# Sensorless Ultrasound Probe 6DoF Pose Estimation Through the Use of CNNs on Image Data

by

Elise Yuan Xue

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018



Author .....
Department of Electrical Engineering and Computer Science
May 25, 2018

Certified by .....
Brian W. Anthony
Principal Research Scientist, Mechanical Engineering,
Institute for Medical Engineering and Science
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair of Master of Engineering Thesis Committee
Thesis Committee

# Sensorless Ultrasound Probe 6DoF Pose Estimation Through the Use of CNNs on Image Data

by

Elise Yuan Xue

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

Ultrasound probe pose estimation has many applications in medical practice and research. Currently, ultrasound probe pose estimation with respect to the human body requires the use of sensors attached to the ultrasound probe, and may get computationally costly. We explore the use of Convolutional Neural Networks (CNNs) to provide sensorless pose estimation. The Ultrasound CNN model proposed in this paper learns to regress the six degree of freedom (6-DoF) camera pose from a single ultrasound image in an end-to-end manner. Ultrasound images are easier to obtain than other forms of medical imaging, but suffer from poor quality, which will be a challenge for the Ultrasound CNN model. The most promising model from our experiments is a 23 layer deep CNN based off of GoogLeNet. In previous literature, CNNs have demonstrated that they can be used to solve complicated out of image plane regression problems. We show how the proposed method can regress the 6DoF pose within a certain degree of accuracy.

Thesis Supervisor: Brian W. Anthony
Title: Principal Research Scientist, Mechanical Engineering,
Institute for Medical Engineering and Science

# Acknowledgments

I would like to thank my supervisor Professor Brian Anthony and my lab mates in the Device Realization Lab for their support and feedback on this project. In particular, I would like to thank Alex Benjamin for his inspiration for this project and for his help in data collection, and Anne Pigula and Jonathan Fincke for lending their computing resources to this project.

This project also would not have gotten this far without the feedback and advice from the researchers in CSAIL's Medical Vision team. I would like to thank Adrian Dalca for his wonderful literature recommendations on machine learning in medical vision. Lastly, I would most of all like to thank Jay Patel for his vast knowledge and incredibly insightful advice on medical imaging and convolutional neural network optimization. I have learned a lot through our meetings and was able to continually improve the quality of the neural network in this project thanks to his advice.

This masters thesis has been examined by a Committee of the Department of Electrical Engineering and Computer Science as follows:

Professor Brian W. Anthony .....
Thesis Supervisor
Principal Research Scientist

Katrina LaCurts .....
Thesis Committee
Chair of Master of Engineering Thesis Committee

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The medical community has major interest in ultrasound probe pose estimation. Doctors could use pose estimation to gain information about the exact location of scans on the body, and pose information could be used in research for 3D reconstruction. Pose estimation typically require the use of sensors attached to the ultrasound probe[22][18], and may get computationally costly due to the algorithms involved. By using machine learning, we can train a convolutional neural network to regress the six degree of freedom (6-DOF) ultrasound probe pose on a body given an ultrasound B-mode image, thus achieving sensorless pose estimation in an end-to-end manner.

This section will discuss the background and related research of this project, as well as the project's motivation.

Chapter 2 describes the implementation of the project. In particular, we will describe the data, network architecture, and computing machinery.

Chapter 3 describes the results of the multiple iterations and improvements of the network. We will discuss the shortcomings and what was learned from each subsequent iteration.

Chapter 4 describes the final results and provides analysis and discussion of the results. We will also suggest future work based on the shortcomings of the current model.

A link and description to the github for this project are provided in the Appendix.

## 1.1 Pose Estimation in Computer Vision

In computer vision, estimating the camera pose from $n$ 3D-to-2D point correspondences is a fundamental and well understood problem[6]. The ability to estimate the relative pose between camera views is essential for many computer vision applications, such as structure from motion (SfM)[23], simultaneous localization and mapping (SLAM)[12] and visual odometry[20]. This is a fundamental problem in augmented reality, with applications of using the camera pose with respect to the object to do 3D rendering, and in the case of robotics to obtain an object pose to later manipulate the object[6].

Estimating the camera pose requires estimating the six degrees of freedom (translation and rotation along the x, y, and z axes) of the pose and the following calibration parameters: focal length, principal point, aspect ratio and skew[6]. It could be established with a minimum of 6 correspondences, using the well known Direct Linear Transform (DLT) algorithm. There are, though, several simplifications to the problem which turn into an extensive list of different algorithms that improve the accuracy of the DLT[24]. As a result, pose estimation is not a trivial problem to solve due to the fact that the most common issue in image processing is the computational cost of applying numerous algorithms or mathematical operations for solving a problem which humans naturally do immediately and in real time. [6]

## 1.2 Neural Networks in Pose Estimation

In terms of using neural networks for pose estimation in ultrasound images, there is little work that has been done. However, CNNs have been used for camera-based pose estimation, as well as in pose estimation related tasks such as camera motion estimation, camera relocalization, and homography estimation. Ummenhofer et al.[28] proposed a CNN architecture for depth and relative camera motion estimation between two images. DeTone et al. [13] proposed a CNN architecture for estimating the relative homography between two images by regressing a 4-point homography parameterization with an Euclidean loss. Both of these methods estimated camera movement between two images, thus predicting the difference vector between the poses for the camera between one image to the next. These are both

camera based methods, which are different than the images given by an ultrasound probe. However, the ability to track features and movement shown by these two methods suggest that the same could be done on ultrasound images.

