Automatic Left Ventricle Segmentation in 2D Echocardiography

Ezequiel Ortiz

2018

Project Plan

Problem

Ultrasound is a widely utilised imaging modality due to its portability and speed of acquisition. An important metric of heart health is left ventricle ejection fraction (LVEF), which quantifies the difference in left ventricle volume between diastole and systole. Currently this metric is calculated most often using manual delineations on ultrasound images. However, this requires an expert to spend approximately 30-45 minutes calculating an approximation of LVEF. Automatic segmentation of the left ventricle would eliminate the need for a trained professional, therefore reducing inter-observer variability, whilst also making image acquisition more time and cost effective.

State of the Art

Deep learning algorithms today are able to segment the left ventricle from an ultrasound image. However, most all of them focus on either a 2 or 4 chamber image. Our algorithm accepts and actually requires both 2 and 4 chamber images, allowing it to better approximate the complex shape of the left ventricle. Current algorithms are also limited in that they typically only segment the blood pool and not the myocardium of the left ventricle. The myocardium segmentation will be imperative in calculating the Left Ventricle Mass (LVM), another key health indicator that is currently not optimized or automated.

Aims

Our algorithm should take two ultrasound videos of the heart as input. One will be a 2 chamber view of the heart cycle and the other a 4 chamber view. The algorithm should then identify the frame of diastole and systole within that cycle for each video. The neural network will segment these four images to then calculate LVEF and LVM.

Work Plan

1. Manual segmentation of images to form a training set

2. Train two networks to segment the blood pool from 2 and 4 chamber images respectively.

3. Train network that will be able to segment both 2 and 4CH ultrasound images

4. Train network to segment the myocardium of the left ventricle in both 2 and 4 chamber images.

5. Automatic calculation of left ventricle ejection fraction and left ventricular mass on training data

6. Automatic calculation of left ventricle ejection fraction and left ventricular mass in a live setting

Review of project plan

Our project differed from the initial work plan in several ways. While the work plan was focused on a deliverable program that would generate cardiac metrics from unprocessed echocardiogram images, the project ended up exploring the effectiveness of deep learning methods on segmenting the left ventricle blood pool and myocardium. We underestimated the complexity of training and managing programming projects of this size. While we are disappointed that we never got to generate any left ventricular ejection fraction or left ventricular mass metrics, we are more than satisfied with our results as we showed the validity of deep learning methods on 2D cardiac ultrasound images.

Contents

1 Abstract

This report outlines our investigation into the performance of deep learning methods on semantic segmentation of the left ventricle blood pool and myocardium from 2D echocardiogram images. We found that a convolutional neural network configured in a unet architecture was effective at segmenting the blood pool but further work is required to effectively segment the myocardium.

2 Acknowledgements

I would like to thank Esther and Andy for guiding me through this project. I did every step of this for the first time and learned more than I ever have. Thank you for your expertise and time. Thank you Bram for helping me understand echocardiograms and the potential space for algorithms such as mine in the NHS. I would also like to thank Nick, Irina, James, and Eric for helping me with deep learning python and bash and most of all for making me feel welcome in Beckett House. Thank you Mom and Dad for supporting me though this project and degree. Thank you Lauren for being at my side I can't picture any of this without you.

3 Intro

While 2D echocardiograms are useful, the dialogue with patients to determine how they view their physical abilities is just as valuable and useful for deciding whether more scans should be taken. Ample information other than quantitative measures of LVEF and EF is required for the cardiologist to form a holistic judgement of the patients well being. For example, whilst a patient may have lower than normal LVEF and reduced heart function, they may present as asymptomatic and this would inform the cardiologists judgement on the appropriate course of treatment. LVEF is well suited to automation because the metric is time consuming and variable compared to automatic methods and the metric is useful but almost never the sole reason to make a diagnosis or treatment description - it is an indicator. While cardiologists are experts at diagnosing what is wrong with a patient's heart, they synthesise an array of information that is comprised of much more than just LVEF and LVM. Many are able to make their diagnoses simply by looking at the echoes and making a qualitative judgement. When qualitative measurements are made

Motivation

Cardiovascular disease is responsible for 17.9 million deaths, or 44% of all noncommunicable deaths per anum [?]. Detection and monitoring of heart disease is important treatment and management of symptoms. Quantification of the interior volume of the left ventricle and thickness of muscle is extremely important due to how illustrative these measurements are to the overall function of the ventricle. [?]

The most prevalent method to assess these metrics is cardiac imaging. While there are many imaging modalities, ultrasound has been cemented as a cornerstone of cardiac imaging due to the non-invasive procedure, low cost and real-time image acquisition. However, interpretation of ultrasound images is time intensive and can be challenging even for trained personnel. The automatic segmentation of the left ventricle blood pool and myocardium would reduce the need for expert evaluation and inter-operator variance. Access to more accurate clinical data with reduced wait times would improve patient assessments and outcomes for cardiovascular disease and reduce monetary strain on hospitals.

