels in every convolution. At the final layer, a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

From the perspective of the training process, this means that the whole images that are scanned can be processed by U-net in one forward pass, resulting in a batch segmentation map. This would allow U-net to take into account the more content of the image, which can be an advantage in contrast to patch-based CNN. Furthermore, in an extended paper by Çiçek et al. [76], it is shown that a full 3D segmentation could be implemented by feeding U-net with a few 2D annotated slices

of images from the same volume. Other authors have also built derivatives of the U-net architecture. It is worth mentioning that although U-net is one of the most cited paper with satisfying accuracy, the heavy structure limits its use in many cases, especially when real-time features are put into consideration. In this research, we find that with deleting some of the layers, the runtime speed can be improved while the accuracy is preserved to be at an acceptable level.

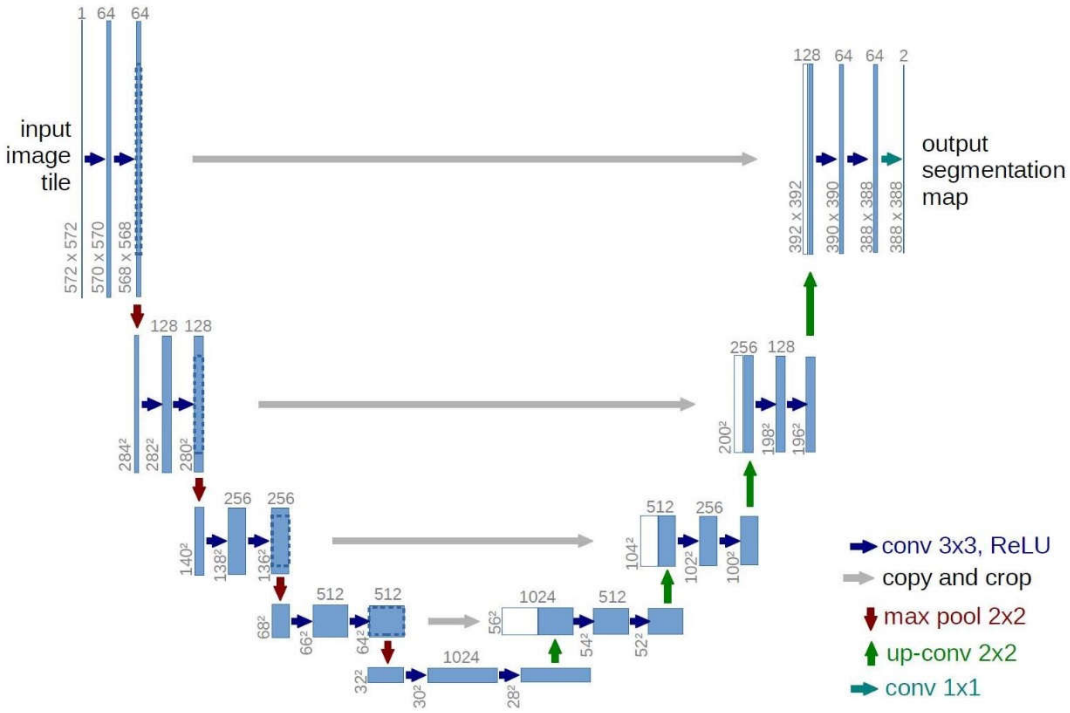

Figure 3-7 U-net architecture (an example of 32x32 pixels in the lowest resolution).

3.4.3 V-net

Milletari et al. [77] proposed a 3D-variant of U-net architecture, called V-net, performing 3D image segmentation using 3D convolutional layers with an objective function directly based on the Dice coefficient. Figure 3-8 showed the V-net architecture build and details of the parameters. In an implementation, the convolutions are performed to aim both at extracting features from the data and so at the end of each stage (similar to the concatenation in the U-net structure), the appropriate stride is used for decreasing the resolution. Compression path is included in the left part of the network, while the right part decompresses the signal until reaching the original size. Appropriate padding is applied to all convolutions also. The left part of the network is separated into several stages in which different resolutions are put into operation. Each stage comprises one

to three convolutional layers [77]. The most original feature for this paper is that the formulate for each stage is designed that it learns a residual function: the input of each stage is (a) used in the following convolutional layers and processed through the non-linearities activations and (b) added to the output of the last convolutional layer of that stage in order to enable the learning of a residual function. The observations from the author had confirmed that this architecture ensures convergence in the given ranges the time required by a similar network that could not learn residual functions.

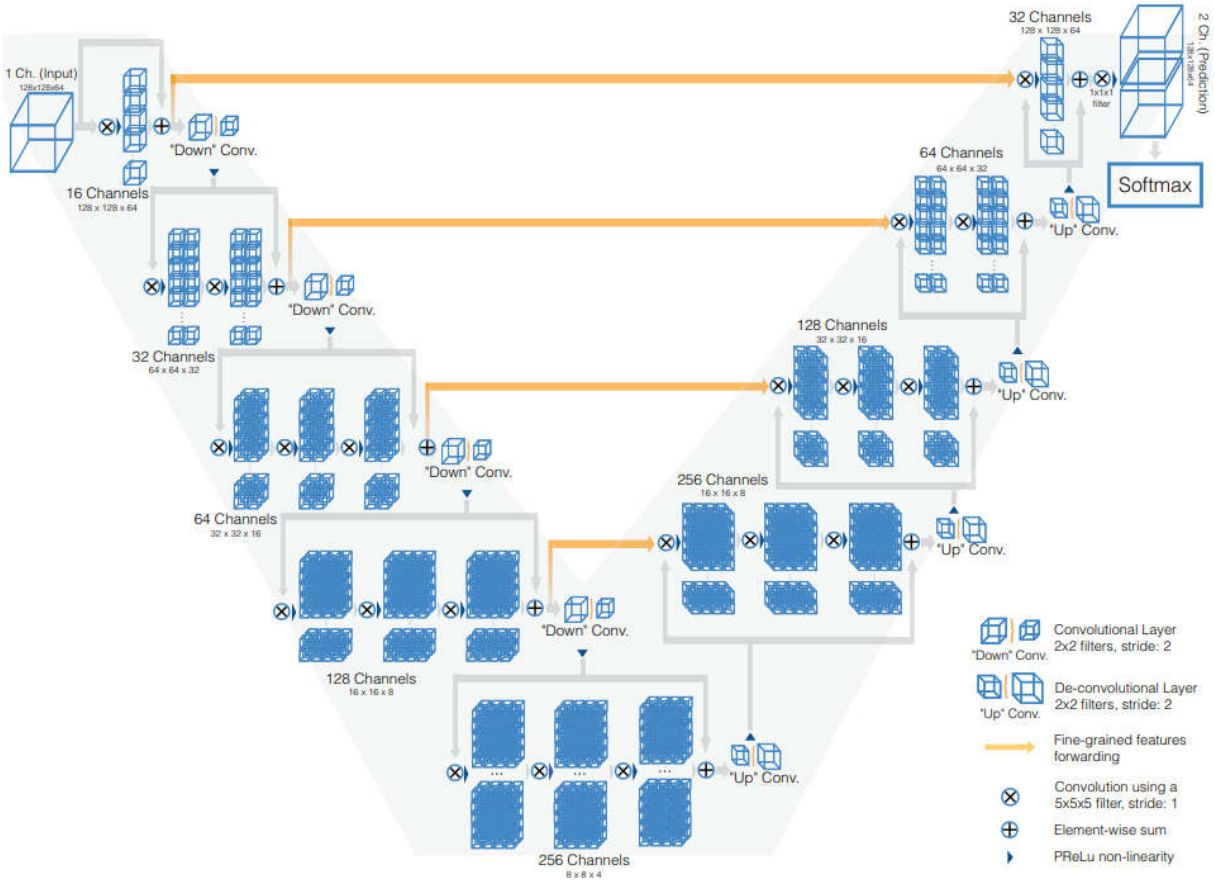

Figure 3-8 Scheme of V-net architecture [77]

The V-net structure reported a satisfying segmentation result for volumetric MRI images, the dice coefficient of 0.86 is reached while a Hausdorff distance of 3 5.71mm is reached. In the original paper, a runtime efficiency of 1sec for the given dataset is reached, but as the platform consists of a Tesla K40 GPU, such performance may not be general for most machines, judging by its network volume, it's safe to assume that it would cost at least a similar time with respect to U-net. Still, it's

a very competitive model for ultrasound segmentation. Especially with Doppler mode that with new information, the ultrasound images would become 3D.

3.4.4 Deep Belief Network (DBN)

If a database of segmented US images is available, as is assumed by the shape-constrained snake methods described above, then this database can also be used to learn the relationship between image features and the sought tongue contours using machine learning. Fasel and Berry [21] investigated this approach using deep neural networks, leading to the publicly available Autotrace software. In Autotrace, one deep neural network is trained on individual segmented US images to build an abstract and compact generative model of the relationship between image features and the location of contour vertices. Another deep neural network establishes the relationship between this abstract model and the unlabeled image data so that labels (i.e., segmentation masks of the tongue) can be inferred based on image data only. The method was extended to optimize the selection of training data, thereby improving training efficiency and segmentation accuracy [78]. Fabre et al. [79] proposed a similar approach which is based on a neural network with a simpler architecture, in that research, they establish a relationship between the principal component representation of images and that of the sought tongue contour. These methods based entirely on machine learning and require no manual initialization. They also perform segmentation independently on each image in a recording, thereby they can stop the segmentation errors to propagate from one frame to the next, but it would also neglecting the temporal consistency of tongue shape which is a useful source of regularizing information.

The model of Deep Neural Networks proposed in [80] is based on the stacking of Restricted Boltzmann Machines (RBMs). The idea of the Restricted Boltzmann Machine is somehow similar to a neural network that consists of a layer with visible units and a layer with hidden units, connected through directional links that are controlled by weights, which are symmetric for RBM. The possibility of activation of a hidden unit depends on the weight of that specific hidden unit in the visible layer (and vice-versa, since the connections are symmetric).

An autoencoder structure is a system which takes an input vector x and transforms it to a hidden representation h using an encoder network; this hidden representation is information rich but difficult to decode directly, so a decoder network will be followed to reconstruct the input from

the hidden representation acquired previously. A widely-used way of building an autoencoder is through restricted Boltzmann machines (RBMs). As a more detailed description, an RBM is a specific type of Markov random field in which the V dimensional input vector x and H dimensional hidden feature vector h are modeled by-products of conditional Bernoulli distributions:

$$f_{\phi,j}(x) = p\left(h_j = 1 \middle| x; \phi\right) = \sigma\left(\sum_{i=1}^{V} W_{i,j} x_i + a_j\right) \tag{3.1}$$

$$f'_{\phi,i}(x) = p(x_i = 1 | h; \phi) = \sigma\left(\sum_{j=1}^{H} W_{i,j} h_i + b_j\right) \tag{3.2}$$

where $\sigma(u) = (1 + e^{-u})^{-1}$, and $\phi = (W, a, b)$. Thus, when used as an autoencoder, f_ϕ is the encoder network and $f'_{\phi,i}$ is the decoder network, using shared parameters ϕ. The marginal distribution over visible vector x is:

$$p(x; \phi) = \frac{\sum_h e^{-E(x,h;\phi)}}{\sum_u \sum_h e^{-E(u,h;\phi)}} \tag{3.3}$$

where E is the energy of the joint configuration (x; h):

$$E(x, h; \phi) = \sum_{i=1}^{V} W_{i,j} x_i + \sum_{i=1}^{V} \sum_{j=1}^{H} W_{i,j} x_i h_i + \sum_{i=1}^{V} b_i x_i + \sum_{j=1}^{H} a_j h_j \tag{3.4}$$

Maximum likelihood parameters ϕ can be estimated by taking gradient steps determined by the contrastive divergence (CD) rule [80], which gives parameter updates:

$$p(x; \phi) = \langle x_i h_j \rangle_{data} - \langle \hat{x}_i \hat{h}_j \rangle \tag{3.5}$$

where expectation $\langle x_i h_j \rangle_{data}$ is the frequency that x_i and h_j are on together in the training set, and $\hat{x}_i \hat{h}_j$ is the frequency that $\hat{x}_i$ and $\hat{h}_j$, which represent samples of x_i and y_j obtained by running the Gibbs sampler initialized at the data for one step, are on together. Thus, one gradient update involves first sampling h from $f_\phi(x)$, then generating the "reconstructions" $\hat{x}_i$ by sampling from $f'_\phi(\hat{x})$, and finally encoding new features $\hat{h}$ by sampling from $f_\phi(\hat{x})$, giving all the terms needed to compute (3.5).

When inputs are real-valued instead of binary, Gaussian units can be used instead. Provided each input dimension xi is normalized to zero mean and unit variance, the only change needed to the above equations is:

$$f'_{\phi,i}(h) = p(x_i = x|h;\phi) = \frac{1}{\sqrt{2\pi}} e^{\left(\frac{1}{2}\left(x - b_i - \sum_{j=1}^{H} W_{i,j} h_j\right)\right)^2} \tag{3.6}$$

Recently it has been shown [80] that higher-order correlations in input can be captured by training a sequence of N RBMs, where the hidden layer of each lower level RBM is treated as the input to the next higher-level RBM. At runtime data is then run through each encoder function f_{ϕ_i} in sequence, and then is decoded by running through the decoder functions f'_{ϕ_i} in reverse order, in what is called a deep autoencoder.

3.5 Summary

The representatives of all the methods related in tongue tracking are the Edgetrak which uses active contour model [45][51], the Autotrace which adopted the Deep belief network as core algorithm [21], and the TongueTrack which is a model based segmentation algorithm [18]. An analysis of the tongue tracking errors was carried out, a comparison of performance achieved by Edgetrak, Tongue-Track, and Autotrace in their default configurations was made [82]. In the research mentioned above, a thorough and detailed comparison was made. The results, whether it is obtained automatically with model-based segmentation or is from the active contour model which requires manual initialization of the contour from the first frame, are put into comparison regardless. Such comparison is made beginning with the first frame in the sequence and is implemented across the entire video. Although the study may somehow be limited by comparatively small dataset size and the lack of diversity of the validation dataset, in the end, it led to some interesting discoveries. One was that for small training data sets, Auto-trace's accuracy was highly dependent on the training and validation data being similar, which denotes a low generality (the capability to generalize). Also, for all automated techniques, the evolution of segmentation error over time followed a similar pattern to that produced by the baseline, suggesting that none of the automated approaches entirely adapted to rapid tongue movement. This effect was particularly marked for TongueTrack, which tended to recover from errors mostly when

the tongue resumed its initial, resting configuration. In another related work [12], it was shown that the robustness of Edgetrak and TongueTrack is improved by periodically reinitializing the tracker. They proposed a method to detect images where the tongue is in its resting state based on similarity to the first frame in the sequence, and then used the initial tongue contour to re-initialize tracking in these frames [2], thereby promoting recovery from previous errors. The starting point for this research was the snake method proposed by [60] (which is also the annotation tool used in this research). Snakes offered a highly flexible representation that captures the subtle tongue shape variations that occur during the speech. However, snake-based tracking as implemented in Edgetrak has three limitations:

(1) it has a limited capture range

(2) it can produce contours that are not typical of speech-induced tongue shape

(3) it does not recover easily from errors.

These errors are due to the characteristics of the active contour algorithms, as well as the nature of the ultrasound tongue image. To address limitations 1 and 3, some work [18] [15] proposed the solutions to global optimization problems in Markov models, or use a particle filter in high order MRF, in which multiple tongue configuration hypotheses are maintained and evaluated over time. To address limitation 2 and simultaneously reduce the dimensionality of the search space for particle filtering, the method [18] enforces global shape constraints based on an ASM. Although many efforts are made for optimizing active contour method, the traditional algorithms are still very prone to random noise and are hard to recover from the error.

Unlike the traditional methods, the learning-based method requires a training set of segmented images, sometimes the database is very expensive (in terms of time) to obtain. However, these fully learning-based methods like Autotrace [21] can cope with the high-deformability and noise in the image much better than the traditional algorithms. Table 2-1 compares the technical requirements and other characteristics of the state-of-the-art tongue tracking methods for ultrasound image and videos. Like the traditional unsupervised methods [60] [18] [83], the automatic machine learning algorithms like [84] perform far better. A detailed performance comparison of the various state of the art algorithms will be given in later section 4.6. And at the time when this research is done, there are no Unsupervised Learning method published in this area

Method	User Input	Training	Main Parameters
Active Contour [16]	Manual initialization	None	Snake energy weights, dynamic programming region size
Edgetrak [60]	Manual initialization	None	Snake energy weights, dynamic programming region size, band energy penalty
Active appearance [65]	Manual initialization	PCA	Number of texture pc, shape pc
Quality Map [85]	Manual initialization	PCA	Similar to Edgetrack, number of shape pc
TongueTrack [18]	Manual initialization	None	Curvature weight, spatial smoothness weight, temporal smoothness weight, Gaussian kernel parameter
Bio-model [83]	Full tongue delineation	None	Minimum distance, optical flow parameters, material constant
Particle filter [15]	Manual initialization	None	Edge track parameters, maximum number of particles, likelihood factor
Autotrace [21] [78]	Fully-automatic	DBN	Deep Belief Network hyperparameters
ANN [79]	Fully-automatic	PCA + perceptron	ANN hyperparameters + number of shape pc
RBN [61]	Fully-automatic	DNN	DNN hyperparameters

Table 3-1 The summary of representative algorithms for tongue contour tracking or segmentation

4 Deep Learning based Real-time Tongue Contour Extraction System

4.1 Overview of the Proposed System

The architecture of the proposed tongue contour extraction system consists of several parts. There are two key blocks constituting the whole system, namely the tongue segmentation system, and the tongue contour extraction. Figure 4-1 below shows an example of input/output of the system. The left fig illustrates the ultrasound probe under the chin to capture the tongue motion. The second image contains the cropped area of the original video from which the contour will be extracted. The third image is the predicted segmentation from the upper right. And the last image is the contour skeleton of the white region of predicted segmentation . The details of the process will be covered in the following discussion.

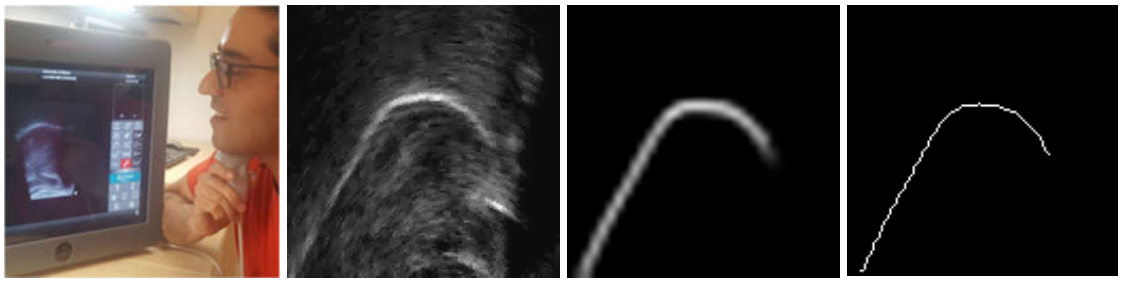

Figure 4-1 The example input/output of the proposed method： from left to right are the scene of data collection, raw UST data (input), the predicted mask from the model and extracted contour from prediction (final output).

In this ultrasound tongue segmentation and contour extraction system, the tongue segmentation part is the first and the most important step for the initialization of the whole system by finding the tongue location in a pixel-wise manner. For each frame containing the image of the ultrasound tongue, a trained network would be applied for extraction of the tongue location in real time. Then in order to extract the contour from the segmented tongue area, the morphological operations will be implemented. And the skeleton of the segmented tongue region will be separated and treated as the tongue contour, and the contour information will be used as the validation data to be compared with the related works in the field for the accuracy and performance. Figure 4-2 shows the pipeline of the whole system, including the data collection, a database built, architecture setup and training

process. The postprocessing and evaluation stages are also included. The detailed description of the workflow would be introduced thoroughly in this chapter.