Melekhov et al. [20] used CNNs for relative camera pose estimation between two cameras. They used a siamese network, which are two feature-based CNNs that are trained in parallel and with respect to each other. The input to the siamese network was two images, one image from each camera. Their network predicted a 7-dimensional relative camera pose vector $\Delta p$ containing the relative orientation vector $\Delta q$ (4-dimensional quaternion), and the relative position, i.e. translation vector $\Delta t$ (3-dimensional), so that $\Delta p = [\Delta q, \Delta t]$. The resulting vector given by Melekhov *et al.* is the same pose vector that this paper will attempt to learn, with the exception that the quaternion will be in Euclidean angles. However, Melekhov *et al.* used input on two cameras to learn the relative pose. Given a single ultrasound image, the Ultrasound CNN network proposed by this paper will attempt to learn the absolute pose of the ultrasound probe. Kendall *et al.* [16] proposed a CNN-based method for absolute 6-DoF camera relocalization pose estimation, using an altered version of GoogLeNet. The PoseNet architecture created by Kendall *et al.* was trained on the absolute poses of cameras given images of a street in Cambridge, UK, and succeeded in pinpointing the cameras on a map. PoseNet will be used in this paper's CNN model as a baseline neural network pose estimation model.

## 1.3 Ultrasound Imaging

Pose estimation is a major field of study in medical imaging. In medical imaging, there are different types of image modalities, such as CT scan, MRI, PET, and ultrasound (US), used to support medical diagnoses. We look at ultrasound imaging, as it has the benefits of free-hand imaging, non-invasiveness, compactness, and low cost[30]. It is often the imaging method of choice for assistance with surgical operations and real-time diagnoses. However, ultrasound image quality is generally low resolution and variable: field of view (FOV) in ultrasound images is very limited, and in some cases only 2D cross-sectional images are obtained. These shortcomings can impede doctors from making correct diagnoses for patients. To address this

issues, literature suggests fusing ultrasound with other, but more time consuming, methods of medical imaging. Before the surgical operation, 3D models of target parts are obtained by richer modalities such as CT, MRI, and PET. Ultrasound images are later superimposed onto these models to create a rich source of information doctors can use for diagnoses[29]. Efficient pose estimation in ultrasound can assist doctors in locating areas of interest on the patient's body, as well as give doctors information on where in the body they are looking at given an image. Pose estimation is also key to being able to superimpose medical images obtained during the operation on 3D models. Current methods of ultrasound pose estimation are computationally costly, and require sensors to be attached to the ultrasound probe to provide enough information to calculate the pose[21][27]. For this thesis project, we propose a way to estimate ultrasound poses without the use of sensors, using convoutional neural networks (CNNs). Although sensorless pose estimation in ultrasound have been attempted before with more algorithms based methods[29], we propose to approach this problem from a machine learning standpoint. After training, using CNNs may dramatically reduce computational cost because the images will be processed through a trained model, and will remove the need for sensors because only the ultrasound image will be needed to predict the probe pose.

## 1.4 Project Motivation

From previous work in using machine learning in pose estimation and related tasks, we hypothesize that pose estimation in ultrasound imaging can be predicted using techniques in machine learning, specifically Convolutional Neural Networks (CNNs). CNNs are neural networks that are designed to have images as input, and can train to extract certain information from the input by performing a series of transformations on the image using convolutional and pooling layers[2]. Although ultrasound image quality is rather low resolution, we can still see a variety of features: veins, bones, tissue, etc. Figure **??** shows an ultrasound scan of the abdomen with organs an features highlighted. A trained doctor can identify these features and could make an approximate estimate as to where the ultrasound image was taken in the organ or on the body. With this idea in mind, we attempt multiple CNN architectures to reach a reasonable setup to arrive at the most accurate pose predictions.