Metrics

Quantification of the performance of the heart is crucial for diagnosis of cardiac pathology. Due to the complexity of the heart and cardiovascular system as a whole, the amount that can go wrong is staggering. Ensuring that the metrics of heart functionality collected are clinically relevant, reproducible and easy to interpret lead to quick and accurate diagnosis and treatment.

The two most common metrics collected during echocardiograms are left ventricular ejection fraction (LVEF) and left ventricular mass (LVM). [?] These quantifications in conjunction with the visual interpretation of the echocardiogram are potent in assessing overall cardiovascular health.

Left Ventricular Ejection Fraction

LVEF is the combination of ventricle volume measurements that are made at different points within the cardiac cycle. LVEF quantifies the effectiveness of the heart as it is the ratio of end diastolic volume (EDV) to end systolic volume (ESV), indicating how much blood the heart is pumping per beat.

EF is calculated according to EQ 2.

$$SV = EDV - ESV \tag{1}$$

$$EF = \frac{SV}{EDV} * 100 \tag{2}$$

Determining what is normal LVEF is not trivial as it is very patient specific. What is normal for one patient might be unhealthy for another. LVEF can also change depending on the physical state of the patient when the measurement is made i.e.the patient's level of cardiovascular stress and whether they are supine or seated. M. E. Pfisterer *et al.,* set out to determine ranges of normal LVEF by assessing the mean LVEF within a population of 1200 patients using ultrasound. The results from this study estimated the lower limit of normal LVEF values for men to be $51.1 \pm 4.2\%$ with the upper limit of normal being $76.6 \pm 3.8\%$ [**?**]. These values were assessed with echocardiogram while the patient was supine and at rest. They also assessed LVEF with the more accurate but invasive x-ray angiography; no signifiant differences were found. LVEF was for the most part independent of sex or age group, with the exception of elevated LVEF levels in patients over 60 years old and athletes. Athletes can often test for LVEF below normal even though their hearts are healthy and functional. These findings show the difficulty in determining what is normal or abnormal LVEF.

The American Society of Echocardiography has published general guidelines for LVEF classification. Reduced LVEF should be classified as equal to or below 40% with borderline being 41%-49% and normal function occupying the 50%-70% range. [**?**]

LVEF is an excellent indicator for change in cardiac function because it is repeatable and can be used to measure change over time. Changes in LVEF can indicate common heart ailments like hypertrophic or dilated cardiomyopathy.

However, LVEF does not correlate directly to cardiac function. LVEF can be within normal values while the function of the heart is impaired. For example, heart failure preserved ejection fraction (HFpEF) is a condition where the heart is failing to supply the body with enough blood in the presence of normal LVEF. This is because LVEF only depends on EDV and ESV and does not take into account the time duration or the geometry of the contraction.

Left Ventricular Mass

LVM is a measure of how thick the myocardium of the left ventricle is. It is a strong predictor for cardiovascular events [?], as it is an excellent indicator of the health of the myocardium.

LVM is calculated by segmenting the myocardium from the 2D echocardiogram image. From this segmentation, the LV length and myocardial thickness must be measured to then calculate the myocardial volume. This volume is then multiplied by 1.05 g/ml to convert to mass. [?] LVM should be measured at the end of diastole, when the left ventricle is relaxed and full of blood. To approximate the volume with 2D echocardiogram methods the Area Length Method [?] could be used. This and other 2D left myocardial volume methods assume that the thickness of the wall is relatively uniform, so hearts with regional thickness changes like those afflicted with DCM can lead to inaccurate measurements.

Dilated cardiomyopathy (DCM) is indicated by lower than normal LVM and occurs when the left ventricle myocardium thins and stretches, which compromises the strength and function of the left ventricle.

In conjunction, LVEF and LVM can help inform physicians on the efficiency of the heart and the reasons behind that level of function. There are many methods both invasive and non-invasive that can be used to measure LVEF. Non-invasive measurement techniques are always preferred if they offer similar performance to invasive methods.

Cardiac Imaging

The most common imaging modalities used to image the heart are magnetic resonance imaging (MRI), x-ray computed tomography (CT), and ultrasound (US).

MRI employs strong magnetic fields and quantum mechanical properties of particles to get signal from the presence of hydrogen ions i.e. protons. The behaviour of protons within the context of an MRI scan is highly dependent on the structure and magnetic properties of the tissue surrounding each proton.

MRI is usually non-invasive but can require the use of contrast agents. Contrast agents are usually used in cardiac MRI to detect scar tissue in the myocardium. Scar tissue can form when the blood supply to regions of the heart becomes restricted or halted. As scar tissue is not electrically active like healthy myocardium, its presence will impair the hearts ability to pump effectively as well as increase the likelihood of electrical pathologies like ventricular tachycardia or fibulation. As infarcted tissue by definition has limited or absent blood supply, the perfusion of contrast agents through the myocardium will highlight damaged regions.

MRI offers contrast between different biological tissues. Once the heart has been scanned, it is relatively easy to calculate the interior volume of the left ventricle. For example, region growing methods are usually used to directly calculate the volume from the 3D scan. This approach is very accurate as no assumptions are made about the shape of the ventricle. MRI is the gold standard for measuring LVEF.