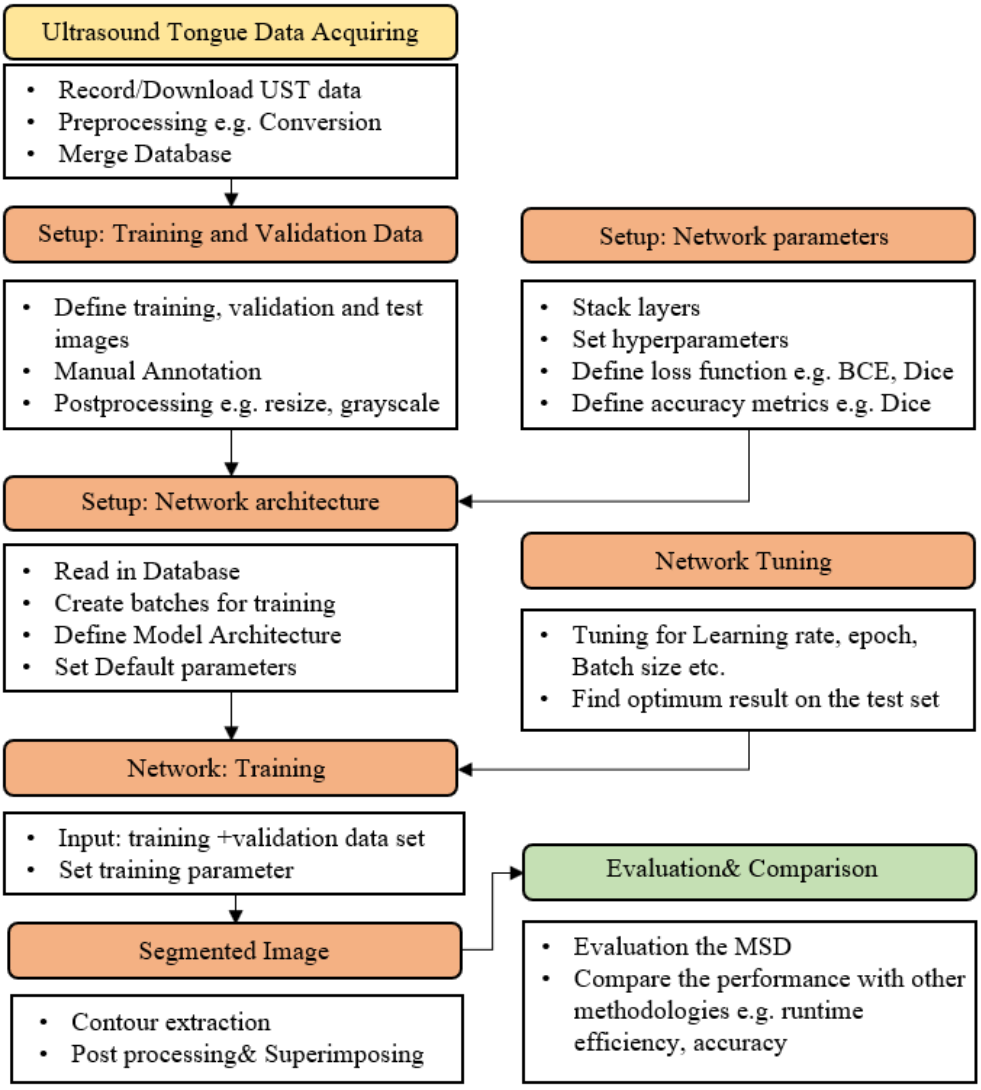

Figure 4-2 Overview of the system framework to perform tongue segmentation

In order to give an elaborated explanation of the whole system, first the emphasis of the chapter 3 would be on discussion of the use the deep learning technique, the details of the operation of convolutional neural network (CNN) and how the stacked fully convolutional network can learn to recognize the image and segment the desired object out. The second part would be of the image processing and morphological operations which are used to give the contour information, such information not only helps the illustration and comparison but also enables the possible follow up research on utilizing this information for e.g., 3D reconstruction.

4.2 Convolutional Neural Network

Neural networks have been studied since the middle of the last century and have experienced a surge in development at the end of the last century [86]. Neural networks, especially CNN structures are a widely used method to implement deep learning.

A neural network is a hierarchical model inspired by the structure of the brain. A network consists of a series of nodes connected according to certain rules [86]. A naive feed-forward neural network can consist of only three layers; these are the input layer, the hidden layer, and the output layer, it is to be noted that, the external input and the output are not directly related. The nodes between two adjacent layers are connected by directed edges. Each of these edges corresponds to a weighted value. The two most obvious characteristics of neural networks are the introduction of nonlinear activation functions and the use of a nested design between layers. This allows the network to be able to approximate a highly non-linear function. Thus, neural networks have a strong modeling capability for complex data patterns [86]. Neural networks have developed into many types. A commonly used neural network is the CNN. The most significant feature of a CNN is the introduction of the convolutional operation and weight sharing. Given the characteristics, a CNN has a good performance on problems where the input is not numbers but images. The convolutional operation replaces the full connection of the feed-forward network [6] with local connections, and the weight of the operations between the different layers is shared [6]. When applying a convolution kernel onto an image, the convolution kernel looks as though it is an observation window at the detection time which gradually moves from the upper left corner of the image to the bottom right corner. Each position corresponds to an output node (each pixel point on the image corresponds to an input node). Specifically, the weights of different output nodes corresponding to the same input nodes are the same [44]. That is weight sharing.

4.2.1 Input layer

The first layer of the CNN is usually referred to as the input layer, which is also called the image processing layer. This layer is responsible for optional pre-processing and filtering. In face detection, the data layer will process the raw pixel values of the input images.

4.2.2 Convolution layer

The convolution layer is the sliding window calculation of the input layer by its convolution kernel. Each parameter of the convolution kernel is equivalent to the weight parameters of a traditional neural network and connects to the corresponding local pixels. The sum of multiplying the parameters of the convolution kernel and the corresponding local pixel values (usually plus a bias parameter) is the result of the convolution layer. Figure 4-3 reveals this layer's working mechanism of a 3 x 3 kernel applied to a 4 x 4 image to gain a 2 x 2 output result. Each step calculation formula is shown in Formula 3.2.1, 3.2.2, 3.2.3 and 3.2.4.

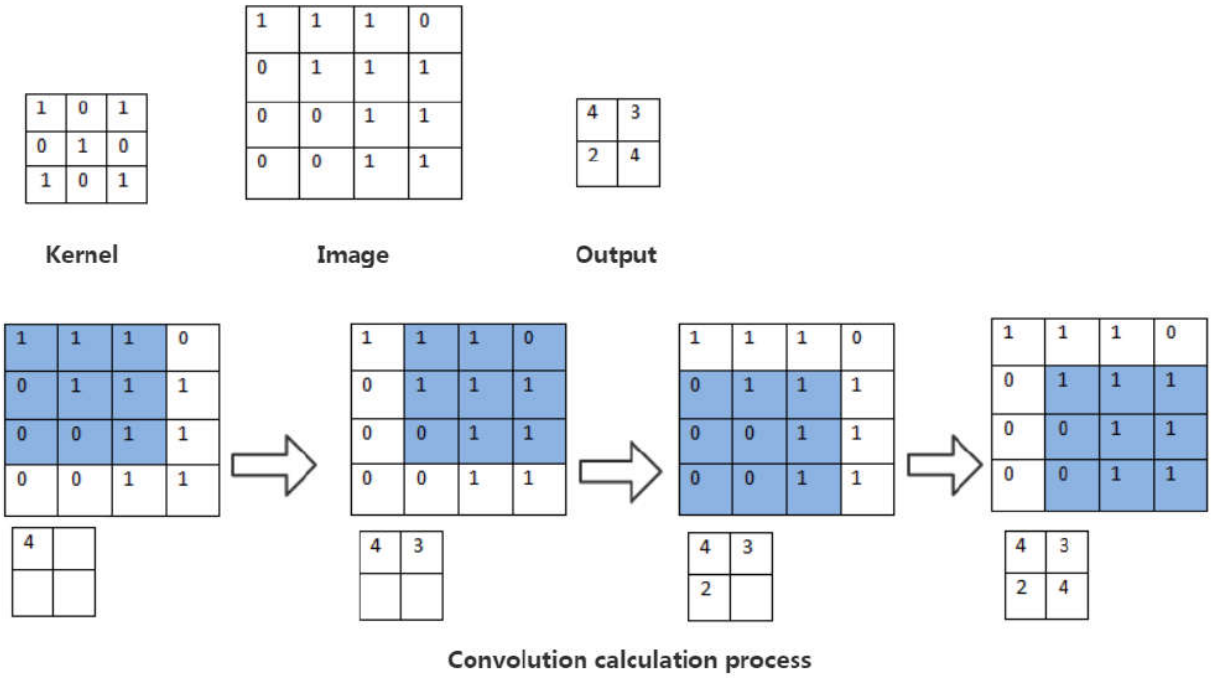

Figure 4-3 The process of the computation of the convolution [72]

$$\text{Output}(1,1) = 1*1 + 0*1 + 1*1 + 0*0 + 1*1 + 0*1 + 1*0 + 0*0 + 1*1 = 4 \quad (4.2.1)$$
$$\text{Output}(1,2) = 1*1 + 0*1 + 1*0 + 0*1 + 1*1 + 0*1 + 1*0 + 0*1 + 1*1 = 3 \quad (4.2.2)$$
$$\text{Output}(2,1) = 1*0 + 0*1 + 1*1 + 0*0 + 1*0 + 0*1 + 1*0 + 0*0 + 1*1 = 2 \quad (4.2.3)$$
$$\text{Output}(1,1) = 1*1 + 0*1 + 1*1 + 0*0 + 1*1 + 0*1 + 1*0 + 0*1 + 1*1 = 4 \quad (4.2.4)$$

4.2.3 Pooling layer

After obtaining the features of the image through the convolution layer, in theory, we can directly use these features to train the classifier. Though it is a big computational challenge, and prone to the so-called over-fitting phenomenon. Over-fitting typically happens when the error on

the training set is driven to a small value, but when new data is presented to the network, the performance difference is huge between old and new data. This is because the network has memorized the training examples; however, it has not yet learned to generalize to new situations. In order to further reduce the network training parameters and the degree of the over-fitting of the model, we will be sampling (Pooling) convolution layer outputs. Pooling/sampling is usually done in the following two ways:

- ***Max-Pooling:***

  Select the maximum value in the pooling window as the sample value;

- ***Mean-Pooling:***

  Sum all the values in the pooling window and take the average as the sample value.

Max-Pooling is helpful to reduce the estimated value offsets resulting from convolutional layer difference error, like in Figure 4-4, the Max pooling process with a (2x2) kernel and stride of s=2 is shown. Maxpooling layers reduce the spatial dimension of the input [87] and keep more image texture information compared with Mean-Pooling, so we use Max-Pooling in our system.

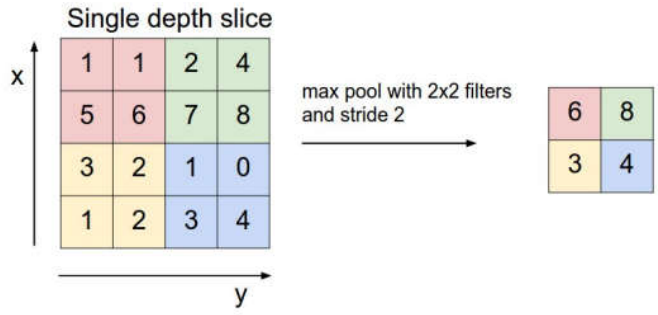

Figure 4-4 Schematic of Maxpooling process

4.2.4 Activation layer

The commonly used activation functions are sigmoid, hyperbolic tangent and Rectified Linear Units (ReLU). In our work, we deploy ReLU for all the activation layers, because the lower computational cost of activation function in the backpropagation gradient derivation is lower with ReLU than with the sigmoid function [88]. The ReLU [89] layer involves the ReLU function in the CNN implementation. This is used to add non-linear factors because the linear model's expression is not sufficient to calculate complicated features. The ReLU function is shown in Formula 3.2.7.

$$f(\mathrm{x}) = \tanh(\mathrm{x}) \tag{4.2.5}$$

$$f(\mathrm{x}) = \frac{1}{1 + e^{-x}} \tag{4.2.6}$$

$$f(\mathrm{x}) = \max(0, \mathrm{x}) \tag{4.2.7}$$

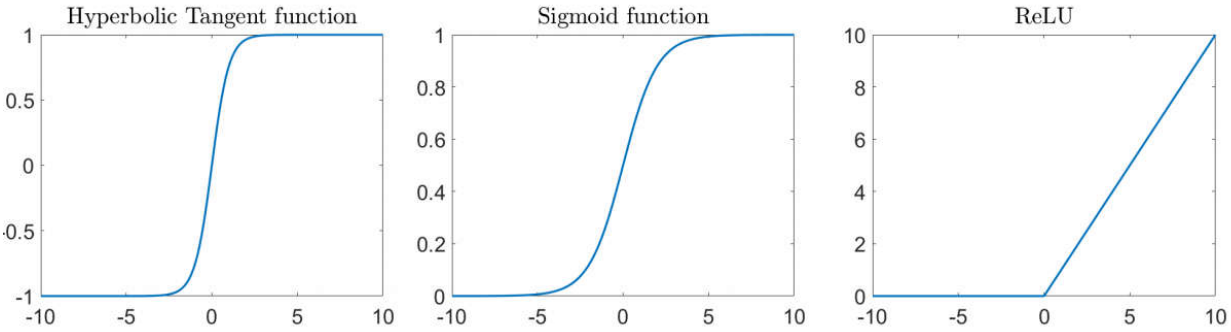

Figure 4-5 The illustration of some widely used activation function

4.2.5 Fully connected layer

The fully connected layer is also called the inner product layer. That is equivalent to a classifier in the network, which calculates the class scores corresponding to output categories.

4.2.6 Dropout layer

Dropout layer was proposed by G. E. Hinton et al. [90]. When training a neural network model, if the training sample is limited, a Dropout layer can be added in order to prevent the model from over-fitting. During training, the dropout layer can force some of the feature detectors to form a random connection instead of depending on one specific neuron too much; this technique can improve the generalization of the network capacity.

Overfitting is a common problem of machine learning that can be mitigated with neural network training scheme. When a model is overfitted to the training data, it loses its capability of generalization. The model has learned the training data, including noise, to such a great extent that it has failed to capture the underlying relationships of general information but only memorizes all the possibilities naively. A large number of weights are to be trained for most CNN structured models. Therefore overfitting can occur due to training too few training examples. Dropout layers are the tool to prevent overfitting problem (Figure 4-6). In dropout, nodes and its connections are

randomly dropped from the network. Dropout constrains the network from memorizing the training set. Consequently, it prevents that the weights from being tuned to memorize all the cases for the data. If not, the difference in performance between training data and validation data will decrease. Dropout layers are used during training only, not during validation or testing.

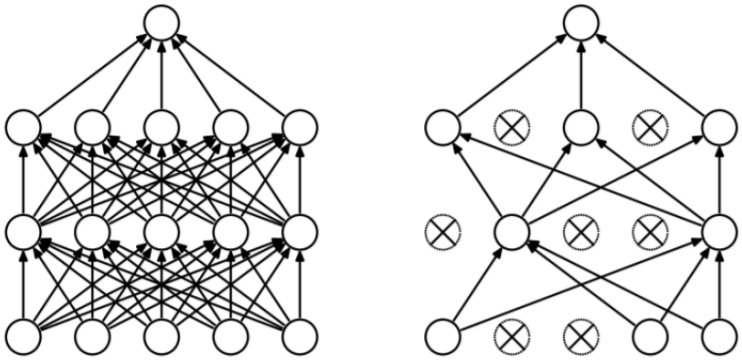

Figure 4-6 A neural network structure before and after applying dropout [90]

4.3 Backpropagation

The information above described the information about building a model. After the model is built, it can give the desired output when given proper input. But in order to find the optimal model parameters of weights and bias, a rule of computing and updating the parameters with respect to the loss should be specified. Backpropagation is one such method that can efficiently compute and adjust the parameters using gradient-based optimization techniques [91]. Backpropagation can find the specific combination of weights and bias which minimize the loss function (or error function). The core concept of this method is to compute the gradient of the loss function with multiple iterations, and after each iteration, the parameter will be updated by a certain amount. Therefore the loss function should meet both conditions at every iteration step, that is to continue and differentiable. And finally, the parameters tend to converge at an optimum value when the setting of backpropagation is tuned properly.

The initial weights of an untrained CNN are randomly chosen. Consequently, before training, the neural network cannot make meaningful predictions for network input, as there is no relation between an image and its labeled output yet. By training the network with a certain training set, test images and their labeled outputs with desired classes, the weights are adjusted accordingly. Training is the process of adaptation of the weights in such way that the difference between desired

output and network output is minimized, which means that the network is trained to find the right features required for classification. There are two computational phases in a neural network, the forward pass and the backward pass in which the weights are adapted.

4.3.1 Forward Pass

An image is fed into a network. The first network layer outputs an activation map. Then, this activation map will be the input to the first hidden layer, which computes the second activation map. The second hidden layer uses the values of the second activation map as inputs, and again another activation map is computed. Continuing this process for every layer until the output layer will eventually yield the network output. And this output is used as the result or for adjusting the network.

4.3.2 Backward Pass

In this phase, the weights are updated by backpropagation. One epoch of backpropagation consists of multiple parts, usually, multiple epochs are required for convergence, and the backpropagation usually consists of several parts as listed below:

- ***Loss Function***

In the forward pass, the inputs and desired outputs are presented. A predefined loss function is used to compute the difference between the input and the desired output. The goal is to give a value that can represent how the network output differs from the ground truth, and it can be used for adjusting the weights so that the performance increases; this is achieved by calculating the derivative with respect to the weights of the loss function. For the detailed property of loss function, please refer to the following section on "*Loss function.*"

- ***Back Propagation***

The backpropagation is the process for computing the loss of network; the computation is made computing the layer-wise gradient with respect to the weights of the loss. During the backward pass, the weights that contributed the most to the loss are determined and are to be adjusted so that the total loss decreases within one backpropagation.

- ***Weight Update***

In the final part, all weights are updated in the negative direction of the loss function gradient (within backpropagation).

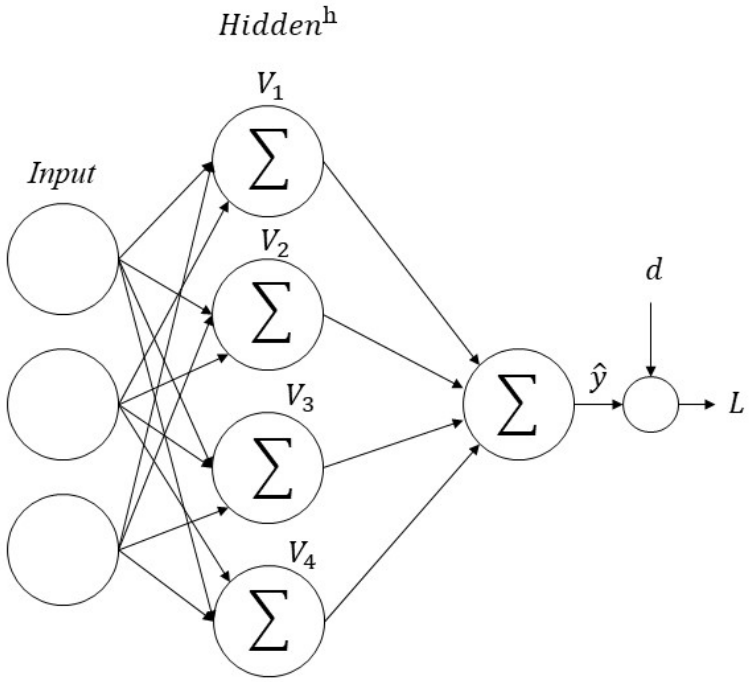

Figure 4-7 An example of multilayer perceptron for interpretation of backpropagation

The core concept of the backpropagation based weight update is to compute the gradient of the loss function with respect to the network weights. Computing the partial derivative ∂L/∂w is essential to minimize the loss function value. Stochastic Gradient Descent (SGD) is the most popular way to optimize neural networks because it cooperates the time efficiency and optimization stability well. Backpropagation example for a Multi-Layer Network A simple example of computation of weight updates in backpropagation will be given using the network shown in Figure 4-7. The cost function L is given below, e_l is the error between the desired output d_l and network output y_l . The network output y_l is computed in the forward pass and depends on outputs of the previous layer v_j and the output layer weights w_j^o.

$$L = \frac{1}{2}\sum_{l}(e_l)^2$$

$$y_l = \sum_j w_j^o v_j \tag{4.3.1}$$

$$e_l = d_l - y_l$$

The Jacobian and the Calculation of the partial derivatives yields the following Jacobian for the output layer is given by:

$$\frac{\partial L}{\partial w_{jl}^o} = \frac{\partial L}{\partial e_l}\frac{\partial e_l}{\partial y_l}\frac{\partial y_l}{\partial w_{jl}^o} = -v_j e_l \tag{4.3.2}$$

Using the SGD update rule, which will be explained in the following section, the output weights are updated using:

$$w_{jl}^o(n+1) = w_{jl}^o(n) + \alpha(n) v_j e_l \tag{4.3.3}$$

After the updating rule of the network are set, the weights in the hidden layers can be updated. As it is a backward pass, first gradients of the output layers are computed, then the gradients of the hidden layers. The Jacobian is given by:

$$\frac{\partial L}{\partial w_{ij}^h} = \frac{\partial L}{\partial v_j}\frac{\partial v_j}{\partial y_l}\frac{\partial z_j}{\partial w_{ij}^h} \tag{4.3.4}$$

Calculating the partial derivatives yields:

$$\frac{\partial L}{\partial w_{ij}^h} = -x_i \sigma_j'(z_j) \sum_j e_l w_j^o \tag{4.3.5}$$

which would yield the update rule for the hidden layers:

$$w_{ij}^h(n+1) = w_{ij}^h(n) + \alpha(n) x_i \sigma_j'(z_j) \sum_j e_l w_j^o \tag{4.3.6}$$

Finally, the network is put into the test with the validation dataset; this dataset contains images that are different from the training dataset. By increasing the amount of training data, the more training iterations are carried out, the better the weights are tuned.