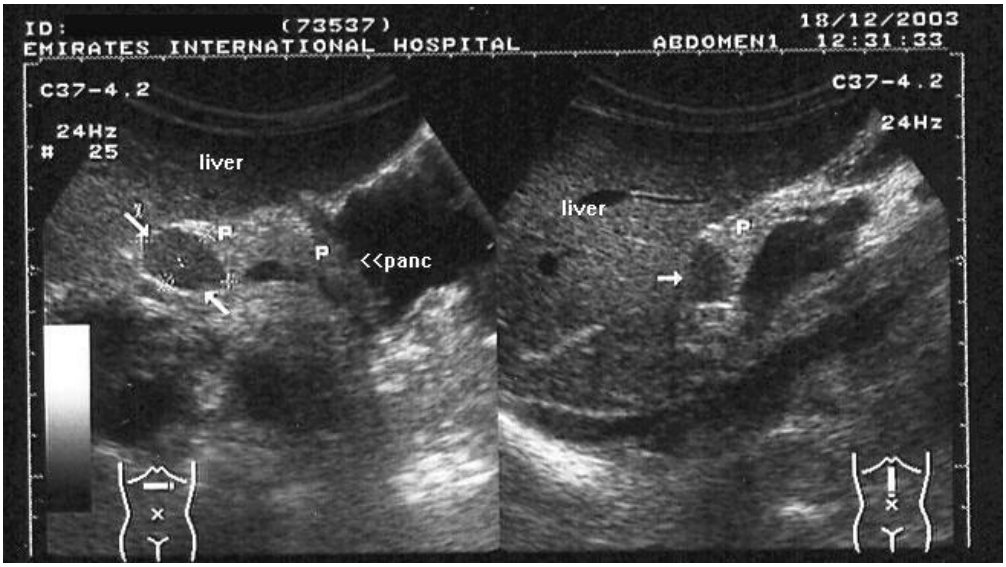



Figure 1-1: An example of a sonography of the abdomen. The ultrasound image shows a rounded mass in the region of the pancreatic head and isthmus, indicated by arrows. The liver and pancreas are labled[10].

## 1.5 Project Use Case

The ultrasound CNN was implemented for use on a patient-to-patient basis. If this model were to be used in hospitals, the network would have to be trained on images from one patient at a time. This is to give the model the best knowledge of the organ that it is trained on for each patient.

For one possible scenario, a patient has repeated visits to the hospital concerning their liver. On the first visit, images of their liver are taken with the ultrasound probe, which are then sent to the Ultrasound CNN model to train. For this data collection, an OptiTrack or similar machine for 6DoF pose generation must be used to obtain ground truth ultrasound probe pose data for each of the images. This step will require that sensors are attached to the probe for accurate data collection.

In the time between the client's first visit and their next visit, the model is trained to be able to predict the pose given an ultrasound image of that patient's organ. During the next several visits of the patient, the model can be used to provide reasonable estimates of ultrasound poses without the use of sensors.

Currently, this individual patient model for the Ultrasound CNN is inefficient, as it re-

quires that the model is retrained every time there is a new patient. For future work, we hope to generalize this model for use on multiple patients. One option for expanding the Ultrasound CNN model is to train on data from multiple patients, however this is challenging because if the model becomes too general than the overall pose estimation accuracy amongst patients will decrease. The goal for the current model is high ultrasound probe pose estimation accuracy for a given patient.

# Chapter 2

# Model and Implementation

## 2.1 Data

In Prof. Anthony's lab in the Institute for Medical Engineering and Science (IMES), we used an OptiTrack motion capture system to collect ground truth ultrasound images with corresponding poses (x, y, and z position, x, y, and z rotation) to millimeter accuracy. All the coordinates were collected using the OptiTrack[7]. The OptiTrack is an optical tracking system which utilizes sensors to track an object's pose. There is extensive use of the OptiTrack's motion capture capabilities in AR and VR[7]. A series of sensors were attached to the ultrasound probe, then the scans were taken in front of the OptiTrack camera to measure the ground truth poses. For convenience, the ultrasound image captures were done on the left forearm. For this first dataset, the scans were taken on the surface of the inner forearm with varying degrees of rotation with the probe. The dataset includes both cross-section and longitudinal cross-section scans, as well as fanning of the ultrasound probe. This was to capture as much of the six degrees of freedom as possible while maintaining contact with the probe to the arm. The data collection setup is shown in Figure 2-2. We collected 9,416 ultrasound images with corresponding pose data encompassing as many ultrasound pose orientations as feasible.

There were several limitations to data collection. Although the objective of the first data collection was to collect images with variation in all six degrees of freedom, the not all degrees were captured due to the constraint that the ultrasound probe needed to be in

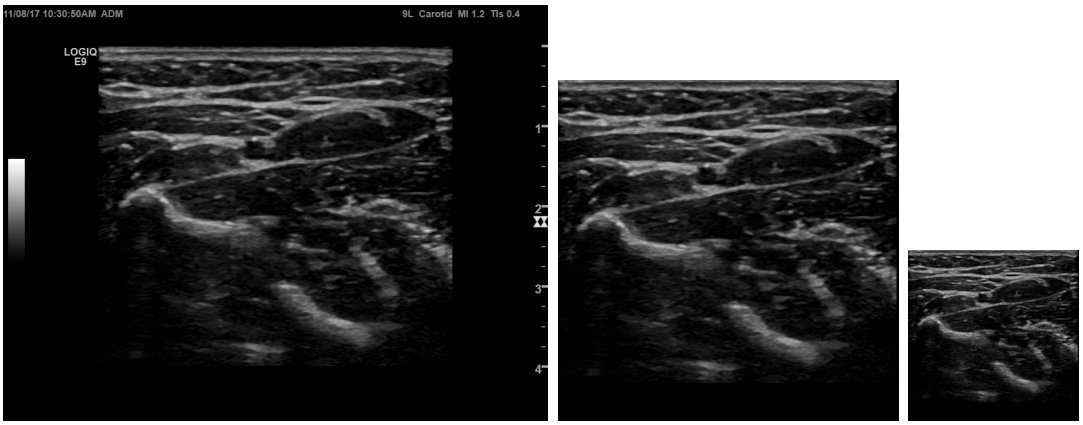



Figure 2-1: Ultrasound images of a forearm taken with a GE LOGIQ 9 ultrasound machine at the MIT Institute for Medical Engineering and Science (IMES). (Left) Original image outputted by the ultrasound machine. Includes image information and machine settings. (Middle) Cropped square image of size 550 pixels. (Right) Scaled and subsampled version of size 256. Used as input to the neural network model.

contact with the arm. As shown in the coordinate system in Figure 2-3, moving the probe in the Y-axis while in contact with the arm would have been very difficult. Also, because the images were taken by hand, there may have been uneven pressure applied to the arm by the probe, resulting in some tissue compression that may have affected the ultrasound scans. These may be issues to consider in future data collection.