However, scanners are usually in short supply, require skilled operators, are expensive to run and maintain, and are slow to acquire images. Long image acquisition time can lead to motion artefacts if the subject is moving and patients must hold their breath while being imaged to avoid introducing such artefacts. Patients that are unable to hold their breath may be ill-suited for MRI. This long image acquisition time also prevents the cardiac cycle from being captured in real time. The cardiac MRI images are captured at equivalent points within the cardiac cycle using the ECG to time each capture point. As cardiac MRI image acquisition gets faster this will be less of a problem, but for now the cardiac MRI images are an average of the heart over many cycles.

Metrics gathered from cardiac MRI are generally easier to calculate and more accurate but eliminate beat to beat variation. Additionally, due to the extremely strong magnetic fields present within the scanner, those with electrical or ferrous metal implants will not be able to be scanned unless they are MRI conditional. However, due to the complex restrictions placed on scanning these devices, the increased expense of MRI condtional technology and the relative rarity of such devices, most clinicians elect to use alernative imaging modalities. [**?**]

CT imaging involves taking many 2D x-ray images from different angles then using computed tomography algorithms to generate 3D images. CT imaging is not generally used for cardiac imaging as soft tissue does not provide sufficient contrast. The high energy x-rays used in CT are not heavily attenuated by low z elements found in soft tissue. Functional imaging of the heart with CT must utilize contrast agents [?]. High z compounds are injected into the bloodstream and subsequently imaged to give images of the blood pool within the LV.

CT imaging has excellent spacial and temporal resolution. Fast image acquisition means minimal motion artefacts when compared to MRI. However, the radiation dose and use of contrast agents make cardiac CT rare for LVEF measurement.

Ultrasound

Ultrasound as used for echocardiograms is the core of cardiac imaging. High frequency sound waves are sent into the body, reflected, refracted and scattered, then picked up by the same transducer that produced the original sound pulse. These sound waves are reflected most strongly across boundaries of differing acoustic impedance. In cardiac ultrasound this is illustrated by a distinct boundary between the myocardium and the blood pool.

However, imaging the heart with ultrasound does present some problems. The ribcage encases the heart and a transducer that can fit in-between the ribs must be used as bone is effectively opaque to the sound pulses. When the heart is imaged across the ribcage the scan is called a trans thoracic echocardiogram (TTE). Even with correct positioning there can still be some shadowing from the ribs present in some images. Imaging around the ribs can be circumvented with transoesophageal echocardiograms (TEE). This type of scan involves placing the ultrasound probe down the patients oesophagus and imaging the heart from within the ribcage. While TEE does provide higher quality images, it is usually reserved for acquiring detailed information about the atria or valves of the heart as it is an invasive procedure.

Pixel intensity within ultrasound images does not directly relate to properties of imaged tissue like the Labert-Beer law for CT. So there will not be a characteristic intensity associated with a certain tissue that would make segmentation easy. Dropout occurs when the border between two different tissues is parallel to the orientation of the scanner. This is similar to attempting to see a reflection in a mirror that is parallel to your line of sight.

Ultrasound is the most widely used and prevalent modality to assess the structure and function of the heart. Ultrasound is robust as different scanner modes allow for 2D and 3D image and video acquisition. Doppler imaging allows for blood flow analysis and the use of contrast agents can provide increased levels of detail. The relatively small size of ultrasound scanners when compared to MRI and CT make ultrasound machines much more versatile and manoeuvrable. Unlike the other modalities where the patient is physically placed within the scanner, the ultrasound probe is positioned around the patient. In most echocardiograms, both a 2CH view and 4CH view is taken. As these image views are approximately orthogonal, information from both can be combined for a better understanding of the hearts 3 dimensional shape. IMAGES 2D 3D 2CH 4CH

Echocardiogram

Echocardiograms are integral to cardiac evaluation. They are often the first type of cardiac scan that patients undergo. The procedure involves an electrocardiogram so that the diastole and systole can be easily found. Qualitative approach How the patient feels

Manual Segmentation

Once the echocardiogram has been completed, the images must be analysed by a trained professional in order to calculate meaningful metrics. Both LVM and LVEF require segmentation, where some regions of the image are identified and marked as part of the myocardium or blood pool. In the case of LVEF, in order to minimise the assumptions being made about the shape of the ventricle, both the 2CH and 4CH views are segmented.

The American Society of Echocardiography and European Association of Cardiovascular imaging have published a set of guidelines for LV chamber quantification. As most ultrasound machines that are used for echocardiograms come with software for segmenting blood pools, these steps should be considered within that context:

1. Frames of diastole and systole must be identified. This will facilitated by the use of the single lead ECG that is present on the images.

2. Then a line will be drawn from the mitrial valve annulus, a fibrous ring at the base of the valve that shows up bright in most echocardiograms,

along the endocardial border and to the other side of the annulus.

3. The beginning and end of the line must be connected with a straight line that ignores the position of the valve.

4. From the straight line across the valve, the longest orthogonal line is then drawn that connects the valve line to the epicardium at the apex of the heart.

5. Now the bi-plane method of disk summation, or Simpson's Bi-plane, can be used to calculate the LV volume.

EDIT The QRS complex indicate the beginning of ventricular depolarisation or systole, while the T wave indicates the beginning of repolarization or diastole.