4.3.3 Loss Function

The value of the loss function L represents the difference between the training image and the desired annotated output image after the input propagated through the network. Two assumptions are made about this loss of function. First, the loss function should be defined that is able to average over the individual training images, as the training often is carried out in batches. The loss function is computed and averaged at the end of each batch; then the weights are updated. Secondly, the loss function should be defined as a function that represents the network outputs. Below a brief overview is given of some widely used loss functions, where x_i are the neuron outputs and $\hat{x}_i$ are the desired outputs.

- ***Quadratic Cost Function***

  Mean Squared Error (MSE) cost function is one of the simplest cost functions:

$$\mathrm{L} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{4.3.7}$$

- *Cross-Entropy Cost Function*

  The cross-entropy cost function is commonly used in convolutional network applications:

$$\mathrm{L} = \frac{1}{N}\sum_{i=1}^{N}(\hat{x}_i \ln(x_i) + (1 - \hat{x}_i)\ln(1 - x_i)) \tag{4.3.8}$$

4.3.4 Stochastic Gradient Descent (and variants)

In both Gradient Descent (GD) and Stochastic Gradient Descent (SGD) [92] parameters are updated according to the update rule described above in an iterative manner until the optimization process is done. Computing the exact gradient using GD in large datasets is often too computationally expensive (GD is deterministic, which means it can guarantee the convergence if the proper data are present),

As the batch gradient descent method runs through all training samples to perform a single update for one iteration step, such operations often take up huge memory spaces and cost lots of time. In SGD, an approximation of the true gradient is computed, the computation is made by using only one subset of training samples for a parameter update. When the subset of training

samples contain more than more than one sample, this method is called mini-batch GD. SGD is similar to GD in computation and weight update: in order to decrease the loss function value, the learning rate η determines the step size to get to the local or global minimum. The update rule is given in equation 3-2-16.

$$\theta = \theta - \eta \nabla_\theta L(\theta) \tag{4.3.9}$$

- *Mini-batch Gradient Descent* [93]

Like introduced above, this method performs an update for every mini-batch of n training samples(n >1). Mini-batch GD reduces the variance of the parameter updates. Larger mini-batches reduce the variance of SGD updates by taking the average of the gradients in the mini batch. This allows taking bigger step sizes. In the limit, if each batch contains one training sample, it is the same as regular SGD.

4.4 Customized FCN for Image Segmentation

Detecting the location and size of a lesion in medical imaging is of as great an interest as classification. CNN is usually applied for data classification/recognition producing a single class label output for each image input. Image segmentation differs in that each pixel in an image receives a class label.

FCN [26] has demonstrated image semantic segmentation capabilities. For these networks, the amount of training data is of great importance [21][22], and hyper-parameter tuning is vital for higher accuracy. In order to increase segmentation accuracy for biological applications, FCN has been modified and extended to a new structure, similar to deconvolutional networks [95], and named U-net [27]. U-net extracts desired features from an image with the help of encoder-decoder style convolutional filters. In encoding layers, features are detected similarly to FCN networks while in decoding, instead of using one fully connected layer, parts of previous feature maps from encoder layers are concatenated with the result of the deconvolution, producing a new operator called “up-convolution.” This enhancement helps the network to localize features with more details.

For these networks, the amount of training data is of great importance [96], [75] and hyper-parameter tuning is vital for higher accuracy. In order to increase segmentation accuracy for

biological applications, FCN has been modified and extended to a new structure, similar to deconvolutional networks [95], and named U-net [75] as introduced, a detailed description of U-net had been given in section 3.4.1. U-net extracts desired features from an image with the help of encoder-decoder style convolutional filters. In encoding layers, features are detected similarly to FCN networks while in decoding, instead of using one fully connected layer, parts of previous feature maps from encoder layers are concatenated with the result of the deconvolution, producing a new operator called "up-convolution." This enhancement helps the network to localize features with more details.

The u-net architecture achieves very good performance on very different biomedical segmentation applications. Thanks to data augmentation, it only needs very few annotated images and has a very reasonable training time also.

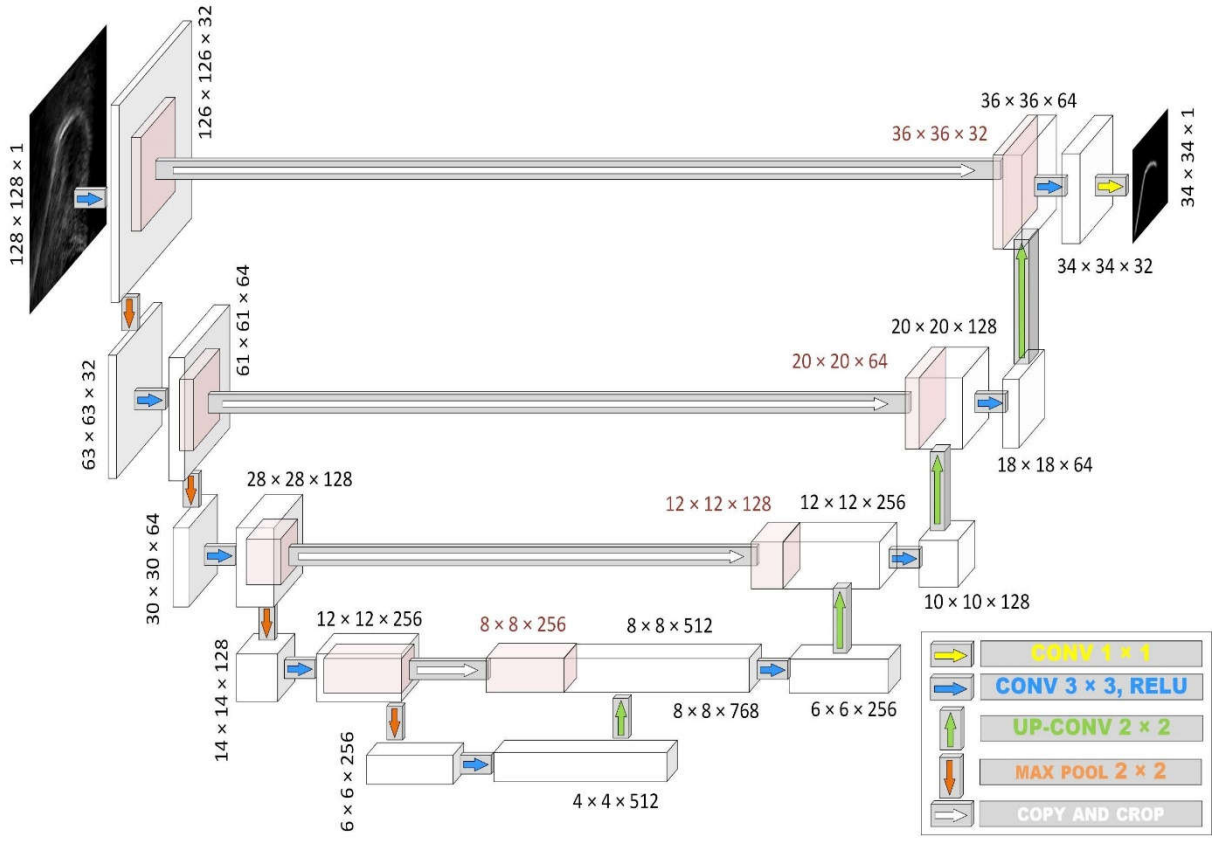

Figure 4-8 The proposed sU-net architecture.

We have adapted and modified the U-net architecture for real-time applications. Figure 4-8 illustrates our proposed architecture. It is comprised of shrinking expanding style layers. Each cube

corresponds to a multi-channel feature map. Arrows with different colors illustrate the different operations. The numbers indicate size and depth. In the top layers, the number of filters is small, and the size of feature maps is relatively large while in the lower layers, there are more filters with smaller size feature maps. Like the original U-net, our network consists of repeated (3 × 3) convolutions with no zero padding, each followed by a rectified linear unit (ReLU). A (2 × 2) max pooling operator with a stride of 2 is applied in each down-sampling phase. In the return path from the lowest contracted layer to the output feature map, each layer consists of an up-sampling of the feature map followed by a (2 × 2) convolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the corresponding contracting path, and a (3 × 3) convolution followed by a ReLU. Due to the loss of the pixels in the border for every convolution, the cropping is necessary for each concatenation step. At the final layer, a (1 × 1) convolution is used to find the desired segmentation image. In comparison with a U-net network with 23 convolution layers, sU-net has only 14 convolutional layers.

One important modification in our architecture compared to the original U-net is that we decreased the number of duplicated convolutional and deconvolutional layers as well as activation functions in each layer. The main motivation of this modification is that the only difference among frames resides in the tongue and all frames are relatively similar to each other. Thus, there is no need to have many convolutional layers to localize tongue position. As there are already hundreds of thousands of parameters in each layer, many layers would be redundant in our application.

In contrast to FCN, U-net does not have any fully connected layers, and the network employs only a valid part of each convolution. A method, called overlap-tile strategy, allows a seamless segmentation of images with arbitrary sizes [75]. We omitted this extrapolating process for the black border region of the tongue ultrasound images. Therefore, our simplified version of U-net (called sU-net), is significantly faster in training and validating and can automatically delineate/segment the tongue dorsum from ultrasound video in real-time.

4.5 Contour Extraction using Morphological Operations

After implementing the sU-net, a prediction mask denoting the tongue area would be given, although the accuracy maybe enough for visualization. Such a mask is hard to be compared with

other methodologies for evaluation. And under most circumstances a thin, clean line is more desirable, thus, extracting the line from raw prediction mask become an important task. In this thesis, the morphological skeleton is used for extraction of tongue contour.

An informal definition of a skeleton would be a line representation of an object that is:

1) one-pixel thick, 2) through the "middle" of the object, and, 3) preserve the topology of the object. The skeleton transformation is a widely used transformation in the field of image processing and mathematical morphology [97]. The skeleton *S (X)* of a set $X \in R^2$ can be described intuitively as the set of the median lines of x, i.e., the lines which are equally distant from two parts of the boundary of X. From *S (X)*, one can derive its end points, multiple points, length, loops, etc. all being important features for set description and characterization. With the minimum loss of topological information, the reason why skeletonization is successfully applied to various image processing problems is clear, such as optical character recognition, biomedical imaging, materials science, etc. [98]

The notion of the skeleton was introduced by Lantuéjoul [99], who derived the following morphological formula for the skeleton of a continuous binary image. But in practice, the digital images are not continuous, for processing discrete images. A new notion introduced by Serra [100].

$$\mathrm{S}(A) = \bigcup_{k=0}^{K} S_k(A) \tag{4.5.1}$$

In which

$$\mathrm{S}(A) = (A \ominus kB) - (A \ominus kB) \circ B \tag{4.5.2}$$

Where B is a structuring element, and $(A \ominus kB)$ indicates k successive erosions of A k times, and K is the last iterative step before A erodes to an empty set. The formulation given in 3.4.1 and 3.4.2 states that S (A) can be obtained as the union of the skeleton subsets $S_k(A)$. Also, it can be shown that A can be reconstructed from these subsets.

By implementing the skeleton algorithms, the single-pixel-wide contour could be extracted from the raw prediction mask, and the topological information would be preserved. Figure 4-9 shows an example of the image before and after implementing the skeleton algorithm.

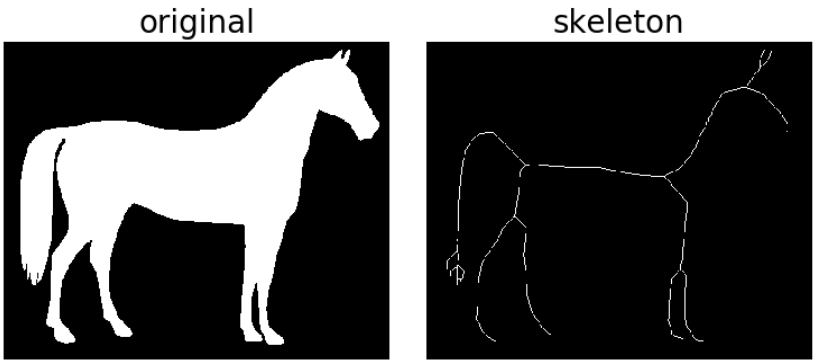

Figure 4-9 Example of the morphological skeleton, the extracted line is one-pixel wide

4.6 Tensorflow

The implementation and experiments made in this work are based on Tensorflow-python and OpenCV-python, for OpenCV had been popular opensource software overtime while the emerge and programming style of Tensorflow is relatively new, it is worth introducing some important concept and feature in Tensorflow in order to gain a better understanding of the work. Tensorflow is currently the top learning framework published by Google, allowing a user to implement various algorithms and optimize within the structural details efficiently. Given the wide range of functions already made available, as well as the community support and various open source libraries, TensorFlow was gain popularity rapidly among researchers. TensorFlow is developed for efficient parallel computations. This makes the framework popular among many researchers. It also cooperates with multi-GPU training well , which speed up the training process by much.

In this section, the core concept: a computational graph will be introduced, and the general workflow of executing Tensorflow would also be given.

4.6.1 Computational Graph

The meta-graph is one way of visualizing mathematical operations in sensor flow, with every computation operation forming a node in the graph, the whole process can be visualized as a data-flow graph. A core concept in functional programming, the backpropagation algorithm can be intuitively understood with this representation. For a deep neural network, computational graphs can get notoriously complex, with a few million parameters at least. Figure 4-10 showed a simplified version of the computation graph, in which is computation operation is defined as the

core part of the graph, and the variables are pre-defined as empty placeholders. The parameters like weights and biases would be initialized using specialized techniques while the input data like training/validating set would be fed into the model after all parts of the model are initialized.

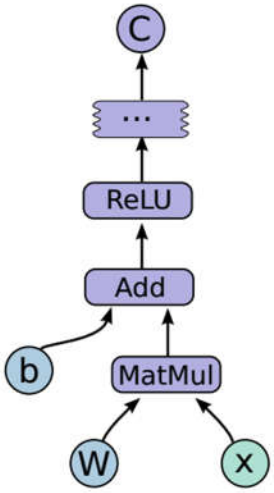

Figure 4-10 An example of Computation Graph

In Tensorflow and many similar machine learning libraries, the computation graph defines the computational relationship between variables, only after the parameters and variables are initialized and the operation is executed, which differs from the traditional function. And tensor is the most important concept in the computation graph.

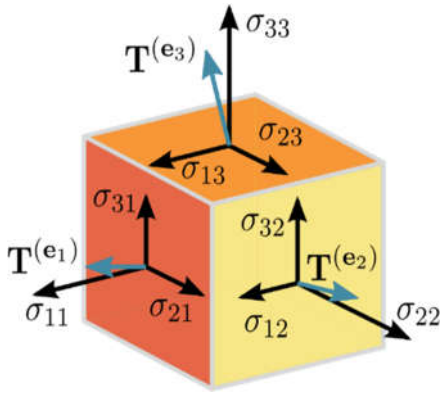

Figure 4-11 Schematic of a Tensor

When building the network, the weights, bias, input/output data are not assigned at first; only a tensor is created. The definition of the tensors is that they are geometric objects, which describe linear relations between the vectors, scalars, and other tensors. The examples of tensors include

the dot product, the cross product, and linear maps, etc. Geometric vectors are often used in physics and engineering applications. In Tensorflow, the tensors are defined as the multi-dimensional arrays. Figure 4-11 showed the schematic of a tensor, a tensor is identical to a multi-dimension array in data shape, while each of its dimension would contain practical meaning such as a color image is stored as a 3-D array, and its dimension would represent the height, width and color channel of the image.

4.6.2 Execution of Tensorflow

The difference between TensorFlow and most other deep learning frameworks is that an "operation" for tensorflow only defines a node representing a particular operation in a meta-graph structure and the execution is not initiated immediately. Work-flow is hence divided into two separate phases: the graph construction and an explicit execution phase.

1. Construction: Here, one declares symbolic operations that represent equations and functions of the chosen architecture. This includes convolutions, loss and cross entropy calculation, dropout probabilities, pooling, their constituent operations and more.
2. Execution: This is the stage that data is actually fed into the graph, and the above-defined model is run in an executable environment, referred to as sessions. of TensorFlow [101], in Python. TensorFlow is a recently open-sourced deep TensorFlow ships with a convenient web-based visualization tool, Tensor-Board [102]. With tensorboard, the parameter and the graph connection, as well as the change of the variables, were visualized throughout the learning process, whenever necessary. This allowed for a much deeper understanding of how the subtle changes emerge during the training process. The data flow graph for the model used was also be visualized, as in Figure 4-10 and Figure 5-6.

Deep learning implemented with Tensorflow (or other similar architecture like [103] [104]) could bring many benefits, for example, when training a model, variables should be used to hold and update parameters. In deep learning, the number of variables could be at the level of millions or more. Defining variables with tensor give better clarity and management in building/executing the network. These variables are in memory buffers containing tensors and can later be restored to exercise or analyze the model.

5 Database Generation & System Implementation

5.1 Databases

The dataset used for the research contains two parts: one is from an online dataset, and another one is captured with the ultrasound machine in the lab, with the annotation added manually, the intention of getting two datasets is to augment the data for the enhancement of the performance, as well as doing the comparison of the generalization capabilities of the deep learning algorithms, to gain a better insight of the how well deep learning algorithms can do, when applied two dataset from different distribution, namely the dataset shift in machine learning.

5.1.1 Database A: Online dataset from Seeing Speech

"Seeing Speech" is a collaborated project at six Scottish Universities, which recorded Ultrasound tongue imaging (UTI) videos, MRI videos of all the vowels and consonants, even some other symbols like voiceless labial-velar fricative or approximant. A set of corresponding animations showing how the phonemes are articulated is also made based on the information from UTI and MRI. This is one of the complete databases regarding the topic that can be found online. It's worth mentioning that although in the official website the annotation with UTI along with possible MRI is available, the manual annotation made by Ph.D. Candidate Mr. Hamed in our lab, who used active contour method [62] in implementation. Figure 5-1 illustrates the database provided by Seeing speech.uk. The plot on the left shows a typical image with annotation, the plot on the right shows the ultrasound imaging system used for acquiring UTI. The ground truth mask was not provided though, and manual annotation was implemented.

In total 4017 raw video frames are cropped and annotated out of 45 videos from seeing the speech. These processed videos frames build up the first dataset for this research.