From the images in Figure 2-1, we can see the skin layers, veins, tissue and bone give cues as to where the image was taken. From general observation we see that the image quality is low resolution. Although boundaries and edges can be seen clearly in the grayscale image, there is a lack of distinct texture and trackable features. These are all properties of current ultrasound imaging technology.

Several different datasets were collected to use on the ultrasound CNN architecture. The first dataset, with full six degree of freedom motion consisted of 9418 images and corresponding 6DoF pose vectors on a subject's forearm. A second dataset of 1687 images and pose vectors was taken of the same subject's forearm, but with only movement in one axis. Images were taken only along the x-axis as shown in Figure 2-3 while maintaining constant rotational orientation. Finally, a third dataset of 4059 images and pose vectors was taken of an ultrasound phantom organ. In each dataset, 70% of the images were used for training and the remaining 30% were used for testing. Position data taken from the OptiTrack are


Figure 2-2: (Left) Data collection setup. (1) GE LOGIQ 9 ultrasound machine, (2) Ultrasound probe with Optitrack sensors attached, (3) Optitrack camera. (Right) Close-up image of the ultrasound probe. The Optitrack sensors are circled in green.

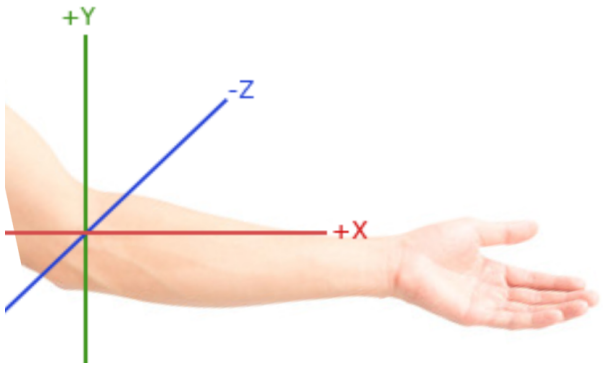



Figure 2-3: Diagram of a coordinate system similar to the one used in the OptiTrack superimposed on a model forearm.

in centimeters and rotation data is in degrees.

### 2.1.1 Data Processing

The original basis of our network is GoogleNet, which takes in `256x256` pixel images as input. The ultrasound image data had to be cropped and subsampled to the correct size to be passed through the convolutional neural network model. The original ultrasound images were of size `850x649` pixels, and contained a wide border and text that was not needed in our network. The border was cropped, leaving a `555x555` size section containing the actual ultrasound information. These images were then subsampled down to `256x256` pixels so they could be put into the model network.

## 2.2 GoogLeNet

Drawing inspiration from the works of Kendall *et al.* [16] and Tajbakhsh *et al.* [26], the first implementation of the ultrasound pose regression network was an altered version of GoogLeNet[25], a deep neural network architecture for image classification.

GoogLeNet is a 22 layer (27 layers counting the non parametrized pooling layers) convolutional network with nine "inception modules". GoogLeNet also included two additional intermediate classifiers which are discarded at runtime. The changes we made to GoogLeNet will be explained in section 2.3.

GoogLeNet, also known as Inception, from Google was the winner of the ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2014 competition, an image classification competition. It achieved a top-5 error rate of 6.67%, which was very close to the best human level accuracy, which was 5.1%. GoogLeNet was inspired by LeNet[1] but implemented a novel element called an inception module. The module utilizes several very small convolutions in order to drastically reduce the number of trainable parameters. The resulting 22 layer architecture had 4 million parameters, much less than AlexNet[2], which only contained 8

[1]Also known as LeNet-5, a pioneering 7-level CNN by LeCun *et al.* in 1998. LeNet was used to classify handwritten digits in the form of 32x32 pixel grayscale images.[19]

[2]ILSVRC 2012 winner, achieved top-5 error of 15.3%, more than ten percentage points lower than the runner up. The network had a very similar architecture as LeNet but was deeper, with more filters per layer,