Trabecualtions are regions on the epicardium where blood becomes trapped and calcified within the myocadium. [?] They can look like myocardium within echocardiograms and should not be counted as such when determining LV volume. EDIT

Eq. 3 shows how to calculate the volume of a single oval shaped puck with length d1, width d2, and thickness h [?].

$$V = pi(d1/2) * (d2/2) * h \tag{3}$$

The continuous LV interior is divided up into n disks with volumes calculated according to Eq. 3.

This method is widely used but suffers from endocardial dropout, user variation, and incompatiability with some deformed ventricles. [?]

Automatic Segmentation

While most vital signs can be measured automatically with machines, we find that it is best practice to take blood pressure and heart rate by hand because of the patient contact [?]. Humans will also pick up on variation within the heart rate that could be more insightful than the digital readout of automatic systems. Furthermore, it is beneficial in some circumstances to have patient contact both for confirmation of symptoms and to ensure that patients do not feel as though their care has been relegated to machines. However, manual segmention is prone to interoperator variance [?] and time spent manually segmenting images is time spent away from patients, therefore an automatic procedure would be more beneficial.

Historically, finding the endocardial border automatically relied statistical shape and active contour models. Jacob *et al.* [?] and Bosch *et al.* [?] both propose border detection methods that utilise statistical shape models or active appearence models (AAMs). These models are trained on echocardiogram videos and learn how the hearts move through time. PCA analysis is used to calculate the principal means of LV motion. This model encompasses the main modes of motion that LVs undergo during a cardiac cycle within several parameters. These parameters were then iterated over to fit the AAM to a test echocardiogram.

Chalana *et al.* [**?**] propose an active contour model. These models try to minimize the energy of a given set of points or contour. For endocardial border detection, there is an energy defined within the blood pool that causes the contour to be smooth and continuous and an external energy that pulls the contour to the endocardial border. Gradient descent minimized the energy of the system and aligned the contour. However, this contour required a competent manual initialisation in order to converge.

Classically, neural networks had been used to classify images. When an image is fed into the network, the network assigns a label to classify the contents of the image. This task requires a transformation of the spacial information of the image into feature information, or information about what the image contains. Automatic semantic segmentation is the logical next step, where we not only want to know what is in an image, but what pixels contain our object.

Deep Learning

Neural networks are computational structures that mimic the function of biological neurons. Networks with sufficient complexity can model any mathematical function. The functional unit of neural networks is the neuron. Individual neurons, like the whole, will each receive an input and provide an output based on an internal activation function. Activation functions can vary in sophistication from simple step functions that provide a binary output based on a hard coded threshold, to sigmoid functions that organically map the input to a range of values between zero and one. The selection of activation functions will depend on the desired behaviour of the network.

One of the simplest configurations of neurons into a network is the perception. We will examine a generic multi-layer perceptron (mlp) classifier to investigate how neural networks make decisions. The structure of most networks consist of layers. In mlps, there is an input layer, hidden layers, and the output layer. The input layer accepts whatever information is to be classified. The hidden layers contain the neurons that make the classification decision. The output layer will have a neuron for every class within the dataset. The output layer will also be a softmax layer. The sum of all neuron values in a softmax layer must sum to one. An example of two class classification output could be one output neuron is valued at .75 and the other .25. In this case the network is 75% sure that the input belongs to the first class. For a given input each pixel value of our image will get sent to every neuron in the first of the hidden layers. These layers map the image input to a decision of what digit the network thinks it is seeing. If every neuron within the first hidden layer is getting input from every input layer neuron, then the output of every hidden layer neuron will be the same. In order to prevent this, we need to introduce the idea of weights.

Weights signify the strength of connections between neurons. If a neuron has been strongly associated with a three, in our digit classification example, then there will be neurons that fire strongly when the image provided is a three. The weights in any neural network determine the flow of information through the network and the final decision. In this classification example, each digit from zero to nine has different characteristics about it that make it easily recognisable to our brain as well as a trained neural network. A fully trained network is structurally identical to its untrained counterpart. It is the weights that dictate the strength of the connections and by extension the pathways responsible for the pseudo cognition.

The difference between the mlp and a deep network is simply the number of layers. The mlp by convention is limited to one hidden layer and any network with more than one hidden layer is considered deep. Deep networks have seen much more use than mlps recently as computer hardware advancements have allowed the training and implementation of networks with many more than two hidden layers. While the mlp simply took the pixel intensity levels as input and subsequently made decisions based purely on pixel intensity, more and more deep networks are utilising convolutions to extract relevant features from the input images.

Convolutional Neural Networks

Convolution involves generating a new image from an input image. Each pixel of this new image is the result of an element-wise multiplication of a region in the original image and a filter. For example, a 3 by 3 averaging filter averages the intensities of a pixel and its eight neighbours to give the intensity of the output pixel. This convolution is carried out repeatedly as the filter is swept across the input image.

Convolution naturally reduces the resolution of the output due to the thickness of the filter. Padding the input image prevents this loss and is as simple as adding pixels to the edges of the image to increase the resolution.