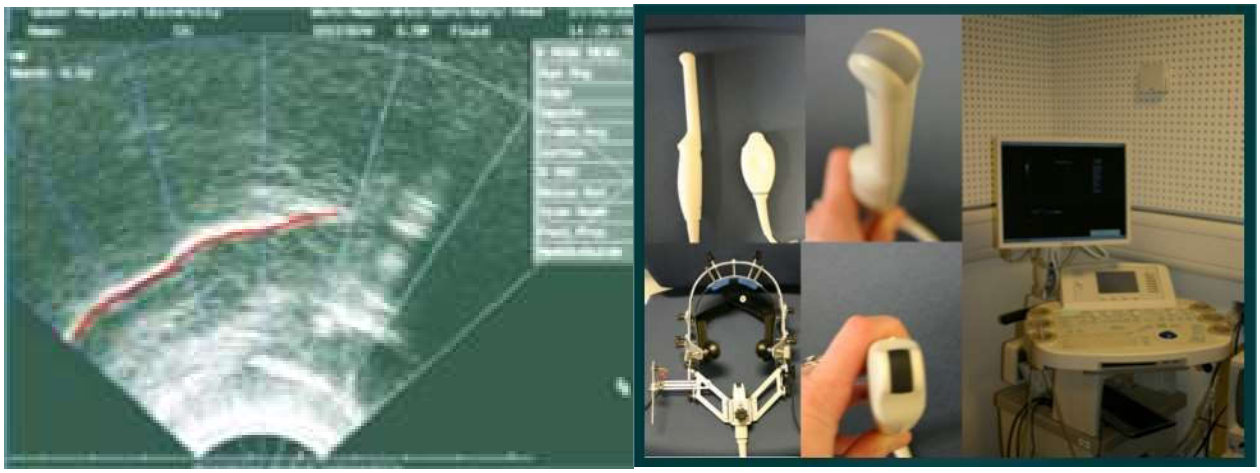

Figure 5-1 Database from Seeing-speech.uk

5.1.2 Database B: Dataset from In-house Ultrasound Machine

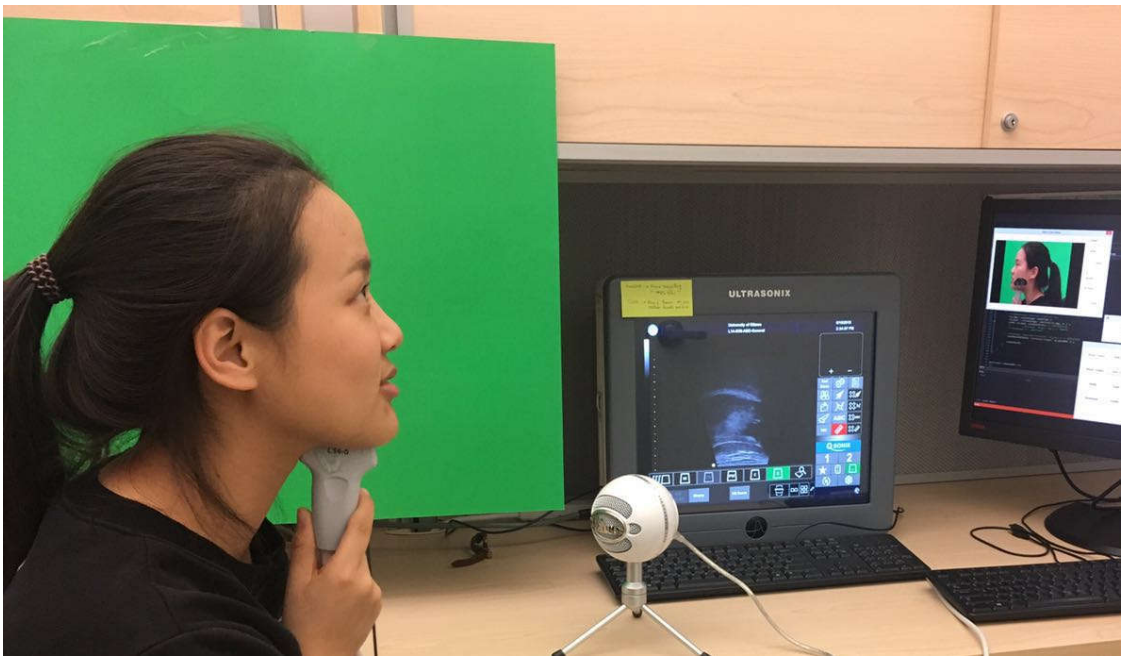

Figure 5-2 The experiment setting of acquiring a new dataset

The figure above contains the scene of the experimental setting when gathering ultrasound tongue imaging, BK-Ultrasound^{TM}: ULTRASONIX provides the ultrasound device for recording the signal, during the experiment the subjects were asked to pronounce some certain words, and the UT videos are recorded accordingly. For the specification of ultrasound frequencies, penetration depth, and other ultrasound imaging priors and tongue characteristics, please refer to

the Sections 2.2 and 2.3. For annotation of dataset, the same method mentioned above (active contour model) is adopted, for each frame, the annotator is required to put some dots close to the tongue contour, and the active contour model would fit a line to the tongue contour automatically, this process takes around 5-10s (including putting dots manually) for each frame. And if every frame is annotated without the tool, it would take roughly 20s, so the annotation tool would greatly improve the annotation efficiency. Another advantage is that for this tool is widely adopted; it would be easier for us to compare our result with other researchers, including the work done by active contour model itself.

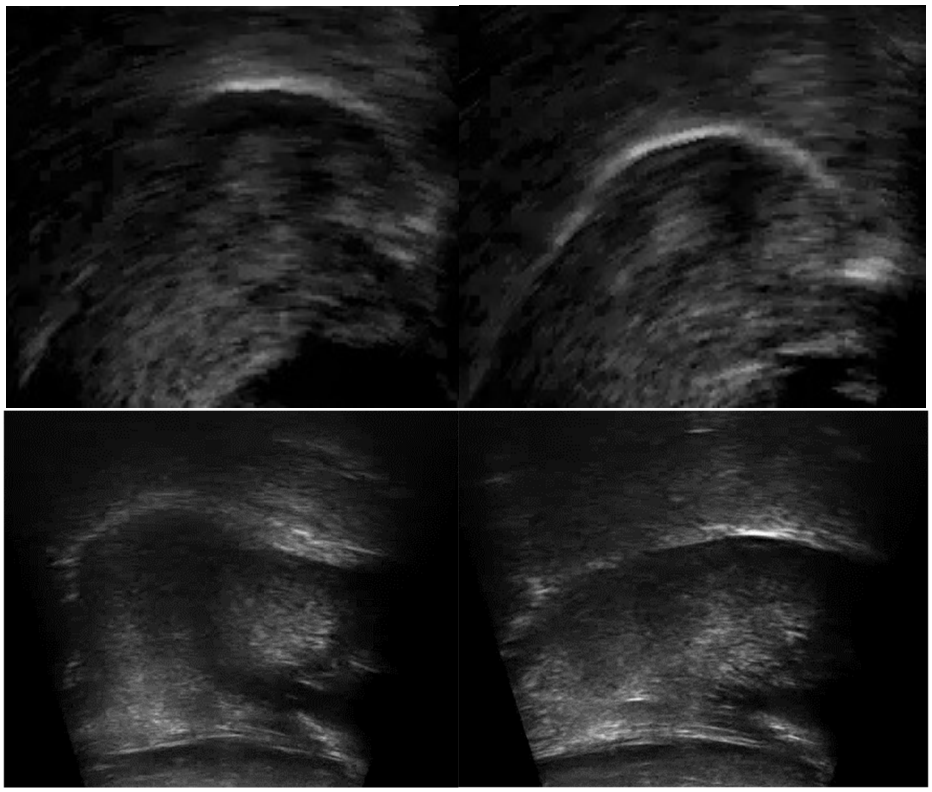

Figure 5-3 The comparison of the old (first row) vs. new dataset (second row)

After getting all the datasets, they are separately stored, so we can train on one dataset but validate on the other because, in real practice, each machine from different company, or with different probe and setting would have different image priors like the intensity of speckle noise, the penetration depth of the ultrasound would also be different, so the tongue contour characteristic would also be altered. Thus, it's important to have multiple datasets for evaluating the generality of the model by validating on a different dataset. In Figure 5-3, the first row is from seeing-speech and the second row is from our machine, they have been described in the previous section. In the

figure it's intuitive to see they have different image priors, the distribution of noise and tongue shape are slightly different, for a human if he/she is trained to recognize tongue in one dataset, it's easy to tell where the tongue is in another dataset, but it may not be easy for a machine learning model to do so. Thus, in the implementation stage, one main goal is to have a good performance on both datasets.

In total, 2058 raw video frames from our machine were extracted and annotated, with the same annotation method and processing standard applied with respect to the first database. So now in all, we have 4017+2058=6065 annotated frames as the dataset for training and evaluation.

Property / Dataset Name	A	B
Source	Seeing-speech.uk	Our Machine
Data Capacity	45 videos of UTI	5 videos of UTI
Device	Sonix RP	ULTRASONIX
Annotated data	4017 frames	2058 video frames
Total	6065 annotated videos frames	

Table 5-1 Summary of Dataset A and B

5.2 Implementation of sU-net

We assessed sU-net on the data which is described in the previous section. The total dataset contains more than 56 ultrasound videos which covered tongue motion articulating a set of phonetic alphabets. We created a semi-automatic annotation tool using a B-spline interpolation method along with Snake method, which was also introduced previously. The rectangular annotation mask was the same size and location for every frame.

The database was divided into a training set and a validation set with an 80/20 split ratio. The sU-net was implemented with the Keras library [105] and TensorFlow [106] backend. Adam [107], an algorithm for first-order gradient-based optimization of stochastic objective functions, was used with its default parameters, in which a learning rate of 0.001, β1 of 0.9, and β2 of 0.999, in which the β1 is the exponential decay rate for the first moment estimates. And beta2 is the exponential

decay rate for the second-moment estimates. The details of Adam optimization is not going to be covered due to the limitation of thesis content capacity. Lastly, a scheduled decay of 0.05 after each epoch, were adopted to achieve a better convergence.

The segmented output image has size (34 × 34) and is smaller than the input (128 ×128) due to various levels of convolution computation without zero padding. To equalize the sizes of input and output, the tile mirroring strategy introduced in the U-net paper [75] is useful, but it also slows down the performance. Thus, we retained the small output size and scaled up the final contour in our contour extraction step. Results and visual validation will be described in the following sections. In which we randomly selected some frames for the sake of fair presentation. But before we continue to the result, some important part of the building of the network, as well as the pre and post processing techniques, will be introduced and discussed.

5.2.1 Data Partitioning

All the available US images are divided into two main sets. In the first set, the data from the old and new dataset are treated equally and divided into training and validation data with an 80/20 ratio. The second partition the data from an online dataset and our machine, they are put into a different set for further investigation(which would be introduced in the next chapter). The method used for generating the data sets is described below. It is essential to keep training and validation data separate in predictive modeling, as it prevents too optimistic results or data leakage. There are two types of training schemes used here, the first is the “batch-training”, which means all of the data are put together and randomly shuffled, then it will be split into train/validation dataset by an 80%/20% ratio This scheme is to test the learning ability of the model, if the model can learn the relationships of all the available data successfully, then we can go to the next stage. To test the generality of the model: whether the model can learn the feature hidden behind the data. In order to do so, the dataset A and B would not be mingled together, for training, the model will only take in the dataset A and validate on dataset B, and vice versa.

5.2.2 Data Augmentation

Under most circumstances, more data means better performance and higher accuracy. In practice, the data is very hard to acquire, which requires the recording and preparing of raw data and manual annotation of them painstakingly, the data augmentation is a simple but useful way for getting more data to help to improve the network performance.

In terms of image, some most widely used technique for data augmentation include Rotation of images by some small degrees, e.g., 5 °~15 °, this small rotation usually won't compromise the data quality and is similar to the situations that can be encountered in reality. The resize of images, and the position shift is similar to the rotation. Both of them introduce more samples when the overall shape of data is unchanged. Lastly is adding the random noise into the images, when setting the appropriate parameters of noise, this often helps improve the robustness of the model. The intuition is as the model is said to be a "Blackbox," the visualization and interpretation of hidden layers are not easy, and some minor changes which won't affect human perception could have a huge effect on machine learning model. Thus, by introducing some minor gaussian noise, the performance of the model can be more robust. Here in this paper all of these popular data augmentation methods are implemented and tested; the result will be discussed in the according to section.

All the image augmentation are implemented with the online augmentation library: Augmentor [108]. It can provide various classic image augmentation algorithms easily. A rotation between -15 °~15 ° randomly is applied. Also, the mirror flip, random zoom between [75%,100%] of the image size and their combination are put into practice, in total, a new database of 10 times of the original images and their corresponding masks are generated. In Fig 5-4, one can see the result after applying data augmentation, random rotation (second row), mirror flip (third row), random zoom (fourth row), the operations applied on original images and masks are same.

So now in summary, according to section 4.1. We have a dataset of 4017 frames from online seeing-speech, and 2058 frames from our machine, after implementing the data augmentation, 10 times more data had been generated. So now in total, we have 44187 frames for seeing-speech, and 22638 frames from our machine. And the total database contains 66715 frames. They are all normalized and scaled with the preprocessing methods in the following section and saved as an H5py file for easier access.

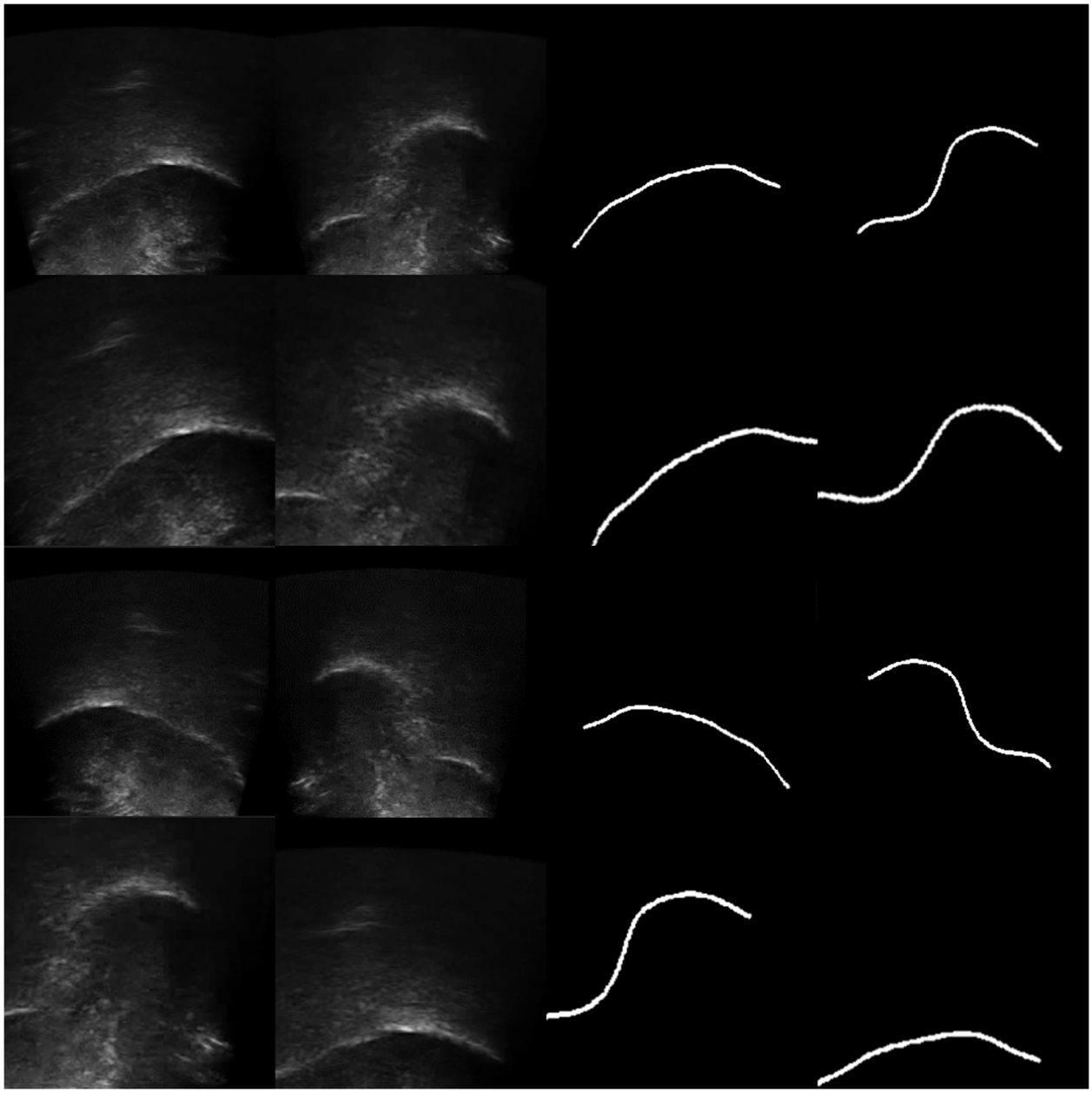

Figure 5-4 The original images and their masks (first row) after applying data augmentation.

5.2.3 Preprocessing

The preprocessing contains the reading and conversion of the input data format, resizing and other basic operation. Normalization is the most important technique in the preprocessing part which would affect the model accuracy. Normalization is a very helping method to improve the model to become invariant, that is, things which make no difference in the meaning of the symbol,

but only change the representation. And thus, to enable the model to gain more generalization capabilities

There are a large number of other types of transforms that can be used in processing variables in general. These can include:

- ***Scaling***

  Adjust all the variables to have a variance of 1, but sometimes a more robust estimator is used, such as adjusting to having a MAD(Mean Average Deviation) of 1. These resolve the need to learn the scaling invariance.

- ***Centering***

  Set the mean of a variable to 0; sometimes, it can instead adjust the median to be zero. The centering is applicable even if it can be assumed that the range of the variables is limited. The centering normalization can be used to remove the need to learn invariance of location.

- ***Whitening***

  Transformations to make variable distributions more Gaussian, such as the Box-Cox transform or using an empirical Gaussianization transform. Empirical Gaussianization is very flexible but has difficulty extrapolating (also can be interpreted as generalizing) beyond data already seen. These transforms resolve the need to learn distribution invariance of data.

- ***Stabilization***

  Transformations to stabilize the variables that are often nonstationary. After the transform, the noise levels will be nearly independent of variable values, and hence easier for machine learning model to generalize. These reduce the need to learn the amplitude invariance or more generally statistical temporal invariance.

- ***Component analysis***

  Various types of component analysis, such as PCA, ICA, etc. These can often be used (especially in temporal problems) to reject various types of noise. It can be interpreted as-as removing or reducing the need of the ML to learn invariance to SNR.

And here we will list some commonly used normalization formula:

- ***Minimax***

The min-max normalizer linearly rescales every feature to the [0,1] interval. The Rescaling to the [0,1] interval is done by shifting the values of each feature so that the minimum value will be zero, and then dividing by the new maximal value (which is the difference between the original maximum and minimum values). This is also the normalization method which is adopted in this research work. The values are transformed using the formula given below:

$$\mathrm{z} = \frac{x - \min(x)}{\lceil \max(x) - \min(x) \rceil} \tag{4.2.1}$$

- ***Logistic***

The logistic curve could simulate the exponential growth of an entity, at the first stage it would grow exponentially then begin to saturate, the growth rate decreases, at last, the growth tends to stop when the system is matured:

$$\mathrm{z} = \frac{1}{1 + \exp(-x)} \tag{4.2.2}$$

In this research, the MinMax method is used for normalization. The reason is MinMax has a relative easy expression allowing a fast computational speed. Also, it has a stable and satisfactory output result.

5.2.4 Network Architecture

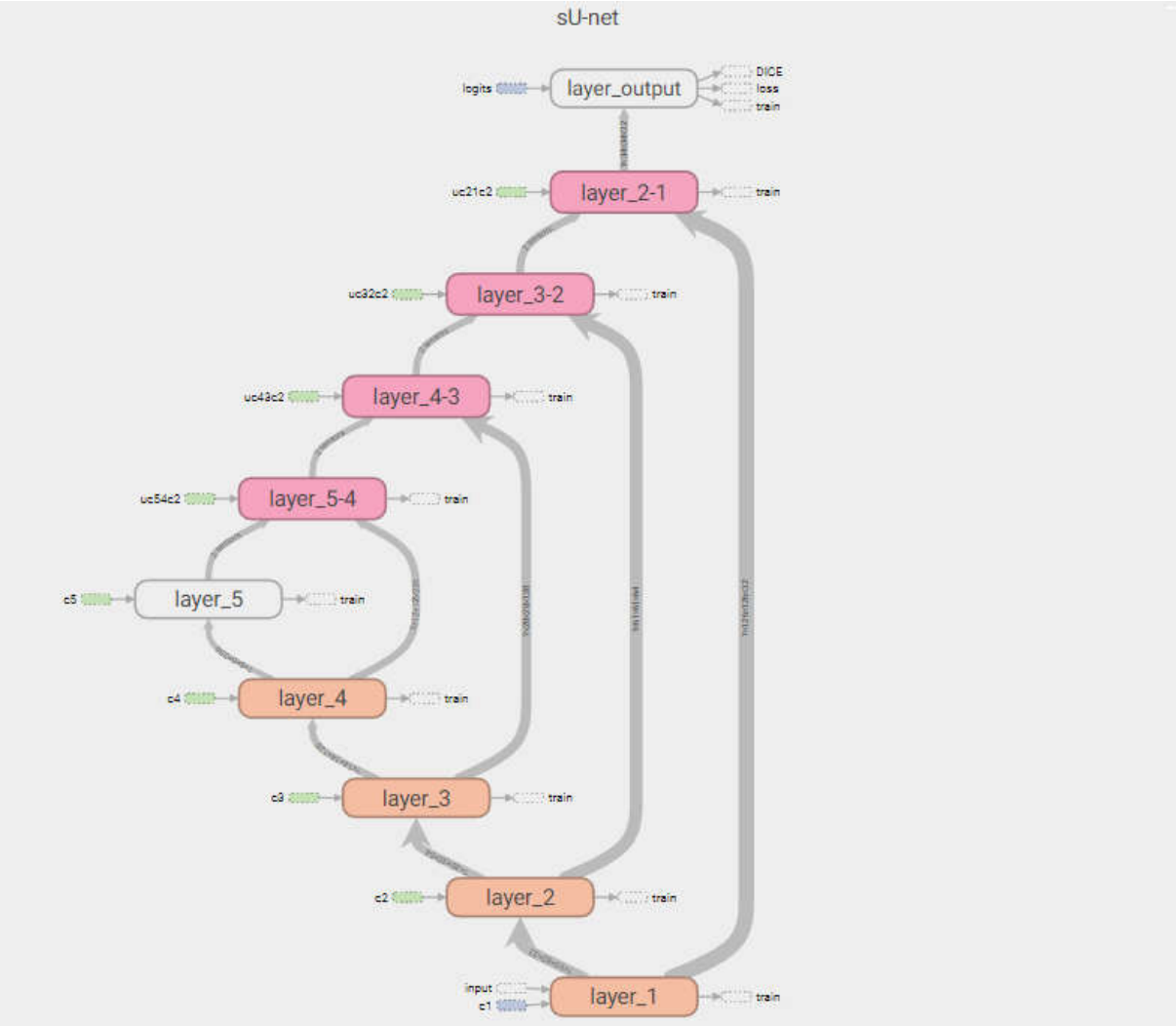

Figure 5-5 The network architecture visualized by the tensor board [102]

The model architecture had been introduced in section 3.3, hereafter implementation, the visualization by tensor-board, a built-in tool for deep learning metadata visualizing and event/histogram recording can give us a better intuition on how the nodes are connected. The layer-wise introduction and input-output relation had been documented in the following table.