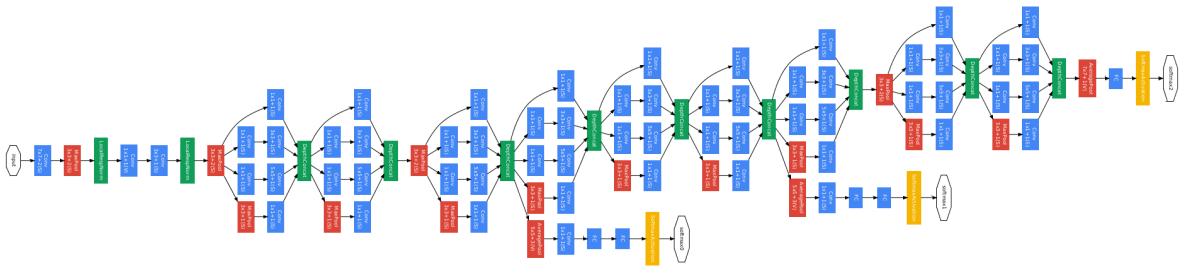


Figure 2-4: GoogLeNet[25] architecture, displaying the six inception modules and three output layers. Blue nodes are convolution layers, red nodes are max pooling, and the yellow nodes are output layers. The first two green nodes are normalization layers, and the other green nodes seen after each inception module is a concatenation layer.

layers but had 60 million parameters.[4]

### 2.2.1 Inception Layers

The basis behind the inception layer is to cover a bigger area of the image, but also keep a fine resolution for the higher resolution details on the images. Inception uses convolutional layers in parallel with different filter sizes to get higher resolution features (1x1 filters) to lower resolution features (5x5 filters). Using 1x1 filters allow for cross-channel correlations, while larger filters (i.e. 3x3) allow for both cross-channel and cross-spatial correlations. The inception modules are a series of trainable filters with different sizes to handle better multiple objects scales. The most straightforward way to improve performance in deep learning is to make the network deeper (add more layers) and train on more data. GoogLeNet uses 9 inception modules and has 4 million parameters, meaning that it is more prone to overfit. GoogLeNet mitigates the issue of overfitting by using techniques such as dropout and data augmentation[25].

and with stacked convolutional layers.[17]

## 2.3 Ultrasound CNN Network Implementation

In order to emulate the six degree of freedom absolute pose relocalization system implemented by Kendall *et al.* , the following changes were added to GoogLeNet as mentioned in their PoseNet paper[16]. The PoseNet model will serve as a baseline to the Ultrasound CNN model:

- Replace all three softmax classifiers with fully connected layers with linear activation functions. The softmax layers were removed and each final fully connected layer was modified to output a pose vector of 6 dimensions representing position and orientation.
- Insert another fully connected layer before the final layer of feature size 2048. This is to form a feature vector with pose information which will later be analyzed for generalization.
- Change the original cross entropy loss function to L2-norm least squares loss.

Through experimentation and literature search, the following changes have also been added to the network architecture:

- Remove the local response normalization layers and replace with batch normalization layers after each convolution layer. The original GoogLeNet used local response normalization, which has been widely faded out in favor of techniques such as batch normalization and dropout[2].
- Change the L2-norm least squares loss function in PoseNet to L1-norm least absolute deviations loss.

### 2.3.1 Batch Normalization

When training deep neural networks, the distribution of each layer's inputs changes due the parameters of the previous layers changing. This slows down training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. This is called internal covariate shift[19], and this

problem can be addressed by normalizing layer inputs[15]. Normalization is made a part of the model architecture and the network performs the normalization for each training mini-batch. In the case of this network, the batch size is 5. Batch normalization allows for much higher learning rates and more flexibility in weight initialization. It can also act as a regularizer, in some cases eliminating the need for dropout[3]. Applied to a state-of-the-art image classification model, Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin. An ensemble of batch-normalized networks trained on ImageNet data achieved the best published result (as of 2017) on ImageNet classification: reaching 4.9% top-5 validation error (and 4.8% test error), exceeding the accuracy of human raters, which was 5.1%[15].

Batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. However, after this shift/scale of activation outputs, the weights in the next layer might no longer optimal. Stochastic gradient descent (SGD) undoes this normalization in order to minimize the loss function.

Batch normalization thus adds two trainable parameters to each layer, a standard deviation parameter ($\gamma$) and add a mean parameter ($\beta$). As a result, batch normalization lets SGD do the denormalization by changing only these two parameters for each activation, instead of changing all the weights in the network. The equations for batch normalization are shown in Figure **??**, which are from the original batch normalization paper[15].

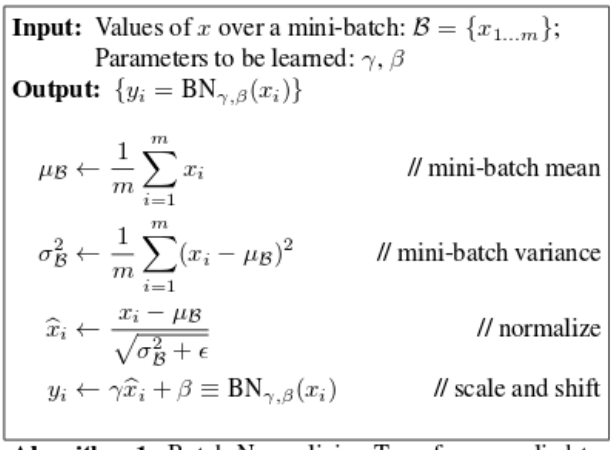
## 2.4 Loss Function

### 2.4.1 L2 Loss

The first pass of the network was the PoseNet model from Kendall *et al.* , which used GoogLeNet as it's base. This version used an L2-norm least squares loss function, given by