There are many types of mask that can be used in convolutions and they produce a wide variety of output images that highlight different features of the original image. Sobel and Laplacian filters make edges bright and flat surfaces dark. Averaging filters can mitigate noisy images and produce blur. While these filters are not explicitly programmed into CNNs, networks can learn their own filters that do the same tasks.

The size of mask can also be adjusted. We used an example of a 3x3 mask but larger masks such as 5x5 are also common. Masks are usually square with an odd width and height to ensure a middle pixel. All of the convolutions we have talked about are predetermined. A known mask is applied in order to produce a specific output. In deep convolutional neural networks (CNN), the masks are analogous to the mlp weights in that they will be defined through training. The nature of optimizing the network to complete segmentations means that the filters within a CNN will be optimized to better select and highlight the features within the image that are relevant to the segmentation task at hand. For example, the network trained to segment cars from an image of a street will have filters that help it identify the circular shape of the wheels or smooth paint texture.

Unet

The Unet architecture is the standard approach for segmenting both 2D and 3D biomedical imaging data. It is strongly reliant on image augmentation to increase the size and variance of comparatively small biomedical image datasets. Augmentation involves the application of rigid or non-rigid transforms to an image to produce another altered image. The signature architecture is comprised of a downsampling pathway followed by upsampling to produce a mask that is the same dimension as the input image. As the input image is convolved and downsampled, spacial information is converted into feature information. What constitutes as feature information would not make much sense anywhere but within the workings of the neural network. The information that is distilled as the image progresses through the downsampling phase is directly related to the overall purpose of the network. The feature information in our case will be vital to the network to generate predicted masks for a given image.

The subsequent expansion after the initial distillation of information gives the unet its ”U” shape.

During the expansion phase, images from the contracting end are combined with the upsampled image. The network is ”reminded” what the original image looks like with these connections.

The downsampling of the input image also allows us to use one filter size. Without downsampling we may need to use many filters of different sizes to capture differently sized features.

The structure of the unet from the original paper can be see in figure x.

4 Method

Programming Environment

During the first stages of the project, we worked within the Google Colab environment. Google colab is an online coding suite where python code can be run from a web browser. We chose colab because of the free cloud GPU that would be needed to train the network. Colab mimics jupyter notebooks functionality in that code can be split up and run in sections even within the same script. We were also unconstrained by a physical computer and code could be written and ran from any web browser given access to the project. However there were some downsides to colab that prompted the eventual switch. All of the data for training and validation had to be stored in the google drive of the account that owns the project. Any changes that were made to the dataset had to be uploaded to the drive, and we had to make sure that the dataset was the most recent iteration.

The performance of the cloud GPU was not as fast as a local GPU. Some longer training regimes were halted on account of google colab becoming idle and ending our processes. This was more of a issue before we implemented model checkpoints that allowed us to save model progress as it was training. Google colab was an excellent starting point for the project as it removed lots of start up cost for a deep learning project such as this, but the downsides outweighed the benefits in the end.

The move to running our code and storing our data on a local machine came once we started getting promising results with the blood pool segmentation. As our project progressed, the need for more intense training prompted a move. We moved our project to a pc containing a Nvidia GTX 1060 GPU and running Ubuntu 18.04. After installing the relevant cuda drivers for the GPU and anaconda to manage our python libraries our training regimes were faster and could run for longer.

We chose Ubuntu as an operating system because it is free and open software. As the project is as much about working with data as it is programming, the Bash terminal has been an indispensable tool. While the learning curve has been steep, we find ourselves settled into Linux using vim to write all of the code and report. Vim is extremely powerful and extensible terminal text editor. Tmux, a terminal multiplexer, has extended the capability of the terminal by allowing us to perform many tasks within one window. The power of bash scripts has allowed to perform monotonous manual tasks such as renaming every file in a directory with one line of code.

Python Modules

Python was the language of choice due to the many well maintained modules geared for deep learning.

We used anaconda to maintain our python programming environment. Anaconda gave us complete control over package versions to avoid dependency problems. For example, we had to use python version 3.6.8 instead of the more recent python 3.7.3 as tensorflow could not yet run on python 3.7 at the time of writing. With anaconda we could have a tensorflow environment where we could use the older version of python and still have the ability to use the most recent releases by changing environments.

The core of the project from a programming perspective was keras. Keras is a high level python module that wraps tensorflow. Tensorflow is the most ubiquitous deep learning library and converts python code into GPU optimised c++ code. The parallel processing power of GPUs was extremely well suited for the intensive image operations and extensive micropropagation steps that training CNN's involves. Keras is mostly used for quick prototyping projects. Pure tensorflow is much more powerful, but significantly more complex than keras, however, we never found our work impeded by keras.

Numpy is an extremely popular python matrix library. It has an extensive set of methods that allow for a diverse set of optimised matrix operations. As our images were stored in numpy arrays, most every operation in the dataset preprocessing section relied on numpy functions. Keras is designed to work with datasets that are stored in numpy arrays due the optimizations built into the module.