Layer Group	Layer	Kernel/pool size	Output Size
	Input	N/A	N_s ×128×128×1
Layer_1	Batch Norm	N/A	N_s ×128×128×1
	2D-Conv (32)	(3x3)	N_s ×126×126×32
	Maxpooling	(2x2)	N_s ×63×63×32
Layer_2	2D-Conv (64)	(3x3)	N_s ×61×61×64
	Maxpooling	(2x2)	N_s ×30×30×64
Layer_3	2D-Conv (128)	(3x3)	N_s ×28×28×128
	Maxpooling	(2x2)	N_s ×14×14×128
Layer_4	2D-Conv (256)	(3x3)	N_s ×12×12×256
	Maxpooling	(2x2)	N_s ×6×6×256
Layer_5	2D-Conv (512)	(3x3)	N_s ×4×4×512
	Concat&Upsample	(2x2)	N_s ×8×8×768
Layer_5-4	2D-Conv (256)	(3x3)	N_s ×6×6×256
	Concat&Upsample	(2x2)	N_s ×12×12×256
Layer_4-3	2D-Conv (128)	(3x3)	N_s ×10×10×128
	Concat&Upsample	(2x2)	N_s ×20×20×128
Layer_3-2	2D-Conv (64)	(3x3)	N_s ×18×18×64
	Concat&Upsample	(2x2)	N_s ×36×36×64
Layer_2-1	2D-Conv (32)	(3x3)	N_s ×34×34×32
Layer_output	Output-Conv (1)	(1x1)	N_s ×34×34×1

Table 5-1 The network architecture specified by group name and layer-wise feature map (tensor), including layer name, kernel size, and output size

Table 5-1 above corresponds to Figure 5-5, with a more detailed description of the network parameters, the layer group reflects the connectivity of nodes in the figure, and the layer-wise input-output relations are also included. The introduction of Convolution Layer (2D-Conv) and the max-pooling layer can be found in Section 3.2. Their kernel or pool size is in the third column of the table above. For the output size, N_s is the number of training samples (or batch size) based on the different training scheme. Because the filter size is (3x3), after each convolution operation,

the output size would be reduced by original size – filter size +1, and the max-pooling size is (2x2), thus the output size would be divided by 2. The up-sample can be viewed as the reverse operation of max-pooling, in the original paper the deconvolution is used, but here we simply use the up-sampling to reproduce the original sized output.

5.2.5 Hyperparameter Tuning

In deep learning structures there are millions of parameters to be trained, for this reason, the deep neural network structures could be notorious for convergence is often hard to reach. So hyperparameters need to be tuned manually for the best performance. The hyperparameters which are listed in the following table were introduced in Chapter 4, and the optimal value of, e.g., Initial learning rate and Dropout are chosen by experiments.

Hyperparameter	Setting
Activation function	ReLU [89]
Output layer	Conv-sigmoid
Weight initialization	Glorot normal [109]
Convolution border mode	Same
Stride	(2×2)
Kernel size	(3× 3)
Dropout	0.5
Optimizer	Adam
Initial learning rate	2e-4
Batch size	50
Weight regularization	None

Table 5-2 Setting of the hyperparameters of the model

5.2.6 Training Process

The weights of the neural network learned in the training process largely depend on the content of the training dataset. Successfully accomplishing the segmentation task depends on the information that the network is exposed to. Therefore, when the training/validating dataset is properly applied to the model, the model began to compute the difference between output and the

ground truth mask, as discussed in Section 3.2 on a loss function. The training scheme is also important to the performance of the model. Ideally, all the available training dataset should be sent to the model, and the batch gradient would be computed, but the batch gradient is too exhaustive in practice, the time of training could be unacceptable under many circumstances. And for stochastic gradient descent, although the training time for each epoch is much faster, the gradient direction could be inconsistent with the global optimum [92]. The mini-batch gradient descent provides a balance in the aforementioned methods and could reach a good performance in both training time and training results.

So here in this thesis, the mini-batch training is implemented, the detail of mini-batch gradient descent can be referred to in Section 3.2. Figure 5-6 and Figure 5-7 shows us the visualization of the training processing. The first column denotes the original image, the second column denotes the ground truth mask, and the last row denotes the output from the final layer. Each row corresponds to epoch 0, 2, 4, 8 respectively in Figure 5-6, and 10, 20, 30, 40 in Figure 5-7 respectively. The optimization is clearly illustrated during the training process, and after 30 epochs, the performance becomes stabilized and precise.

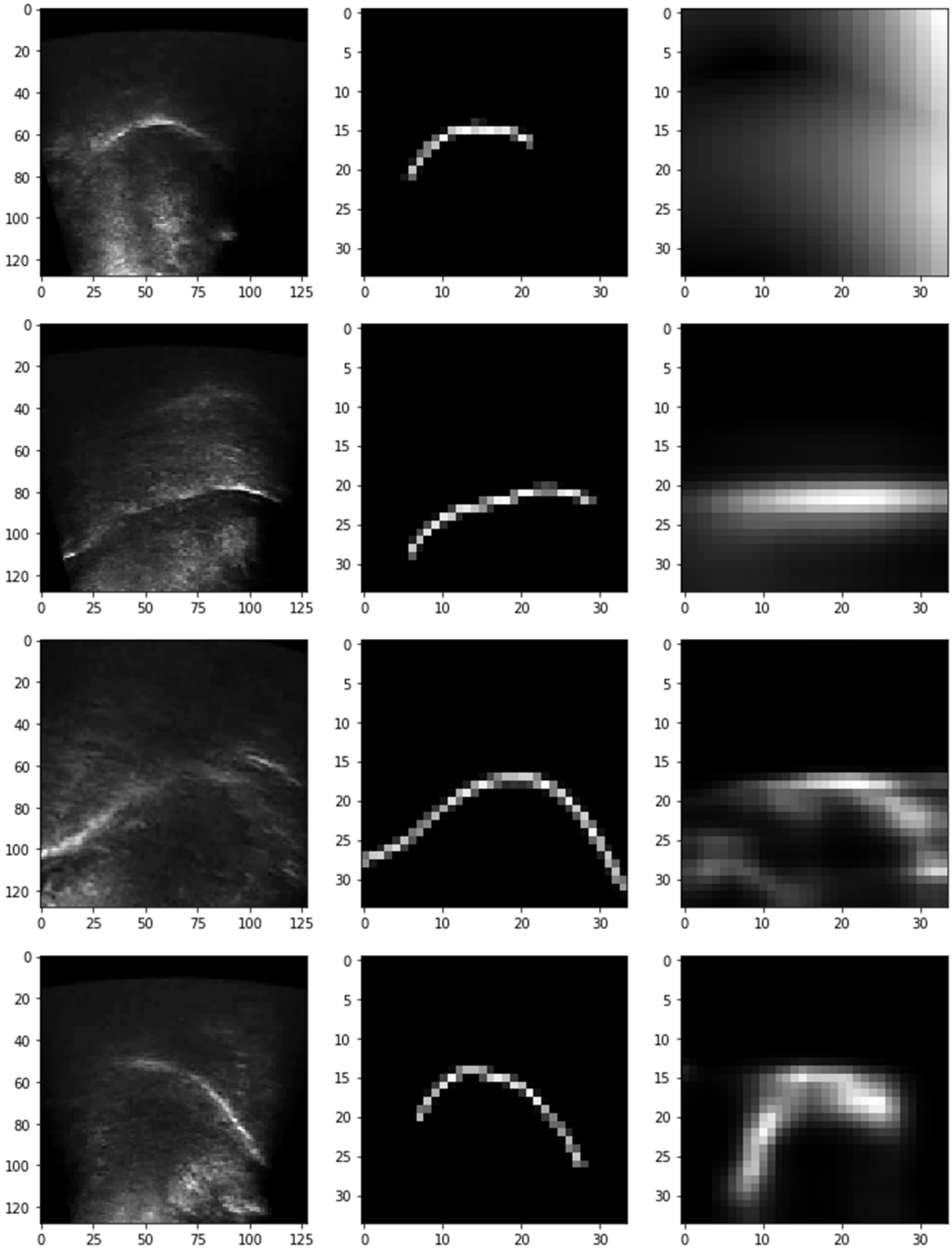

Figure 5-6 The output layer visualized by tensor-board during training, (epoch 0,2,4,8)

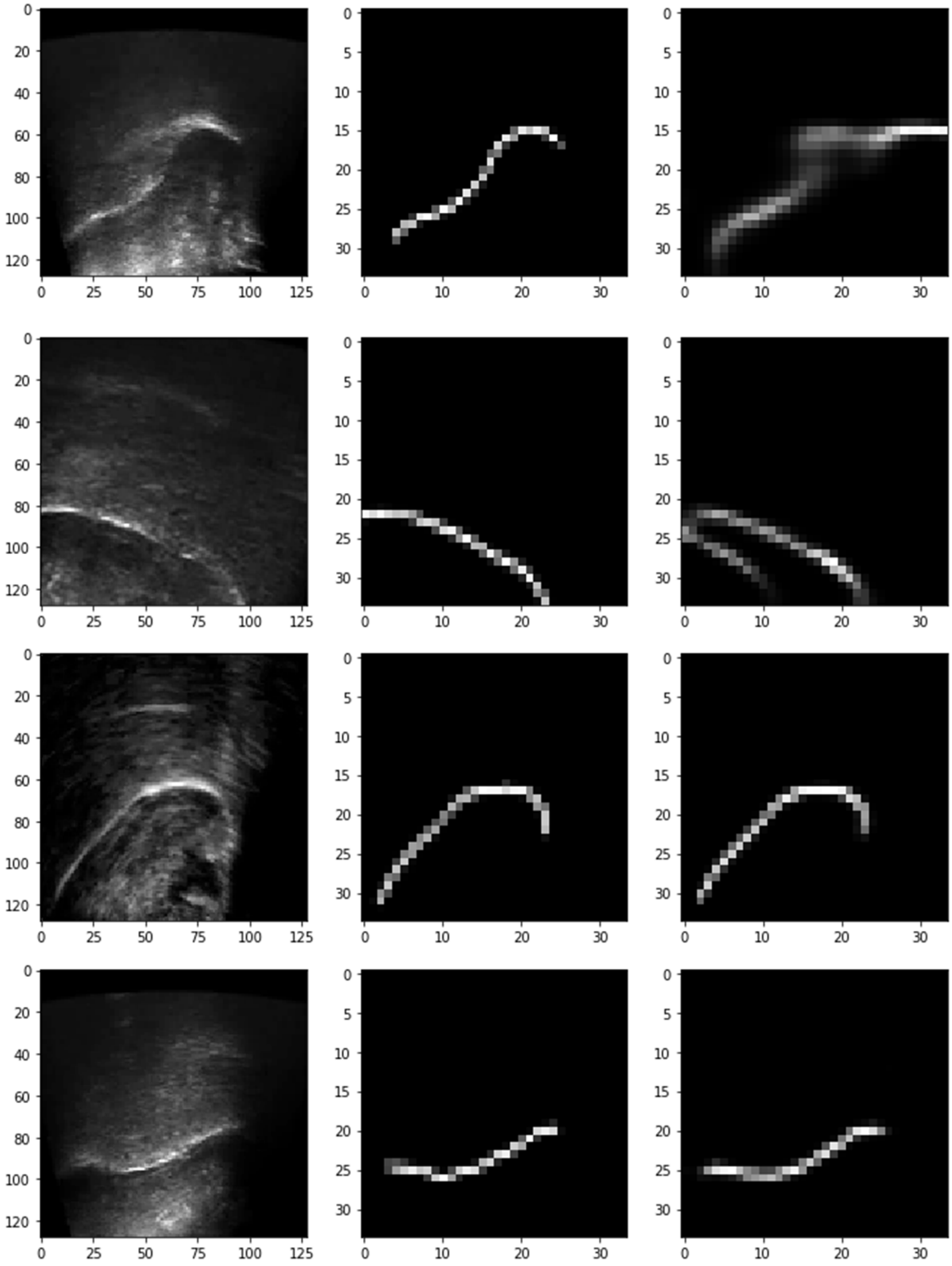

Figure 5-7 The output layer visualized by tensor-board during training, (epoch 10,20,30,40)

5.3 Evaluation Metric

We used Dice coefficient loss as the loss function. This calculates the difference between the prediction and the labeled data, as defined in the following section. To evaluate the performance of the proposed method, we calculated the Dice coefficient and MSD at the same time; the Dice coefficient is a straight and direct metric showing how good the training is done and how similar the predicted mask is from the ground truth mask.

5.3.1 Dice-Coefficient

The Dice Coefficient is a metric for evaluating the overlapping area between the group truth mask and the predicted mask; also, recently it has been discovered that it can be used as the loss function for machine learning algorithms. The original use of Dice Coefficient was proposed in the year 1945 as a statistical index, or known as Sørensen–Dice coefficient which is named after their inventor *Thorvald Sørensen* and *Lee Raymond Dice* [111]. The dice coefficient has many other names; the most popular one is the F_1 score in the F test. Before continuing to the discussion of the dice coefficient, is very useful to go to the F_1 score first.

The F_1 score may be the most widely used metric in both statistics and machine learning area. For the statistics, the description of the F_1 score is based on Sensitivity and Specificity, as for the machine learning (pattern recognition), the Precision and Recall are often referred to, they are related as in the formula below:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3.1}$$

$$Specificity = \frac{TN}{TN + FP}$$

Where the definition of TP, FP, TN, and FN are given in the following table:

	Actual value	
	Positive	**Negative**

Predicted value	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Table 5-3 Table of Positive/Negative Matrix (Confusion Matrix)

It's obvious that the Precision and Recall is the group of samples that the numerators are true positive, regardless of whether the denominator is positive or negative, the total number of selection (TP+FP) and the total number of properly chosen are used respectively for the evaluation of the proportion of the accuracy. F_1 score merges this two metric (Precision and Recall) with the same weight (β=1), which is also known as the harmonic mean of precision and recall for the balance of information contained in precision and recall.

$$\frac{1}{F_1} = \frac{1}{Precision} + \frac{1}{Recall} \xrightarrow{yields} F_1 = \frac{2PR}{P+R} \xrightarrow{yields} F_1 = \frac{2TP}{2TP + FP + FN} \tag{5.3.2}$$

Now, we can continue the discussion of the common expression of the Sørensen–Dice coefficient:

$$\text{Qs} = \frac{2|X \cap Y|}{|X| + |Y|} = \frac{2TP}{2TP + FP + FN} \tag{5.3.3}$$

Qs is the abbreviation for Quotient of Similarity, which is the value of the Dice coefficient, it will only range between [0,1]. In the area of image segmentation, the mask segmented by the model will be the total number of selection from the original image, and the mask annotated by a human expert will be the total number of correct pixel. So that we would know from the formula that the *TP+FP* will be *X* (which denotes the prediction of the model) and the *TP+FN* (which denotes the ground truth label) will be *Y*, their intersection will be *TP*. Thus, the Dice coefficient is equal to F_1 score substantially. Intuitively speaking, they both computed the similarity of *X* and *Y*, but inherently, both precision and recall were hiding inside both of them.

Like the Dice coefficient, the Dice coefficient loss is actually the reciprocal of Dice coefficient. When Dice coefficient approximates 1, a higher similarity between segmentation result and the ground truth label can be expected. So, because the model tends to find the minima in the data space, the common Dice coefficient loss function will be *Loss=1-coefficient*, or *Loss= -coefficient*.

5.3.2 Mean Sum of Distance (MSD)

For another comparison metric, we used the mean sum of distance (MSD) [21] as shown in equation (4.3.1). It provides an evaluation metric defined as the mean distance between pixels of a contour U and a contour V, even if these curves do not share the same coordinates on the x-axis or do not have the same number of points. The schematic of MSD is illustrated in Figure 5-8. We carried out contour extraction using the skeleton method (as described in Section 3.4) on both mask image manually obtained for validation purpose and predicted image obtained in the testing step. The latter one consisted of a contour set U of 2D points ($u_1, \cdots, u_n$) and the former one consisted of a contour set V of 2D points ($v_1, \cdots, v_n$). The computation of the MSD is given in the formula below:

$$MSD\ (\mathrm{U},\mathrm{V}) = \frac{1}{m+n}\left(\sum_{i=1}^{m} \min_{j} |v_i - u_j| + \sum_{i=1}^{n} \min_{j} |u_i - v_j|\right) \tag{5.3.4}$$

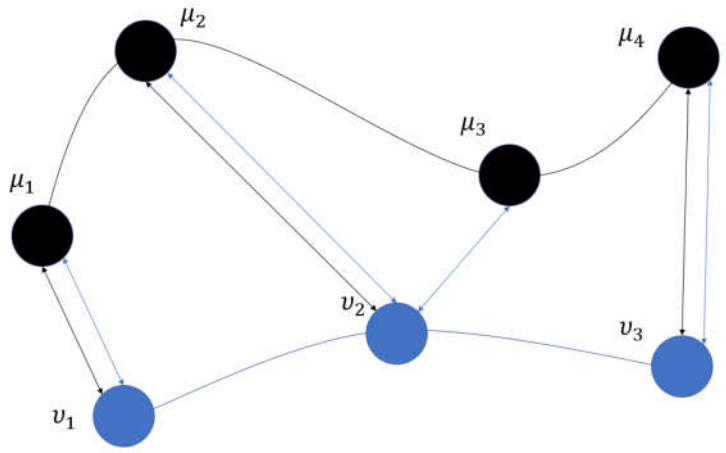

Figure 5-8 Comparing two tongue contours using MSD would enable the comparison between two tongue shapes even if the number of points of the two contours is not the same.

The mean sum of distance (MSD) only gives the distance in terms of pixels, but the sampling range could be different due to different probe type for device setup, so in order to standardize the MSD for comparison, it's important to convert the MSD (pixel) to MSD (mm), by inspecting the sampling range of the device and compute the coefficient for conversion. The distance in millimeter could be derived. For the online dataset, such conversion coefficient is 1px=0.295mm [112], for our device, such coefficient would be 1px=0.638mm.

6 Training Results and Evaluation

The training results are shown in Figure 6-1, Figure 6-2, and Figure 6-3 respectively. In Figure 6-1, the train and test loss are put into visualization; the loss metric is Binary Cross entropy here, a lower number means the better performance, it's obvious that the loss is decreased over time, indicating the learning process of the network. After 3k steps, the loss converges at a value of 0.026 for the training set and 0.028 for the validation set. Showing that the training process is work properly because the loss is decreasing and finally stabilized.

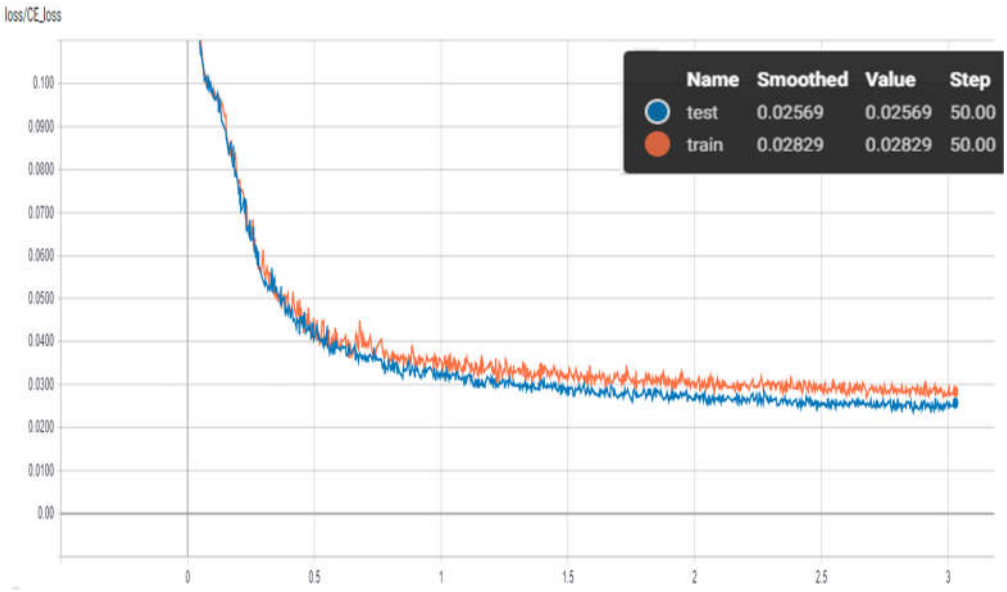

Figure 6-1 Training loss (BCE) on the merged dataset. The horizontal axis denotes the epoch number in (k). The loss had been decreasing during the training process and converged around 3000 epochs, where we stop training.