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

[3]Dropout refers to "dropping" or ignoring nodes (i.e. neurons) at random (with probability $p$, also known as the dropout rate) during the training phase. These nodes are not considered during a particular forward or backward pass. This is to prevent the network from overfitting the training dataset[14]



**Input:** Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: $\gamma$, $\beta$
**Output:** $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

**Algorithm 1:** Batch Normalizing Transform, applied to activation $x$ over a mini-batch.

Figure 2-5: Batch normalization from Ioffe *et al.* . Normalization is done by subtracting the inputs by the batch mean and dividing by the batch standard deviation

where $S$ is the loss, $y_i$ is the ground truth pose vector of the $i$th sample, and $f(x_i)$ is the CNN output of the pose vector of given by the $i$th sample. L2-norm is a widely used loss function, as it is very sensitive to differences between the training output and ground truth (using a squared loss as opposed to linear) and it has a stable, single solution. A "stable" solution means that for any small adjustment of a data point, the regression line will only move slightly as opposed to jumping past a configuration. This is due to the fact that the regression parameters are continuous functions of the data. L2-norm loss is also less computationally expensive than L1-norm loss.

However, the one of the shortcomings of L2-norm loss is that it is not very robust. Outliers in the data can greatly affect the regression solution given by L2-norm loss[5].

### 2.4.2 L1 Loss

In later iterations of the network, using L1-norm least absolute deviations loss performed better in terms of accuracy. L1-norm loss is given by

$$S = \sum_{i=1}^{n} |y_i - f(x_i)|$$

where the same as above, $S$ is the loss, $y_i$ is the ground truth pose vector of the $i$th sample,

Table 2.1: AWS p2.8xlarge Instance Specifications

| Instance Name | GPU Count | vCPU Count | Memory | Parallel Processing Cores | GPU Memory | Network Performance |
|---|---|---|---|---|---|---|
| p2.8xlarge | 8 | 32 | 488 GiB | 19,968 | 96 GiB | 10 Gigabit |

and $f(x_i)$ is the CNN output of the pose vector of the $i$th sample. L1-norm is another widely used loss function in deep learning. While not as sensitive as the L2-norm, L1-norm is more robust, meaning that outliers do not have as much of an affect on the regression solution as the L2-norm.

However, the shortcomings of the L1 loss is that it is not as stable and can possibly have multiple solutions. L1-norm is also more computationally expensive. It does not have a closed form solution because it is a non-differenciable piecewise function, as it involves an absolute value[5].

## 2.5 GPU and Computing Resources

GoogLeNet is a deep neural network with many parameters, and would need a considerable amount of computing power to train in an acceptable amount time. This network was trained on a `p2.8xlarge` computing instance on Amazon Web Services' (AWS) Elastic Compute Cloud (EC2). The specifications of Amazon's `p2.8xlarge` instance are listed in Table 2.1[1].

The `p2.8xlarge` instance uses 8 NVIDIA K80 GPUs, which are compatible with CUDA and OpenCL. Amazon's P2 instances also use GPUDirect™(peer-to-peer GPU communication) capabilities for up to 16 GPUs, so that multiple GPUs can work together within a single host[1].

The instance was launched from an Amazon Machine Image (AMI) with Ubuntu 16.04 LTS. The AMI included Tensorflow GPU packages, the CUDA computing platform, and the CUDA Deep Neural Network library (cuDNN) pre-installed and set up[8].

Each version of the network took one to three days to train 30 epochs on the aforementioned architecture.

## 2.6 Final Implementation of the Ultrasound Pose CNN

The final implementation of the ultrasound CNN was an implementation of GoogLeNet that took in 256x256 pixel grayscale ultrasound images as input and regressed the 6 degree of freedom pose vector as output. The network used batch normalization after every convolutional layer and L1-norm loss on the output vector. In terms of other iterations of our model, this configuration worked the best by far in terms of accuracy in both position and rotation.

# Chapter 3

# Network Iterations and Analysis

This section will discuss the different iterations of our ultrasound pose CNN network, and the changes and improvements with each step. Each version of the network trained for 20 epochs on the forearm dataset with the full unconstrained 6DoF range. The training set was 6592 images, and the results are given by the performance on the test set of 2826 images. The results for all of the iterations are given by Table 3.2, and will be referenced in the proceeding sections in this chapter.

## 3.1 GoogLeNet First Pass Based on PoseNet

The first implementation of the network was based off of Kendall *et al.* 's implementation of PoseNet, which was an altered version of GoogLeNet that focused on the regression of 6DoF pose vectors from camera based images of buildings in Cambridge, UK[16]. The network was trained with the hyperparameters in Table 3.1.