As the images were stored in nifti files, we needed a way to properly read the files into memory. After looking at both simpleitk and nibabel, we settled on nibabel. We used nibabel to read the nifti files into memory from storage. We then extracted the image information, file header and coordinate affine from the nifti file. The image information was then stored in numpy arrays. The image header contains the dimensions of the pixels in the image. Our end goal was to generate approximations of volume, therefore we needed to correspond pixels to physical distances. The affine of a nifti image contains information that relates the coordinates of the nifti image to world coordinates in RAS+ space. This has to do with radiological conventions that govern how medical data is interacted with and viewed in computers. As each image and mask pair should have the same voxel dimensions, we used the affine and header from the image to transform any associated masks from numpy arrays to nifti files.

Dataset

The raw dataset that was provided to us consisted of pathological echocardiography images from 96 patients. Most patients had both a 2CH and 4CH view of their heart. The files were stored as zipped nifti images and were 2D greyscale videos of the heart through multiple diastole systole cycles. Constructing the final network that segmented cardiac ultrasounds was broken up into several successive steps:

1. Train a network to successfully segment ultrasound cone from the image

2. Train a network on 2CH blood pool data

3. Train a network on 4CH blood pool data

4. Train a network on both 2CH and 4CH blood pool data

5. Depending on the success of combining 2 and 4CH data, train a network to segment the myocardium

The purpose of segmenting the ultrasound cone from the rest of the image was to make successive training easier. Eliminating image data that is irrelevant to the segmentation of the heart saved the network from having to learn that the extraneous information is extraneous. The build up to the final network allowed us to gradually build up the training dataset.

The images needed to be converted into numpy arrays before they were used to train the network. The python library nilearn was used to convert the zipped nifti files into numpy arrays. The images were then resized to a resolution of 512 by 512 and the intensities were remapped from 0-255 to 0-1. The images were then collectively stored in a 4th order tensor with dimensions (number of images, X dimension, Y dimension, color channel). This allowed for easy manipulation with Keras

The masks went through the same resizing conversion from nifty to numpy array. As the mask images can contain cone or ventricle masks, one layer of the masks had to be chosen for training. This was accomplished via a thresholding operation to eliminate either the cone or ventricle mask. Once the correct mask configuration had been chosen, the masks were be resized and stored in a fourth order tensor as well.

Dataset Preprocessing

The data was presented to us in the form of 96 folders, each corresponding to a certain patient. The names of these folders were all unique as they came from different hospitals at different times and were acquired by different machines and operators. Most patients had both two and four chamber images, with some having multiple of either.

Once the images were sorted, manual segmentation began. Behind every competent neural network is a meticulously crafted dataset. To create a data point the blood pool and ultrasound cone from an individual frame of the nifti file was segmented. To do this ITK-snap was used to both open the nifti file and produce the segmentations. In order to increase variance within the dataset and therefore robustness of the trained network, the chosen frames were at the general point of diastole and systole. Most all of the scans taken were of pathological hearts, so finding exact diastole and systole was challenging. We usually settled on two or three frames from each scan that looked sufficiently different. If more data was needed at a later time, additional frames could be segmented. The product of the segmentation was a mask that contained information on both the boundaries of the ultrasound cone and blood pool of the left ventricle. The background of the mask was always zero. If a cone mask was present, the cone was always one with the ventricle mask, valued at two. If the ventricle was the only mask present then it had a value of one. TODO(image of mask with cone and vent and mask with only cone) As the ultrasound cone was quite simple to identify within the images, we only ended up producing 25 data points with the cone manually segmented.

In these cases we simply tried to find frames that looked like they contained the largest and smallest LV volumes. We also struggled with endocardial dropout as highlighted in the echocardiogram LVEF quantification guidelines.

In order to train a network on the images, they needed to be a standard resolution. Initially this resolution was to be 512 by 512. 512 is a power of two in order to avoid non integer resolutions as the images are downsampled in the unet. The raw images in the dataset varied in resolution with 512 by 512 being a rough mean. We later moved to a resolution of 800 by 800 as the largest image was 800 by 600. We wanted to zero pad the images up to a resolution instead of cropping them or interpolating them to a smaller size.Interpolation would give a reproduction of our images at a different resolution, while zero padding would encase the exact original image in zeros. Our zero padding scheme attempts to keep the original image in the center of the padding. While interpolating image data is common practice, the end goal of producing a measurement of LVEF in real world units places a higher degree of importance in preserving the pixel spacing and dimension information contained in the image headers. As the data generation script took shape, moving data from nifti to numpy array and back again without worrying about pixel spacing was a relief. Once the images and masks were a standard resolution, the image intensities needed to be scaled from zero to one. This was achieved with scalar division by 255.

Once the data had been standardised, we were able to construct the datasets for training. As some masks contained both ventricle and cone information, they had to be processed to produce a separate ventricle and cone mask. At the initial round of training the cone dataset consisted of 25 image and mask pairs and the 2CH dataset consisted of 43 image and mask pairs.

Training

Due to the small amount of data that we were working with along with memory restrictions, on the fly data augmentation was going to be crucial to the performance of the project. The premise was to apply random image transformations to an image and mask pair to generate a ”new” piece of data from a base image. We decided to use the keras data generator class, which is capable of performing a multitude of rigid image transformations.