In Figure 6-2 the Dice-coefficient for training is depicted, the tendency is the opposite with loss graph, the Dice-coefficient denotes the overlapping area between prediction and mask, a value closer to 1 is better. And finally, after 3k steps, the Dice-coefficient reaches 0.942 for the training set and 0.914 for the validation set. It had validated that the prediction capability of the model is promising for a high value of the overlapping area is reached. For the detailed description of Dice-coefficient, please refer to section 6.2.

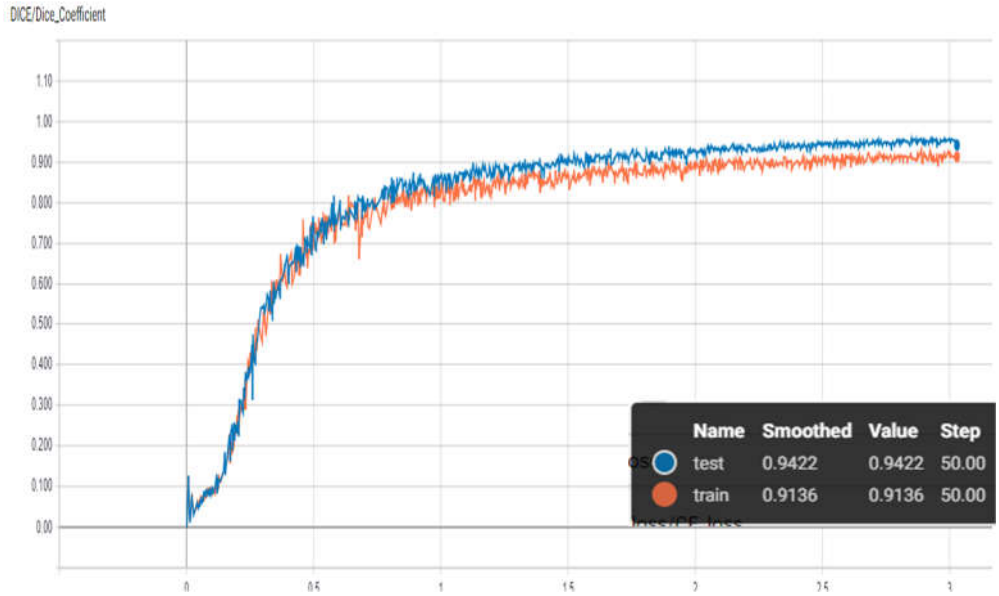

Figure 6-2 Dice-Coefficient After training. The higher dice coefficient means greater matching between prediction and manually annotated mask, which denotes the model learned to predict correctly.

And Figure 6-3 is a sample for the model output; the left column is the collection of raw images which are also the input of the model. The column in the middle shows the masks which are obtained manually, the column on the right is the prediction from the proposed model. It's easy to observe that the topological shape is well preserved, and the high accuracy prediction of shape is achieved. Although there are tiny differences in the output from the original mask, those differences are hard to tell with human perception. Such performance is enough for the purpose of the project.

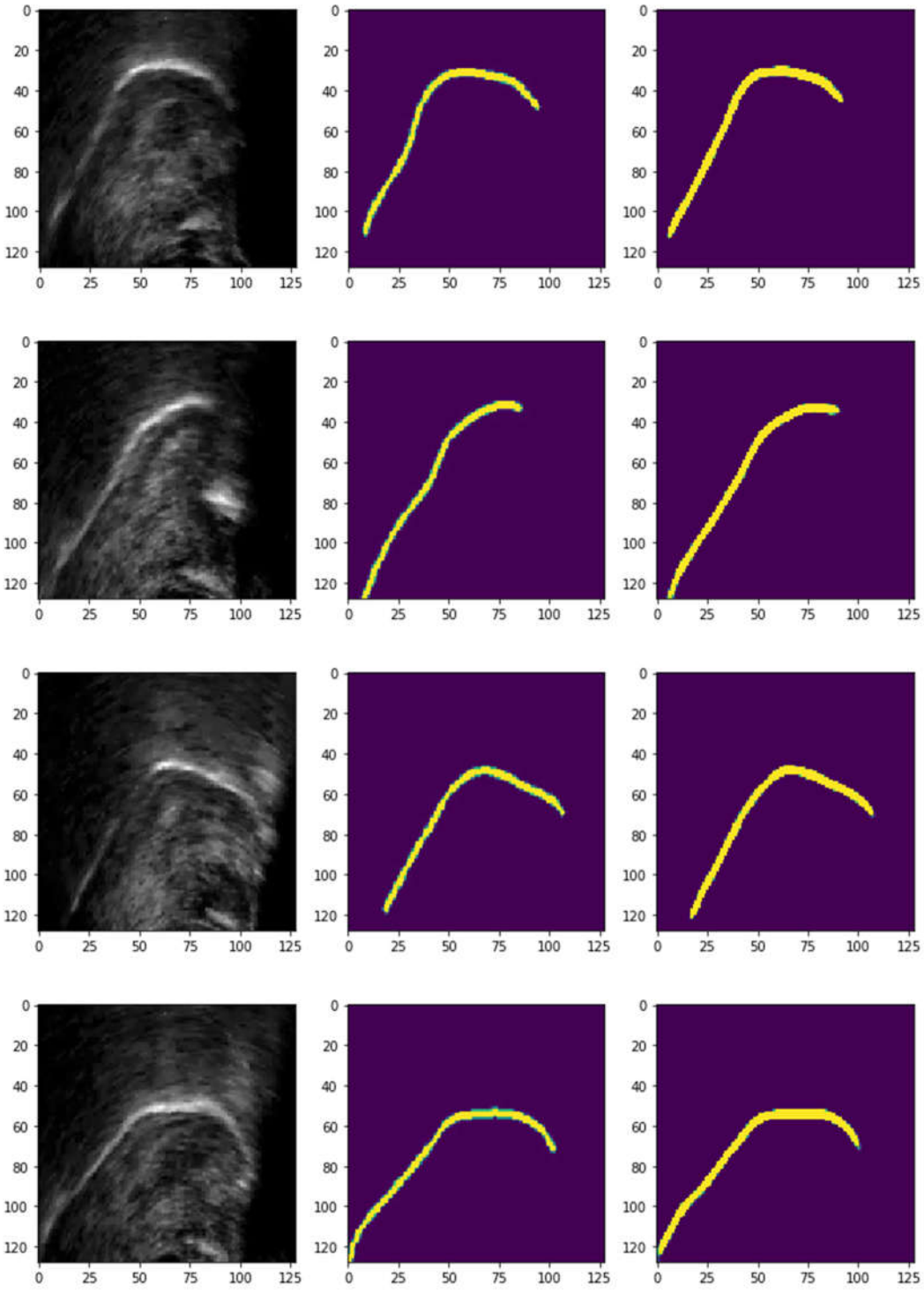

Figure 6-3 Various examples of prediction, including the original image (left), the manual mask (middle) and the prediction given by sU-net (right)

6.1 Contour Extraction

Because the predicted mask of the tongue is an area, parametrization of contour to be one-pixel-wide curve is made. By doing so we can compare our result with other researchers easily, and also one-pixel wide curve could be helpful for future research that may need to extract the moving and shape pattern of the tongue.

6.1.1 Thresholding and Blob Detection

Thresholding is a very important step which enables the various other postprocessing algorithms including the skeletonization. Because the output layer (section 4.2.5) of the deep learning model (Section 3.3) is a 2D-conv layer with ReLU as the activation function (Section 3.2), the output is actually a 2-D array with the range in between [0,1], it can be scaled to [0,255] to be visualized as a grayscale image, this value indicates the confidence of whether a specific point belongs to the tongue or not, higher the output value is, higher the probability it possesses. In order to extract the contour information from the predicted mask, binary thresholding is implemented, it will rule out all the points with intensity value lower than the given threshold while keeping all the points that are higher or equal to the threshold. The threshold value is settled by experiment, after multiple tests, it is found that binary thresholding of 0.1 (the intensity value higher than 0.1, corresponding to a grayscale value higher than 26) will give the best performance. The figure depicting the performance of thresholding will be given in Figure 6-4.

Sometimes the performance of the network is not always trustworthy, there can be multiple activated areas in the output images, in observation, it is found that in most circumstances the tongue shape would be correctly predicted, only with some small artifacts merged inside, so it is important to remove such artifacts using image processing techniques. The blob [110] in image processing, is said to be a group of connected pixels in an image that shares some common property (e.g., grayscale value), because the corrected prediction of the tongue is almost always the biggest component in the output image, as long as we labeled all the blob region in the output image, then remove all the small regions with respect to the area size. In the implementation, all the blob area will be labeled, and the maximum area size will be recorded, then, all the blob area that is smaller than the maximum area will be removed. Thus, a single connect region denoting the tongue will be given, and the performance in terms of MSD (refer to Section 4.5) will also be improved.

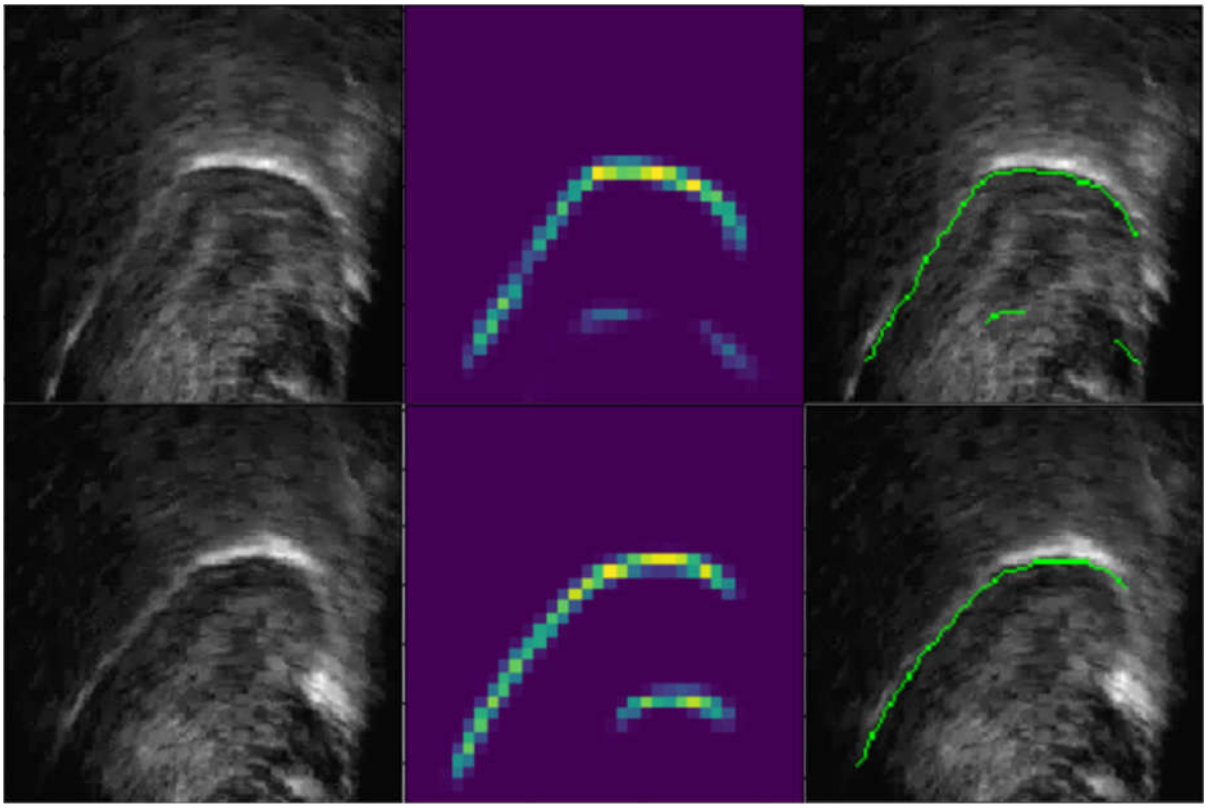

Figure 6-4 The final output without (first row) and with (second row) Blob detection, the blob suppression will only keep the maximum blob for removing artifacts

6.1.2 Skeletonization and Superimposing

The method of skeletonization had been thoroughly introduced in section 3.4. Here the emphasis would be on the implementation and performance of skeletonization. As discussed in the previous section, skeletonization is a combination of erosion and dilation; the processing continues until the stopping criteria are met, that the input shape to be thinned until a single pixel wide. First, the structuring element for erosion and dilation is to be settled, here a (3x3) morphological cross shape structuring element is adopted, and the threshold is set to be 31 after testing.

After the skeleton of the image is extracted, the skeleton contour will be superimposed onto the original image for better visualization, and as the final output, these final superimposed images can be used as images in guided learning tool to help second language learning or language impairment rehabilitation. The post-processing steps are illustrated in Figure 6-5 below

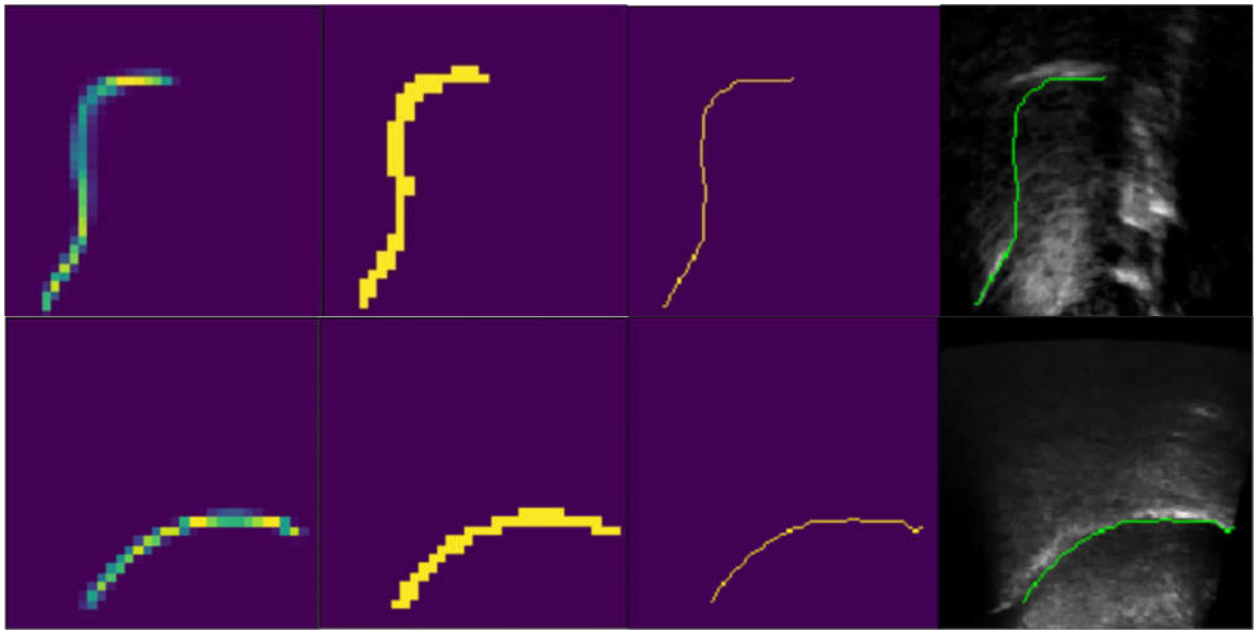

Figure 6-5 The postprocessing steps: from raw prediction array (left) to binarized mask (second), the extracted skeleton (third) and the final superimposed image (last)

6.2 Performance Comparison

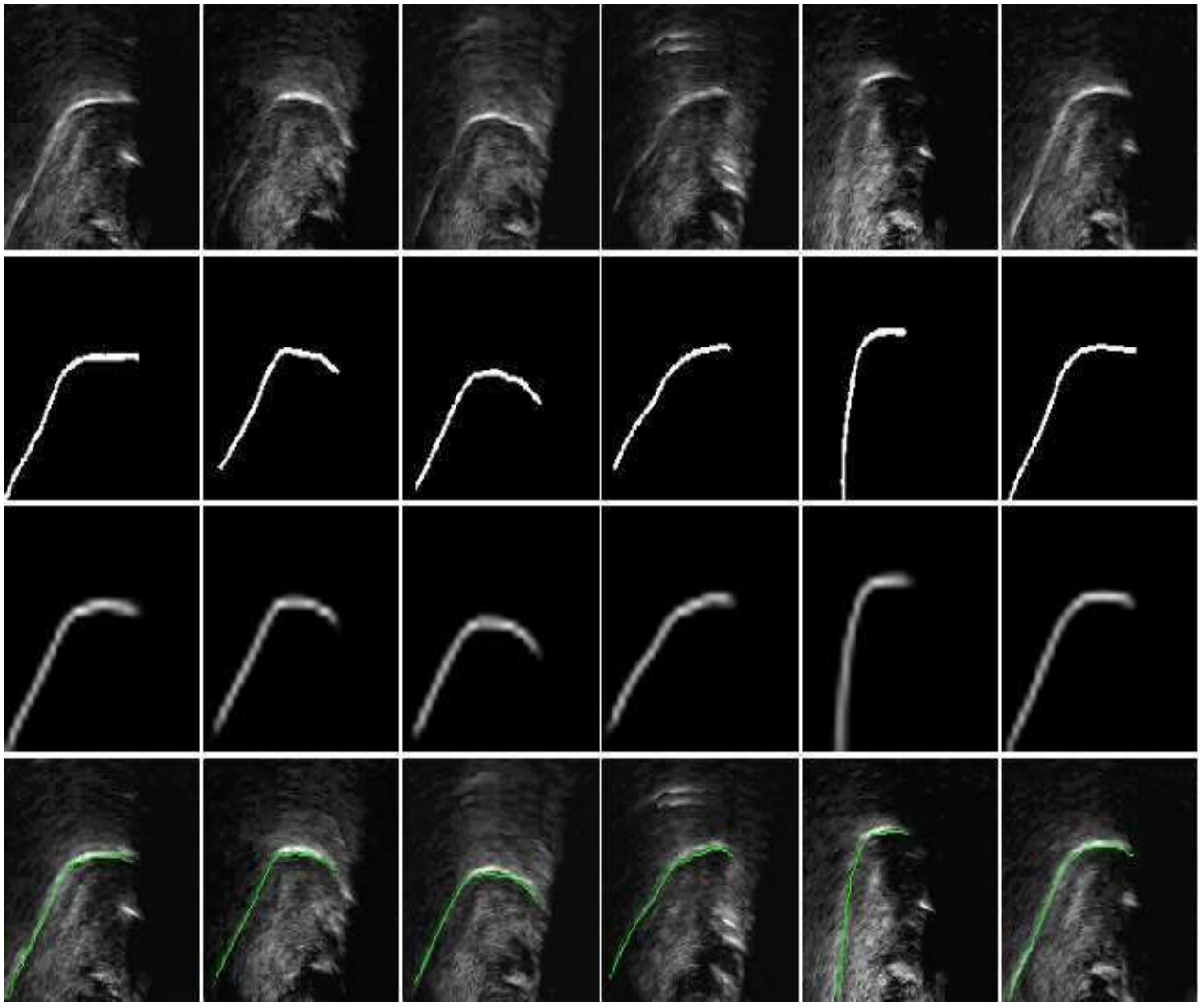

Figure 6-6 below shows the results of the proposed system of tongue contour extraction. The first row showed some randomly selected raw ultrasound frames (128 × 128), the second row is the annotation of tongue dorsum for each corresponding frame to the first row (128 × 128) used for validation, the third row showed the predicted tongue dorsum segmentation (34 × 34) that resulted from testing, the fourth row shows the scaled contours in green, extracted from the third-row images using the methods mentioned in Section 3.4. And then the extracted contour is superimposed on top of the original frames of the first row.