The tensorflow implementation of an exponential decaying learning rate was used with the network. It is defined as:

$$a = a_0 k^{t/T}$$

Where $a$ is the decayed learning rate, $a_0$ is the original learning rate, $k$ is the decay rate, $t$ is the global time steps, and $T$ is decay steps[9]. $a_0$, $k$, and $T$ are set as hyperparameters in

Table 3.1: Hyperparameters for Ultrasound CNN (Default)

| Parameter Name | Value |
|---|---|
| `image_size` | 256 |
| `batch_size` | 5 |
| `thread_num` | 10 |
| `learning_rate` | 0.00316 |
| `lr_decay` | 0.00316 |
| `decay_steps` | 210000 |
| `max_epochs` | 20 |

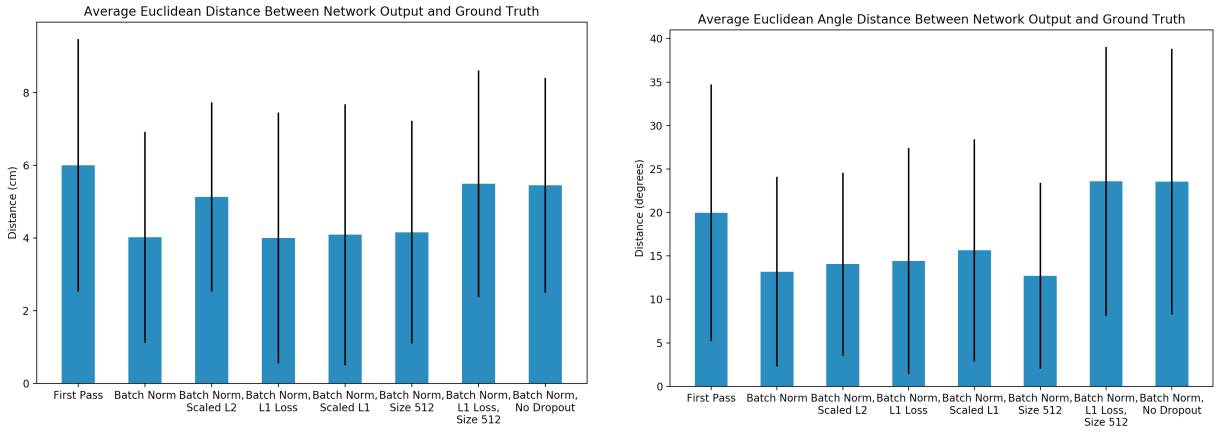



Figure 3-1: Graphs depicting the average Euclidean distance (left) and Euclidean angle distance (right) between the network output and ground truth on the test data.

Table 3.2: Performance Accuracies of Various Ultrasound CNN Models

Performance of various models on the 6DoF unconstrained forearm dataset. Displays percentage of test dataset where the estimate was within 1cm, 2cm, 10°, and both 1cm and 10° of ground truth. The model with the best performance is in bold.

| Model Description | % Accuracy within 1cm | % Accuracy within 2cm | % Accuracy within 10° | % Accuracy within 1cm and 10° |
|---|---|---|---|---|
| Baseline (PoseNet) | 1.91% | 5.52% | 29.98% | 0.39% |
| Batch Norm | 3.47% | 16.34% | 49.75% | 3.22% |
| Batch Norm, Scaled L2-norm | 0.53% | 6.90% | 44.83% | 0.28% |
| **Batch Norm, L1-norm Loss** | **10.65%** | **34.75%** | **50.92%** | **9.24%** |
| Batch Norm, Scaled L1-norm | 10.65% | 35.56% | 45.08% | 8.00% |
| Batch Norm, Size 512 | 5.13% | 28.13% | 52.87% | 4.74% |
| Batch Norm, L1-norm Loss, Size 512 | 2.37% | 8.92% | 25.80% | 1.10% |
| Batch Norm, No Dropout | 1.78% | 8.95% | 22.04% | 1.01% |

the network, and are labeled as `learning_rate`, `lr_decay`, and `decay_steps` respectively in Table 3.1.

PoseNet uses L2-norm loss as the loss function, and the original implementation uses local response normalization instead of batch normalization.

### 3.1.1 Result and Shortcomings

The baseline, PoseNet, had an mean Euclidean loss of 6.00, a mean Euclidean angle loss of 19.95, percent accuracy within 1 cm of 1.91%, percent accuracy within 2 cm of 5.52%, percent angle accuracy within 10 degrees of 29.98%, and a combined percent accuracy within 1cm and 10 degrees of 0.39%. The performance of the baseline is rather poor as the accuracy is very low.

There are a number of reasons as to why the network may have fell short: the starting hyperparameters and network architecture were not optimal to efficiently train the network, or the loss function was not efficient. The issue could also be with the data set: the input data was too low resolution to extract trackable features in the network, or that the images were not spatially unique enough for there to be major differences between the images at different positions or orientations.

In order to make training faster and more efficient, one of the common optimizations is to add normalization to the layer inputs, which was added to the next iteration.

## 3.2 Adding Batch Normalization

The paper for GoogLeNet (2014) was published before Ioffe *et al.* released their paper on batch normalization (2015). However, the concept of normalizing the layer inputs existed beforehand, and we see that GoogLeNet uses two local response normalization layers in the network before the inception modules[25]. In recent literature, however, older methods of normalization such as local response normalization have widely faded out in favor of batch normalization[2].