While some non-rigid transformations could be useful, and are recommended in the unet paper [**?**], we needed them to be precise enough to only deform the ventricle as we wanted the cone to remain unreformed. We needed to tread the fine line of augmenting our data in a way that increased the variation within the dataset without going too far. Excessive augmentation that is too drastic could lead to the network struggling to find any sort of pattern without an immense amount of training.

Due to machine and time constraints training needed to be as efficient as possible. After some trial and error we settled on the following parameters to define our image and mask data generator:

1. Rotation range of 45 degrees
2. Width shift range of 10% of total image width
3. Height shift range of 10% of total image height
4. Shear range of 5 degrees
5. Zoom range of 0.1
6. Fill mode of nearest so most likely zero

We applied this same augmentation scheme to both the training and validation data to increase the size and variance of both sets. It is customary in machine learning, to take some data that could be used for training, and use it instead to test the trained model. Here we would give the trained model an image that it had never seen before and have it predict a mask. Comparison between the predicted and true masks gives us a better understanding at the effectiveness of the model. As the model is literally optimized to perform on the training data, using it to validate the network performance would not illustrate any ability to perform on real life data.

Training Parameters

We settled into training parameters early in the project. We needed to ensure that we would train long enough to get our validation loss sufficiently low, while minimizing training time. For the final cross validation training, we chose 1000 epochs. We knew that this was more than enough, so after one fold was trained, we noted the point of diminishing returns TODO and chose x epochs for the final run.

Our batch size paramter determined how many images were trained on each training step. When we attempted to have a batch size greater than one our machine ran out of memory and the training script crashed. This could be due to machine restrictions, but is more likely the large image resolution and size of unet.

The number of steps per epoch determines how batches are trained on per epoch. Ideally every image is trained on for every epoch. As out batch size was 1, we needed our steps per epoch to be equal to the number of training images, which in our case was 80.

The optimizer is the most important aspect of the network learning. This algorithm performs stochastic gradient descent with an adaptive learning rate. Instead of performing computationally expensive gradient descent on every single batch, the stochastic aspect randomly selects images to optimize for. This ensures that the network is being optimized on a set of images that reflect the variance of the entire dataset. The other major aspect to the adam optimizer is the dynamic learning rate. If the stochastic gradient descent step tells us in what direction we want to tweak our weights, the learning rate defines how much we change our weights. If our learning rate is too high, our steps will be too large and our network will not be able to settle into a configuration that minimizes our loss. If our learning rate is too low, it will take too long for our network to reach a loss minimum. Adam handles this problem by dynamically changing the learning rate based on the parameters of the problem. If a certain feature is very common, for example the apex of the heart in most scans is usually a source of good signal, then adam will reduce the learning rate for this feature. While for sparser features would require a larger learning rate to ensure the network learns about them even though they are rare. An example of a rare feature could be the presence of hypertrophic myocardium. This means that there is no longer one global learning rate but a learning rate that is associated with different features within the training data. The other defining feature of the adam algorithm is the root mean square propagation(RMSprop). RMSprop also adapts the per feature learning rates based on the average of recent learning rates. This gives adam good robustness on noisey sparse datasets [**?**].

loss To track the loss of the network we decided to use binary cross entropy(BCE) and dice loss. Binary cross entropy penalizes (Eq. 4) the network when its prediction is wrong, but the penalty grows much higher the farther away the predicted pixel value is from the true label.

$$bce = -\frac{1}{N}\sum_{n=1}^{N} -r_n \log p_n - (1 - r_n)\log(1 - p_n) \tag{4}$$

The r_n term is a voxel intensity from our ground truth label and p_n is a predicted pixel intensity.

Dice loss measures the overlap between the predicted label and the ground truth. For the two classes in our segmentation task, we use the 2 class dice loss as calculated by EQ

$$DL_2 = 1 - \frac{\sum_{n=1}^{N}(p_n r_n + \epsilon)}{\sum_{n=1}^{N} r_n + \epsilon} - \frac{\sum_{n=1}^{N}(1 - p_n)(1 - r_n + \epsilon)}{\sum_{n=1}^{N} 2 - p_n - r_n + \epsilon} \tag{5}$$

With r_n and p_n as the nth voxels of the ground truth image R and predicted image P. The ϵ term ensures that division by zero does not happen when images R and P are empty.

The dice coefficient ranges from 0-1, with 0 being no overlap and 1 being a perfect match. To convert the dice coefficient into dice loss we simply subtract it from 1 to invert the metric so that we can minimise it

To combine the two metrics we simply add them together and minimize the sum as our loss function.

saving In order to monitor the progress of our network and save training progress we utilized a number of keras callbacks. Callbacks are simply functions that can be called during training. We used the CSVLogger, ModelCheckpoint and LearningRateScheduler. The CSVLogger is run after every epoch and saves developer defined metrics the epoch number, training and validation BCE dice loss, and training and validation dice loss. ModelCheckpoint saves the models

Class Organisation

Once we completed our first successful cone segmentations within google colab, it was clear that we needed to move to a more object oriented approach. We decided to store relevant functions for data preprocessing within a module. We wanted to instantiate a unet object from a predefined parameters object that could then train, make predictions, and load previously trained models.