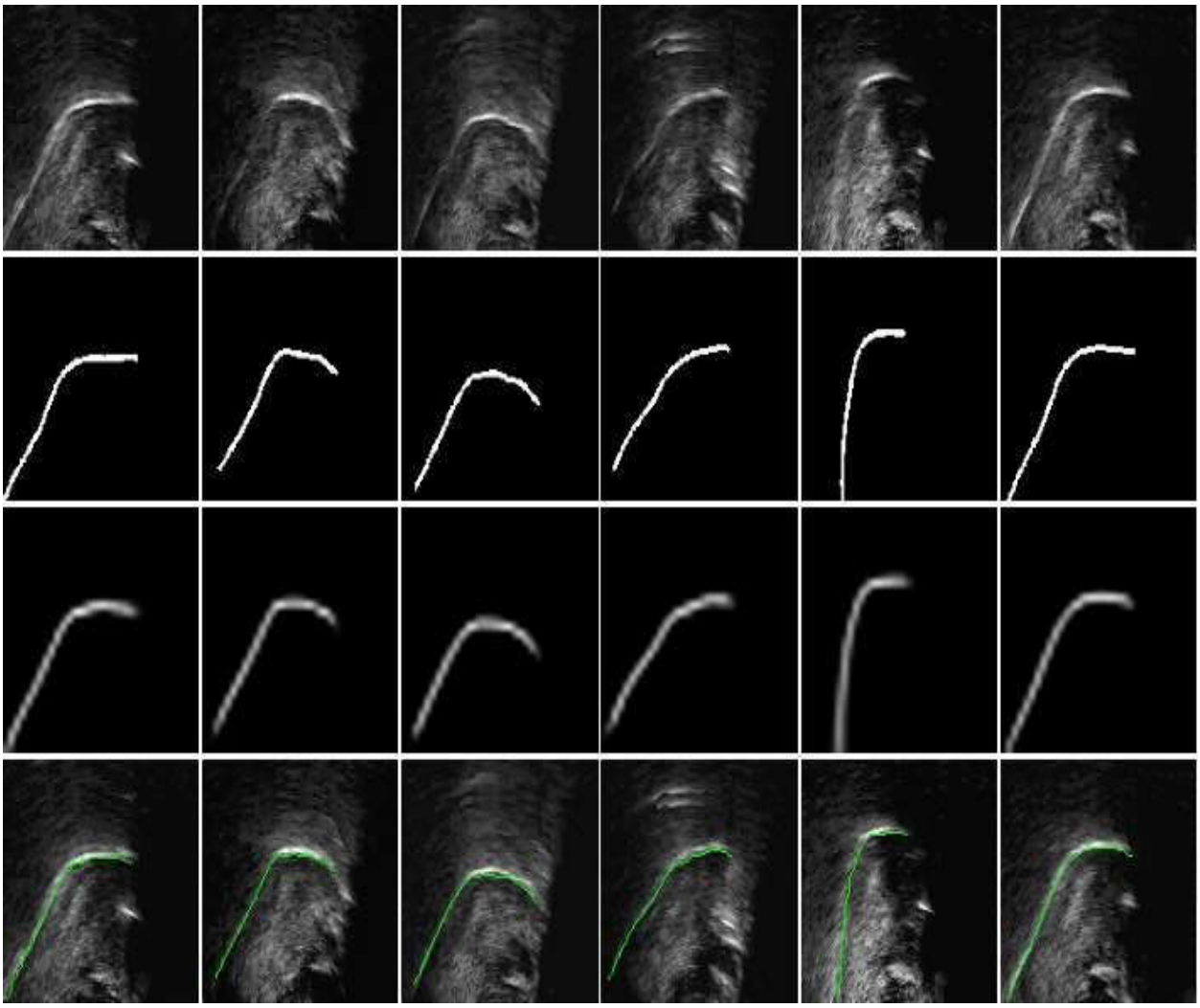

Figure 6-6 Results of ultrasound video segmentation, the first row contains the ultrasound image sequence, the second row contains masks from the manual annotation, the third row contains the prediction from our model, and the last row contains the superimposed contour to the raw frames.

The original training scheme is the "batch training" described in section 4.1, whereas all the obtain a dataset of 6065 frames are divided into training/validating at 80/20 ratio. The training process is described in section 4.2.7, and the training result is presented in section 4.3. After training, a dice coefficient loss of is achieved. And the MSD of 1.43px which corresponds to 0.91mm is also achieved. For a detailed description of MSD, the reader can refer to the previous section 6.2.2.

6.2.1 Comparison of Training Efficiency

We compared our experimental results in terms of training efficiency with the results of DBN [84] which has been used for ultrasound tongue segmentation. Table 6-2 shows the comparison for the Dice-coefficient validation error related to the number of epochs. The latter is related to error and performance speed. In our system, the error declines to around 0.04 when the number of epochs is around 100 and then dropped to around 0.02 when around 250 epochs. The DBN error remains around 0.4 even though the number of epochs increases. Our method outperforms DBN when the number of epochs reaches 50 and levels off at 100 epochs at around half the error of the DBN system. This meant that our architecture is better when comparing with DBN in terms of both training result, as well as the convergence speed, indicating the superiority of the proposed structure.

Number of epochs	**5**	**50**	**100**	**250**
sU-net (ours)	0.446	0.243	0.04	0.02
DBN [84]	0.41	0.38	N/A	0.40

Table 6-1 Comparison of our method and previous work in a Dice-coefficient validation error. From the table, we can conclude that not only the loss drops significantly faster than the DBN model, which proved a fast convergence of our model. But also the final result (a lower loss) of our model is significantly better than theirs.

6.2.2 Comparison of Original Result with Data Augmentation

From the training result and discussion in performance, it is shown that the proposed network is able to learn to fit both datasets, for the performance on the merged dataset is very good. Also, the training efficiency is competitive. But, in order to determine whether the structure is actually "learning," that the prediction is from the generalization of the given samples, instead of memorizing them all. Generalization power is an important feature because in practice it's possible that new types of data that the model wasn't training on could be applied. Thus, the intra/inter-data performance should be tested.

Two different experiments are done by different training schemes, for testing the intra-data invariance, the model is trained on the online dataset or dataset from our machine with basic augmentation, and tested on the combination of augmentations respectively. For testing the intra-data invariance, the model is trained on the augmented online dataset and tested on our dataset,

and vice versa. Before discussing the intra/inter-data invariance, the augmentation technique itself is put into comparison. Two performance achieved with and without data augmentation is listed as below:

Training Data	Training Loss	Testing Data	Performance
A+B	0.03	A/B	0.91
A+ B+ Augmentation	0.02	A/B	0.94

Table 6-2 The performance comparison for data augmentation, the augmentation helped improved the prediction accuracy. The Training loss is in terms of BCE loss (refer to 4.3.3, lower the better), and the performance is in terms of Dice Coefficient (refer to 6.2.1, higher the better).

From the table 6-3 one can see that the data augmentation helped improved model accuracy a little, indicating that the data augmentation won't hurt the performance at least, although the original performance is already satisfactory, further testing would be made to gain a better understanding of how the model would perform under different circumstances.

- ***Intra-data Invariance Testing***

First, the performance of each of the dataset themselves should be guaranteed, for the model performed well over all the available data. In practice, we know that the tongue position and orientation always change due to its own movement, the different poses of the probe, the machine setting, etc. It's impossible to list all circumstances exhaustively, thus the model is expected to learn the invariance from images: especially in terms of zooming (due to machine setting), flipping and rotation (from possible changes of probe direction). In the data augmentation part (section 4.2.2), the flipping/rotation/zoom are implemented, and also their combination. In testing of the Intra-data invariance of these parameters, the model is trained on the original data, and flipping/rotation/zoom augmented data. The test is made on the data with all the combination of the aforementioned operations. Thus, if the performance on the testing is good, we can draw the conclusion that the model has learned to be invariant to rotation, flip, and zoom.

From the table 6-4, the composition of training/testing data is specified in the association to the previous discussion, and the performance shows that the invariance within a dataset is obtained. In Figure 6-7, two sample of output is visualized, the model fitted well to the multi-translation/rotation data.

Method/Data	Training data	Validating Data
Rotation (-15°~15°)	Applied separately	Combination of Rotation/Flip/Zoom with a probability factor
Flip	Applied separately	
Zoom (0.75~1)	Applied separately	
Performance (Dice)	0.95	0.97

Table 6-3 The Intra-Invariance testing, when the model is trained with Rotation, Flip and Zoom separately, it is able to learn to predict a new image with all these operations combined, this showed that the model could learn to be invariant to such changes.

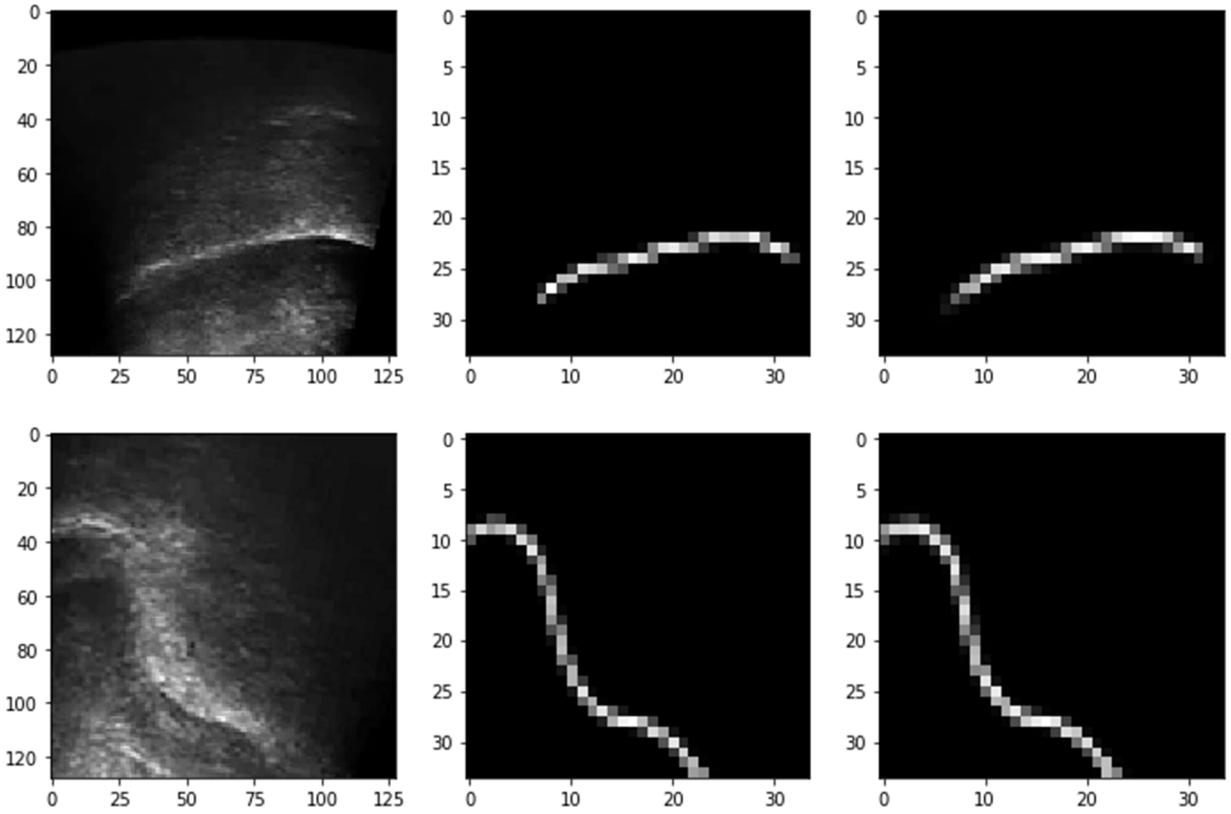

Figure 6-7 Test performance on the dataset whose part was used for training, the first column contains the raw input data, the second row contains the manual labeled mask, the third row contains the prediction from our model. It showed that the invariance of flip, rotation, and zoom is learned by the model.

- ***Inter-data Invariance Testing***

The second goal would be to get a more generalized model which can be transferred to a different machine or different probe setting. In section 4.1 two datasets are illustrated, the difference in image intensity distribution is intuitive to see. In testing, the generality between data is poor, seems that the model failed to learn the invariance from different image grayscale distributions. The reason could be due to the priors of the image are different which were discussed in Section 2.3.

Method/Data	Training data	Validating Data
Source	A	B
Performance (Dice)	0.95	0.31
MSD	0.93	23.22

Table 6-4 The Inter-Invariance testing, the model trained on one dataset performed poorly on the other, showed that the generality of the model is poor between different datasets.

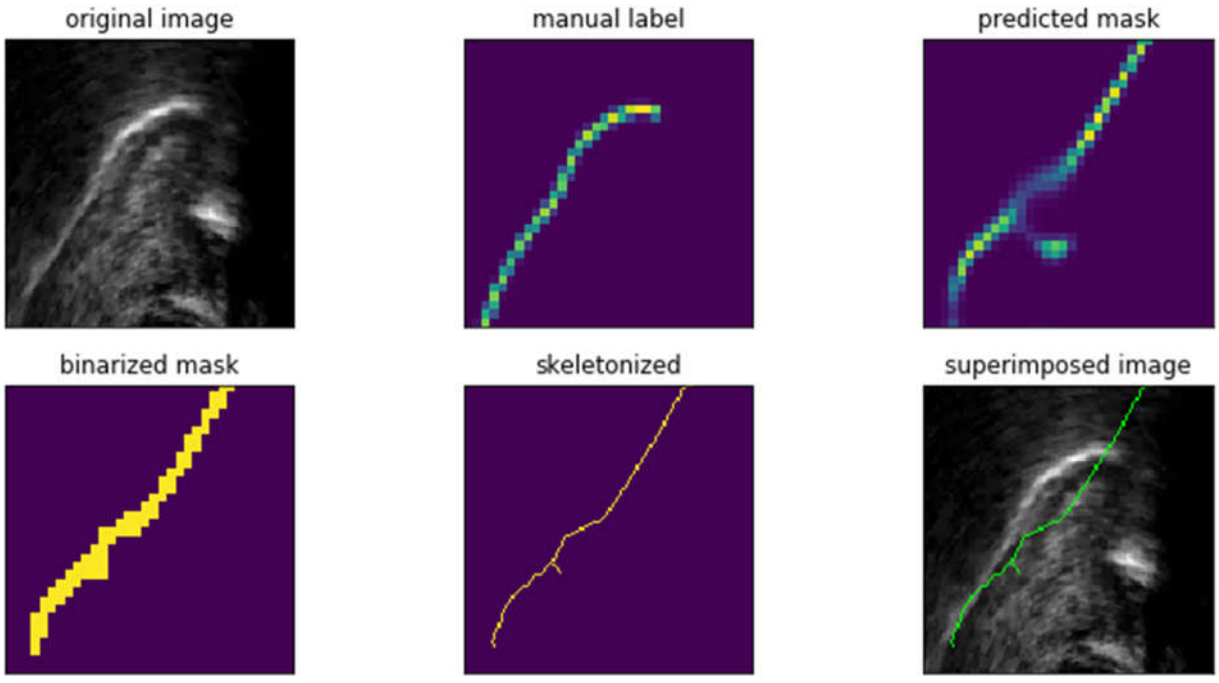

Figure 6-8 Test performance on datasets from different Ultrasound machine, meaning the validation set and training set are from different Ultrasound machines. The result is poor compared to the case of training, and test sets are from the same machine. A possible cause

is the drastically different image priors between two datasets, for the description of the ultrasound image characteristics, please refer to section 2.2.

The performance for the intra/inter-data has shown that the model could learn to be invariant to rotation/zooming and flipping when provided appropriate data, such intra data generality is an inherent feature of the CNN-based model [113]. And the poor performance of the inter-data is also very intuitive to explain: the data augmentation could properly model the inter-data characteristics. The differences between the two ultrasound images from different ultrasound machine are usually in terms of speckle noise and gray-level distribution, as reviewed in section 2.3. Rayleigh or K-distribution are proposed to model such noise. So, if the data is augmented a better generality accordingly between different dataset might be learned.

6.2.3 Comparison with Other Methodologies

Method	Metric	Raw result	Converted (mm)
Active Contour [16]	MSSD [114]	2.850 px	1.69
Edgetrak [60]	MSD	3.02 px	1.02
Active appearance [65]	RMS	3.02 mm	3.02
Quality Map [85]	MSD	0.87mm	0.87
TongueTrack [18]	MED	1~2.5 mm	1~2.5
Particle filter [15]	MSD	2.01 mm	2.01
Autotrace [21] [78]	**MSD**	**2.544 px**	**0.75**
ANN [79]	MSD	1.29 mm	1.29
RBN [61]	MSD	1.0 mm	1.0
Expert 1vs. Expert 2 [60]	MSD	2.910 px	0.86
Proposed Method	**MSD**	**1.43 px**	**0.91**

Table 6-5 The error comparison with other methods. Our model is worse than the Autotrace and human label, but the runtime efficiency for our model is better than Autotrace, which will be compared in the next section.

The active contour may be the first research ever carried out on tongue contour tracking, they used the MSSD as the metric, as mentioned in [114], in the paper they proposed two scenarios, here only the results from the first scenario is reported, they tested on 15 subjects and have a mean value of MSSD of 2.850, because MSSD is very similar to MSD used in this paper, only difference is the terms are squared, we convert the result by apply the root, and get a converted result of 1.69. But because in the paper they failed to mention whether the result is in pixel value or mm, so the comparison is of less confidence. The Edgetrak used the same metric as used in our research, results on 6 subjects were given, a mean value of 2.611mm can be derived. In their research, the result comparison of two experienced Ultrasound device users is also given, the difference between two humans can be different as 2.91mm. This could be a result of the difficulty for some specific ultrasound images presented to the expert, because in some videos, the tongue movement can be inconsistent, causing the annotation task harder. In [65], the RMS (root-mean-square) of distance is used as a metric, and a mean result of 200 continued frames is presented. It is worth mentioning that for all active contour-based methods, the performance tends to degenerate over time after manual initialization, which means the manual initialization should be applied from time to time for maintaining the performance, thus, making the system semi-automatic. In [18], the MED (mean Euclidean distance) and several other metrics are used, for simplicity we report only MED here, in the original paper they report a minimal error of 1-2.5mm for one dataset, and a minimal error of 2.0-3.8mm for another more challenging dataset, thus we adopted the first one here.

As a conclusion of section 2.5. Here we see again that the machine learning based algorithms performed much better than the traditional methods, in [21] [61] [19]. Various machine learning architecture was proposed, e.g., RBM (Restricted Boltzmann Machine), DBN (Deep Belief Network) and ANN (Artificial Neural Network), they all reach to be fully-automatic and a performance which is comparable to human performance [60], or even better [21][50]. Still, the proposed method of sU-net is among the top of all the state-of-the-art algorithms. Our proposed method outperformed other methods with regard to MSD as shown in Table 4-6, only slightly worse than Autotrace. The MSD value in terms of pixels was 1.43 pixels for sU-Net, with a conversion of 1 px = 0.638 mm, giving an average MSD of 0.91 mm, while the DBN [40] model achieved 1.0 mm (1 px = 0.295 mm). For the active contour tracking method mentioned in [115], the average MSD is 1.05 mm. It is important to mention that the two human experts participating in active contour tracking experiment produced two different annotation results having an average

MSD of 0.86mm [60], which may thus be reasonably considered the standard tolerance of MSD for training based automated methods.

6.2.4 Runtime Efficiency

Method	User input	Runtime efficiency	Device/Platform
Edgetrak [60]	Points on the first contour	Semi-automatic	N/A
TongueTrack [18]	Points on first contour	Semi-automatic	Unknown/MATLAB
Biomedical Model/FEM [83]	Manual full tongue Delineation	Semi-automatic	N/A
Autotrace [21]	None	30fps-33ms/frame	Unknown/MATLAB
ANN/Eigen tongue [79]	None	Real-time	Unknown
Quality Map [85]	Manual Guidance	Semi-automatic	Unknown
U-net [73]	None	5.37fps-186.2ms/frame	Intel (R) Core (TM) i7-3770 CPU @3.40GHz/python-Tensorflow
Proposed Method	None	74.7fps-13.4ms/frame	Intel (R) Core (TM) i7-3770 CPU @3.40GHz/python-Tensorflow

Figure 6-9 The runtime efficiency of state-of-the-art methods. Our proposed method reached the highest runtime.

We compared the runtime efficiency of many states of the art algorithms described in [60] [18] [83] [21] [79]and [73]. For the detailed description of these algorithms, readers should refer to Section 2.4. For all the traditional algorithms in the thesis, because the manual initialization is inevitable, the real-time performance is greatly reduced, because trained personnel is required to monitor the processing all the time, and when the tongue moves too drastically, or the position deviates to fall from the original position, the active contour-based algorithms, or their variants tend to lose performance. Sometimes even the re-initialization is needed when the tongue is moving too fast [15]. Thus, for these algorithms, usually, the run-time efficiency is not discussed. Because the manual annotation would take up a lot of time, and this time differs with different people, which made it hard to quantify the run time efficiency.