Data Generation

After constructing the cone and 2CH datasets, we saw a need to streamline the process. We were spending a considerable amount of time entering filenames and clicking on images to open in itk-snap for the segmentation. We then decided to write a script that would iterate over every nifti file in a given directory and:

1. open the nifti in itk-snap for the user to view the image
2. prom the user to input which frames would be segmented
3. give the image to a partially trained network
4. post process the generated mask and convert it into nifti file
5. open the image with the mask in itk-snap for user refinement

As the names of the masks were automatically generated based on the user inputted frame and original image name the user no longer had to type in the name of the masks. We also defined the directory where the masks and images were to saved within the script so the user did not have to specify that within itk-snap either. This script proved challenging due to the movement of image data between np and nifti but the overall time savings outweighed the development time.

Validation

A network is only as good as it can be proved to be.

K Fold Validation

To assess the performance of our model we will both assess the performance with a K fold cross validation scheme and compare the accuracy of the predicted LVEF and LVM. K fold cross validation involves a number of steps.

1. Split the dataset into K equal groups
2. Select one group for the test set and have the rest be training data
3. train K models so that every data point gets to be in the test set

In our case we chose K to be 5. This fit well with out final blood pool dataset size of 100 image mask pairs as it is cleanly divisible by 5.

ROC Curve

As the network returns a probability map of the ventricle and not a binary mask, we must threshold the masks before we compare them to ground truth.

In order to asess the best threshold value we will plot the networks true positive rate over its false positive rate across a range of thresholds.

5 Results

To assess the performance of our network under the 5 fold cross validation scheme, we tested every image in each fold's test set to get an average dice. Our 5 fold cross validation scheme generated the following metrics

The training history for one of our folds is as follows.

To test the effusiveness of our model we constructed an ROC curve for each cross validation model..

The reason LVEF only requires the segmentation of two frames is because of the cost of making those segmentations. As we have a network that can segment every frame, we can plot the volume of the LV through time. This is a result that would have been arduous to perform without our automatic methods.

6 Discussion

Overall we are quite pleased with our network's performance on segmenting the left ventricle. The cross validated ROC curve shows excellent auc metrics om account of the high trur positive rate and low false positive rate. The auc, which is related to the accuracy of the test is very close to 1 for every fold. This shows that the model does not under or overestimate the area of the blood pool. However, the performance on segmenting the myocardium was not on par with the blood pool results. We constructed fewer data points for the myocardium becasue there were many echo images that did not contain the full epicardium. The thickness of the myocardium was also much more difficult to determine. We extrapolated heavily from regions with more defined thicknesses. We also manually segmented the myocardium, while most methods to calculate the LVM manually subtract the blood pool volume from the volume of the epicardium. Considering the effectiveness of our model on the bloodpool, perhaps the simpler segmentaion of the epicardium would have been more effective.

Short axis images would allow us to not only be more sure about the thickness but also actually calculate the LVM of the heart. If short axis, 2CH, and 4CH views are all being collected, we recommend moving from 2D echos to 3D if LVM automatic LVM quantification is desired.

As we have

7 Further Research

If another team or person were to continue exploring deep learning on 2D echocardiogram data, this is where we recommend they spend their efforts.

While we stated the need for short axis images, in order to determine the LVM, we recomend 3D echo methods. 3D images remove geometric assumptions and would lead to more robust algorithms, even when irregular hearts are imaged.

When we were manually segmenting the ventricle and myocardium from the echo images, in many cases it was exceedingly difficult to discern where the walls of the heart were. We also consulted a practising cardiologist, who corroborated this saying that in some cases the segmentation can be difficult. But what humans can do that this network cannot, is watch the video to contextualize a certain frame. In order to allow a network accomplish this, an LSTM may be employed. LTSM's employ a recurrent nature where the network would watch the entire echocardiogram video before attempting to segment the individual frames. The idea being that the network will learn general information about the behaviour of the heart through time and use that information to segment individual frames.

Before neural networks are put into practice in hospitals, doctors and legislators need to be sure that networks can be trusted. One possible solution to this would be to give networks the ability to explain why they make the decisions that they do and how sure they are about their answer. A practical extension of this idea for our network could be the deletion of regions within the image that inform its decisions. There is research being done... We would also need to compare the performance of these networks to humans. We make several claims that this network will reduce inter-operator variance, this makes sense as the nature of learning the dataset averages the biases that went into making it. Perhaps we need to involve a larger group of people in the manual segmentation to increase the variance within the segmentations and allow the network to learn how more people segment echoes.

longitudinal strain computer resources spec around batch size architecture We only investigated a unet architecture. While the unet is standard for biomedical segmentation, perhaps some other architectures like densenet [**?**] could be explored. For our data augmentation, we only performed rigid transformations. Our thinking behind this was the network would probably use the edges of the ultrasound cone to find the ventricle and elastic deformations could disrupt this. Elastic deformations that are constrained to the interior of the cone should be explored.