Method	Total Parameters	Non-Trainable Parameters
U-net	7,760,097	0
sU-net	3,919,169	0

Table 6-11 Parameter Comparison between U-net and our sU-net

We ran our model on a Windows PC with a 7-Core CPU at 3.4GHz and 16GB of memory. In the testing stage, sU-Net provided the segmentation results of 175 frames in 2.343 seconds, which equals 74.7 fps. The testing performance goes down to 29.8 fps when we add tongue contour extraction. When compared with the state-of-the-art machine learning algorithms, the real-time fully-automatic property is usually a default setting for all of them. But among the papers that had discussed the performance, still, the proposed method (sU-net) is among the top of all. It is worth mentioning that the original U-net paper is also implemented, although the performance is desirable, the efficiency is restricted due to many redundant layers, by eliminating these redundant layers, the sU-net was able to reach to real-time fully automatic and outperformed many other algorithms in terms of accuracy. In table 6-11, we see that parameters of sU-net are only half of its original version: U-net, this gives the high processing speed of sU-net because one forward pass now will go through less computational operations.

7 Conclusion& Future work

7.1 Conclusion

In this research,a deep fully convolutional neural network structure is proposed, which is inspired by a popular and recently designed deep convolutional network for biomedical applications U-net. The proposed light version: sU-net, has been used for automatically segmenting tongue contours from real-time ultrasound video data. The choice of network structure was selected and modified in the course of several experiments decreasing the number of network layers and removing unnecessary steps to speed the process while keeping the high performance. The data acquisition and manual annotation are made. And then the normalization and other necessary preprocessing are applied to the dataset, with the proper partition of data and fine-tuning the network; an optimized performance was reached. With post-processing including thresholding, blob detection, and skeletonization, the error is further minimized, while the run-time efficiency is preserved to be real-time.

From the experimental outcomes, we can assert that the sU-net based on convolutional neural networks are superior to their counterparts such as deep belief networks in terms of performance and accuracy. A comparison has been made that the proposed method outperformed other traditional methods largely in terms of run-time efficiency, for most of them can't reach real-time or they constantly require manual initialization. For extracting contours from delineated frames, a skeleton technique was successful. An accuracy comparison is also made, which suggests that the proposed method is one of the top candidates and the best in all the algorithms that can run in real-time. Some more experiments are made, including the training efficiency and the generality of the network. It can be shown that the model can learn to be invariant to local scale, rotation and flip thanks to the data augmentation. The experimental results not only displayed the accuracy and speed of the proposed method but also suggest its potential for other ultrasound problems and other organs.

7.2 Limitation and Future Work

Although the model reached the real-time and high-accuracy, the limitation exists, and future workers need to be made for improving the performance. Firstly, the model only uses spatial information and neglects the temporal information which is very important in ultrasound image recognition. One possible solution is to build a physical model which simulates elasticity and inertia, so each prediction from one frame only provides a gradient to the model, and the temporal trend may be modeled in this way. Another limitation is that the model seems to not generalize well over different machines, although such limitation is inevitable due to the inherent feature of CNN and input data. Still, it can be a problem when someone wants to deploy the model directly on to different machines. One possible solution is that to model the speckle noise of the data, using Rayleigh or K-distribution, and making a new data augmentation for improving the generality.

Reference

[1] K. Richmond and S. Renals, "Ultrax: An animated midsagittal vocal tract display for speech therapy," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[2] K. Xu *et al.*, "Robust contour tracking in ultrasound tongue image sequences," *Clin. Linguist. Phonetics*, vol. 30, no. 3–5, pp. 313–327, 2016.

[3] B. Gick, B. Bernhardt, P. Bacsfalvi, and I. Wilson, "Ultrasound imaging applications in second language acquisition," *Phonol. Second Lang. Acquis.*, vol. 36, pp. 315–328, 2008.

[4] B. Denby, T. Schultz, K. Honda, T. Hueber, J. M. Gilbert, and J. S. Brumberg, "Silent speech interfaces," *Speech Commun.*, vol. 52, no. 4, pp. 270–287, 2010.

[5] D. Fabre, T. Hueber, L. Girin, X. Alameda-Pineda, and P. Badin, "Automatic animation of an articulatory tongue model from ultrasound images of the vocal tract," *Speech Commun.*, vol. 93, pp. 63–75, 2017.

[6] I. Wilson and B. Gick, "Ultrasound Technology and Second Language Acquisition Research," *Gasla*, no. Gasla, pp. 148–152, 2006.

[7] J. Abel *et al.*, "ULTRASOUND-ENHANCED MULTIMODAL APPROACHES TO PRONUNCIATION TEACHING AND LEARNING."

[8] M. Bernhardt *et al.*, "Ultrasound as visual feedback in speech habilitation: Exploring consultative use in rural British Columbia, Canada," *Clin. Linguist. Phonetics*, vol. 22, no. 2, pp. 149–162, 2008.

[9] J. L. Preston, P. McCabe, A. Rivera-Campos, J. L. Whittle, E. Landry, and E. Maas, "Ultrasound Visual Feedback Treatment and Practice Variability for Residual Speech Sound Errors," *J. Speech Lang. Hear. Res.*, vol. 57, no. 6, p. 2102, 2014.

[10] J. Mora, W. Lee, and G. Comeau, "3D Visual Feedback in Learning of Piano Posture," *Proc. 2nd Int. Conf. Technol. e-learning Digit. Entertain.*, no. 9, pp. 763–771, 2007.

[11] J. Mora, W. S. Lee, G. Comeau, S. Shirmohammadi, and A. El Saddik, "Assisted piano pedagogy through 3D visualization of piano playing," in *Proceedings of the 2006 IEEE International Workshop on Haptic Audio Visual Environments and Their Applications, HAVE 2006*, 2007, pp. 157–160.

[12] K. Xu, T. Gábor Csapó, P. Roussel, and B. Denby, "A comparative study on the contour tracking algorithms in ultrasound tongue images with automatic re-initialization," *J. Acoust. Soc. Am.*, vol. 139, no. 5, pp. EL154-EL160, 2016.

[13] O. Rastadmehr, T. Bressmann, R. Smyth, and J. C. Irish, "Increased midsagittal tongue velocity as indication of articulatory compensation in patients with lateral partial glossectomies," *Head Neck*, vol. 30, no. 6, pp. 718–726, 2008.

[14] M. H. Mozaffari and W.-S. Lee, "Freehand 3-D Ultrasound Imaging: A Systematic Review," *Ultrasound Med. Biol.*, vol. 43, no. 10, pp. 2099–2124, 2017.

[15] C. Laporte and L. Ménard, "Robust tongue tracking in ultrasound images: a multi-hypothesis approach," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[16] Y. S. Akgul, C. Kambhamettu, and M. Stone, "Automatic extraction and tracking of the tongue contours," *IEEE Trans. Med. Imaging*, vol. 18, no. 10, pp. 1035–1045, 1999.

[17] L. Tang, G. Hamarneh, and T. Bressmann, "A Machine Learning Approach to Tongue Motion Analysis in 2D Ultrasound Image Sequences," in *Machine Learning in Medical Imaging Proceedings*, 2011, pp. 151–158.

[18] L. Tang, T. Bressmann, and G. Hamarneh, "Tongue contour tracking in dynamic ultrasound via higher-order MRFs and efficient fusion moves," *Med. Image Anal.*, vol. 16, no. 8, pp. 1503–1520, 2012.

[19] D. Fabre, T. Hueber, F. Bocquelet, and P. Badin, "Tongue tracking in ultrasound images using eigentongue decomposition and artificial neural networks," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[20] M. Stone, "A guide to analysing tongue motion from ultrasound images," *Clin. Linguist. Phonetics*, 2005.

[21] G. Bradski, "The OpenCV Library," *Dr Dobbs J. Softw. Tools*, 2000.

[22] I. Fasel and J. Berry, "Deep belief networks for real-time extraction of tongue contours from ultrasound during speech," *Proc. - Int. Conf. Pattern Recognit.*, pp. 1493–1496, 2010.

[23] J. Berry and I. Fasel, "Dynamics of tongue gestures extracted automatically from ultrasound," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 2011, pp. 557–560.

[24] J. J. Berry, *Machine learning methods for articulatory data*. The University of Arizona, 2012.

[25] Y. Ji, L. Liu, H. Wang, Z. Liu, Z. Niu, and B. Denby, "Updating the silent speech challenge benchmark with deep learning," *arXiv Prepr. arXiv1709.06818*, 2017.

[26] K. Xu, P. Roussel, T. G. Csapó, and B. Denby, "Convolutional neural network-based automatic classification of midsagittal tongue gestural targets using B-mode ultrasound images," *J. Acoust. Soc. Am.*, vol. 141, no. 6, pp. EL531--EL537, 2017.

[27] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[28] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9351, 2015, pp. 234–241.

[29] K. Saini, M. Rohit, and M.L.Dewal, "Ultrasound Imaging and Image Segmentation in the area of Ultrasound: A Review," *Int. J. Adv. Sci. Technol.*, 2010.

[30] W. N. McDicken and T. Anderson, *Basic physics of medical ultrasound*, Thrid Edit., vol. 1. Elsevier Ltd, 2011.

[31] K. Martin, *Basic equipment, components and image production*, Thrid Edition., vol. 1. Elsevier Ltd, 2011.

[32] S. L. Bridal, J. M. Correas, A. Saïed, and P. Laugier, "Milestones on the road to higher resolution, quantitative, and functional ultrasonic imaging," in *Proceedings of the IEEE*, 2003.

[33] A. Jaumard-Hakoun, K. Xu, P. Roussel-ragot, and M. L. Stone, "Tongue Contour Extraction From Ultrasound Images," *Proc. 18th Int. Congr. Phonetic Sci. (ICPhS 2015)*, 2015.

[34] J. a. Noble and D. Boukerroui, "Ultrasound image segmentation: a survey," *IEEE Trans. Med. Imaging*, vol. 25, no. 8, pp. 987–1010, 2006.

[35] R. F. Wagner, S. W. Smith, J. M. Sandrik, and H. Lopez, "Statistics of Speckle in Ultrasound B-Scans," *IEEE Trans. Sonics Ultrason.*, 1983.

[36] C. B. Burckhardt, "Speckle in Ultrasound B-Mode Scans," *IEEE Trans. Sonics Ultrason.*, 1978.

[37] M. F. Insana and R. F. Wagner, "Analysis of ultrasound image texture via generalized Rician statistics," *Opt. Eng.*, 1986.

[38] E. Jakeman, "On the statistics of K-distributed noise," *J. Phys. A. Math. Gen.*, 1980.

[39] P. Mohana Shankar, "A general statistical model for ultrasonic backscattering from tissues," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 2000.

[40] M. Mignotte and J. Meunier, "A multiscale optimization approach for the dynamic contour-based boundary detection issue," *Computerized Medical Imaging and Graphics*. 2001.

[41] M. Mignotte, J. Meunier, and J.-C. Tardif, "Endocardial Boundary Estimation and Tracking in Echocardiographic Images using Deformable Template and Markov Random Fields," *Pattern Anal. Appl.*, 2001.

[42] M.-H. Roy Cardinal, J. Meunier, G. Soulez, É. Thérasse, and G. Cloutier, "Intravascular Ultrasound Image Segmentation: A Fast-Marching Method," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003*, 2003, pp. 432–439.

[43] a Zahalka and a Fenster, "An automated segmentation method for three-dimensional carotid ultrasound images.," *Phys. Med. Biol.*, 2001.

[44] D. Boukerroui, A. Baskurt, J. A. Noble, and O. Basset, "Segmentation of ultrasound images-multiresolution 2D and 3D algorithm based on global and local statistics," *Pattern Recognit. Lett.*, 2003.

[45] N. Paragios, M. P. Jolly, M. Taron, and R. Ramaraj, "Active shape models and segmentation of the left ventricle in echocardiography," *Scale Sp. PDE Methods Comput. Vis.*, 2005.

[46] Z. Tao, C. C. Jaffe, and H. D. Tagare, "Tunnelling descent: a new algorithm for active contour segmentation of ultrasound images.," *Inf. Process. Med. Imaging*, 2003.

[47] M. Martín-Fernández and C. Alberola-López, "An approach for contour detection of human kidneys from ultrasound images using Markov random fields and active contours," *Med.*

Image Anal., 2005.

[48] B. Potočnik and D. Zazula, "Automated analysis of a sequence of ovarian ultrasound images. Part I: Segmentation of single 2D images," *Image Vis. Comput.*, vol. 20, no. 3, pp. 217–225, 2002.

[49] A. Mishra, P. K. Dutta, and M. K. Ghosh, "A GA based approach for boundary detection of left ventricle with echocardiographic image sequences," *Image Vis. Comput.*, 2003.

[50] M. Mulet Parada, "Intensity independent feature extraction and tracking in echocardiographic sequences.," University of Oxford, 2000.

[51] G. Castellano, L. Bonilha, L. M. Li, and F. Cendes, "Texture analysis of medical images," *Clin. Radiol.*, 2004.

[52] R. Rhys, *Ultrasound of the neck*, Thrid Edition., vol. 2. Elsevier Ltd, 2011.

[53] D. M. Reeve, "Diagnostic Ultrasound: Physics and Equipment," *J. Nucl. Med.*, vol. 53, no. 6, pp. 1004–1004, 2012.

[54] K. Xu, Y. Yang, C. Leboullenger, P. Roussel, and B. Denby, "Contour-based 3D tongue motion visualization using ultrasound image sequences," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2016.

[55] A. Boezaart and B. Ihnatsenka, "Ultrasound: Basic understanding and learning the language," *Int. J. Shoulder Surg.*, vol. 4, no. 3, p. 55, 2010.

[56] C. a. Chapelle, "Technology and Second Language Acquisition," *Annu. Rev. Appl. Linguist.*, 2008.

[57] B. Bernhardt, B. Gick, P. Bacsfalvi, and M. Adler-Bock, "Ultrasound in speech therapy with adolescents and adults," *Clin Linguist Phon*, vol. 19, no. 6–7, pp. 605–617, 2005.

[58] D. M. Reeve, "Diagnostic Ultrasound: Physics and Equipment," *J. Nucl. Med.*, 2012.

[59] M. Li, C. Kambhamettu, and M. Stone, "Automatic contour tracking in ultrasound images," *Clin. Linguist. Phon.*, vol. 19, no. 6–7, pp. 545–554, Jan. 2005.

[60] A. Jaumard-Hakoun, K. Xu, P. Roussel-Ragot, G. Dreyfus, and B. Denby, "Tongue contour extraction from ultrasound images based on deep neural network," *Proc. 18th Int. Congr. Phonetic Sci. (ICPhS 2015)*, May 2016.

[61] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, 1988.

[62] "Commons.wikimedia.org. (2005). File:Snake-contour-example.jpg - Wikimedia Commons. [online] Available at: https://commons.wikimedia.org/wiki/File:Snake-contour-example.jpg [Accessed 20 Oct. 2018]."

[63] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active Appearance Models," *Proc. Eur. Conf. Comput. Vis.*, 1998.

[64] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active Shape Models-Their Training and Application," *Comput. Vis. Image Underst.*, 1995.

[65] A. Roussos, A. Katsamanis, and P. Maragos, "Tongue tracking in ultrasound images with

active appearance models," in *Proceedings - International Conference on Image Processing, ICIP*, 2009.

[66] G. Hamarneh and T. Gustavsson, "Combining snakes and active shape models for segmenting the human left ventricle in echocardiographic images," *Comput. Cardiol. 2000. Vol.27 (Cat. 00CH37163)*, 2000.

[67] M. Loosvelt, P.-F. Villard, and M.-O. Berger, "Using a biomechanical model for tongue tracking in ultrasound images," in *International Symposium on Biomedical Simulation*, 2014, pp. 67–75.

[68] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation ppt," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 8828, no. c, pp. 3431–3440, 2015.

[69] O. Matan, J. C. Burges, Y. LeCun, and J. S. Denker, "Multi-digit recognition using a space displacement neural network," *Proc. NIPS*, 1992.

[70] P. Moeskops, M. A. Viergever, A. M. Mendrik, L. S. De Vries, M. J. N. L. Benders, and I. Isgum, "Automatic Segmentation of MR Brain Images with a Convolutional Neural Network," *IEEE Trans. Med. Imaging*, 2016.

[71] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, 2012.

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

[73] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10008 LNCS, pp. 179–187, 2016.

[74] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-net: Learning dense volumetric segmentation from sparse annotation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.

[75] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," *arXiv Prepr. arXiv1606.04797*, 2016.

[76] J. Berry, I. Fasel, L. Fadiga, and D. Archangeli, "Training Deep Nets with Imbalanced and Unlabeled Data," *Proc. Interspeech*, no. i, pp. 1756–1759, 2012.

[77] D. Fabre, T. Hueber, F. Bocquelet, and P. Badin, "Tongue tracking in ultrasound images using eigentongue decomposition and artificial neural networks," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2015–January, no. 2, pp. 2410–2414, 2015.

[78] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Comput.*, 2006.

[79] C. M. Chen, H. H. S. Lu, and Y. C. Lin, "An early vision-based snake model for ultrasound image segmentation," *Ultrasound Med. Biol.*, vol. 26, no. 2, pp. 273–285, 2000.

[80] T. G. Csapo and S. M. Lulich, "Error analysis of extracted tongue contours from 2D ultrasound images," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol.

2015–January, pp. 2157–2161, 2015.

[81] B. M. O. Loosvelt M, Villard P F, “Using a biomechanical model for tongue tracking in ultrasound images,” *Int. Symp. Biomed. Simulation. Springer, Cham*, pp. 67–75, 2014.

[82] S. Ghrenassia, L. Ménard, and C. Laporte, “Interactive segmentation of tongue contours in ultrasound video sequences using quality maps,” vol. 903440, no. March 2014, p. 903440, 2014.

[83] Y. LeCun and Y. Bengio, “Convolutional networks for images, speech, and time series,” *Handb. brain theory neural networks*, 1995.

[84] F.-F. Li, “Stanford university computer science class cs231n: Convolutional neural networks for visual recognition.” 2017.

[85] J. Platt, “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods,” *Adv. large margin Classif.*, 1999.

[86] X. Glorot, A. Bordes, and Y. Bengio, “ReLU,” *AISTATS ’11 Proc. 14th Int. Conf. Artif. Intell. Stat.*, 2011.

[87] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res.*, 2014.

[88] Y. LeCun *et al.*, “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Comput.*, 1989.

[89] L. Bottou, “Large-Scale Machine Learning with Stochastic Gradient Descent,” *Proc. COMPSTAT’2010*, 2010.

[90] M. Li, T. Zhang, Y. Chen, and A. J. Smola, “Efficient mini-batch training for stochastic optimization,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’14*, 2014.

[91] G. Litjens *et al.*, “A survey on deep learning in medical image analysis,” *Med. Image Anal.*, vol. 42, no. 1995, pp. 60–88, Dec. 2017.

[92] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

[93] R. M. Haralick, S. R. Sternberg, and X. Zhuang, “Image Analysis Using Mathematical Morphology,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 1987.

[94] H. Talbot and L. M. Vincent, “Euclidean skeletons and conditional bisectors,” *Proc SPIE*, 1992.

[95] C. Lantuéjoul, “Sur le modèle de Johnson-Mehl généralisé,” 1977.

[96] J. Serra, “Mathematical Morphology, Vol. I,” *London, UK Acad.*, 1982.

[97] G. Zaccone, M. R. Karim, and A. Menshawy, *Deep Learning with TensorFlow*. 2017.

[98] TensorFlow, “TensorBoard: Visualizing Learning,” *TensorFlow*, 2017. .

[99] Theano Development Team, “Theano: A Python framework for fast computation of mathematical expressions,” *arXiv e-prints*, 2016.

[100] N. Ketkar, “Introduction to PyTorch,” in *Deep Learning with Python*, 2017.

[101] F. Chollet, “Keras: Deep Learning library for Theano and TensorFlow,” *GitHub Repos.*, 2015.

[102] M. Abadi *et al.*, “TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’16)*, 2016.

[103] D. P. Kingma and J. L. Ba, “Adam: a Method for Stochastic Optimization,” *Int. Conf. Learn. Represent. 2015*, 2015.

[104] M. D. Bloice, C. Stocker, and A. Holzinger, “Augmentor: An Image Augmentation Library for Machine Learning,” pp. 1–5, 2017.

[105] X. Glorot, A. Bordes, and Y. Bengio, “Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach,” *Proc. 28th Int. Conf. Mach. Learn.*, 2011.

[106] L. R. Dice, “Measures of the Amount of Ecologic Association Between Species,” *Ecology*, 1945.

[107] E. Lawson *et al.*, “Seeing Speech: an articulatory web resource for the study of phonetics [website],” 2015.

[108] R. C. Gonzalez, R. E. Woods, L. Mcdowell, T. Galligan, and P. P. Hall, *Digital Image Processing*. .

[109] N. van Noord and E. Postma, “Learning scale-variant and scale-invariant features for deep image classification,” *Pattern Recognit.*, 2017.

[110] V. Chalana and Y. Kim, “A methodology for evaluation of boundary detection algorithms on medical images.,” *IEEE Trans. Med. Imaging*, 1997.