The second iteration removed the local response normalization layers and added batch normalization after each convolution layer in the network. As explained in Chapter 2, batch

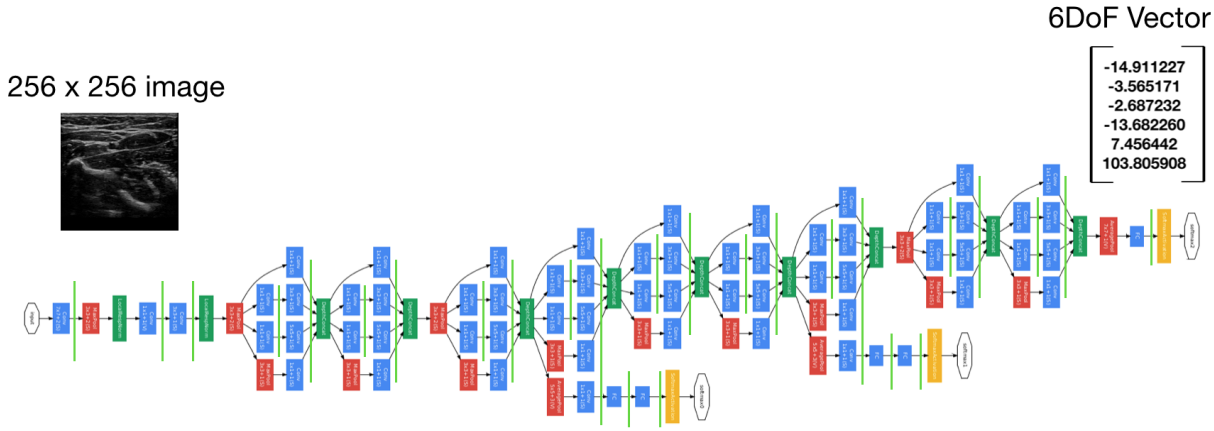



Figure 3-2: Diagram of added batch normalization layers on top of the original GoogLeNet. The bright green lines represent the points of the network where batch normalization was added. These are after each of the convolution layers (blue nodes).

normalization addresses the problem of internal covariate shift, in which the range of inputs changes with each training iteration as a result of changing network parameters. The normalization is done for each training mini-batch[15].

### 3.2.1 Result and Shortcomings

Both angle and position accuracy performance of the network improved with the addition of batch normalization. The second iteration yielded a mean Euclidean loss of 4.02cm, a mean Euclidean angle loss of 13.19°, percent accuracy within 1 cm of 3.47%, percent accuracy within 2 cm of 16.34%, percent angle accuracy within 10 degrees of 49.75%.

The resulting network ended up being around 2x more accuracte in terms of position and around 1.5x more accurate in rotation. All future iterations used batch normalization, as it noticeably improved the network's performance. However, the positional accuracy was still very low. The network's performance could be due to sub optimal network architecture and parameters, but the main challenge faced by the Ultrasound CNN is that the dataset is low resolution and rather uniform. For future iterations we experiment with different optimizations with the network, but the 6DoF unconstrained forearm dataset continues to be a major challenge with the lack of spatial uniqueness and defining features.

The next step for optimization was to try different loss functions. L2-norm loss is a very

standard loss function, which is optimal if there is a constraint with computing resources and if the data has outliers. The ultrasound data is constrained within a set area on the body, so there is no expectation for outliers. In this case, experimenting with the loss function is ideal to optimize the performance of the network.

## 3.3 Scaling the Loss Function

Experimenting with the loss function is critical for optimal network performance. In the PoseNet paper, Kendall *et al.* added a single scaling factor for the rotation values of their position vector. This is intuitive as position and rotation have completely different ranges and units, and the loss function might give more weight to one over the other. Instead of a single scaling factor for the rotational values, a scaling factor was calculated for each element of the vector. Our loss function thus looks like:

$$S = \sum_{i=1}^{n} \frac{1}{\text{range}(y_i)} (y_i - f(x_i))^2$$

The scaling factor of each element is equal to one over the range of that element. The range was calculated by parsing the training set ground truth file prior to training the network.

### 3.3.1 Result and Shortcomings

Both angle and position accuracy performance of the network unexpectedly decreased when the loss function was scaled. The third iteration yielded a mean Euclidean loss of 5.13cm, a mean Euclidean angle loss of 14.06°, percent accuracy within 1 cm of 0.53%, percent accuracy within 2 cm of 6.90%, percent angle accuracy within 10 degrees of 44.83%, and a combined percent accuracy within 1cm and 10 degrees of 0.28%.

The results for the scaled loss function were very poor and unexpected, so the network was retrained twice to verify these results, once with the same hyperparameters, and once with different hyperparameters. However, there was little change in the results. It was concluded that scaling the loss function was detrimental to the network's performance. The

loss function dramatically changed the network's performance, so further experimentation was needed if a similarly dramatic increase in performance could be reached.

## 3.4 Using L1 Loss

The next iteration used L1-norm least absolute deviations loss for the loss function. While not as sensitive as the L2-norm, L1-norm is more robust, meaning that outliers do not have as much of an affect on the regression solution as the L2-norm. L1-norm has similar form to L2-norm except we take the absolute value of the differences instead of squaring them.

L1-norm loss is give by:

$$S = \sum_{i=1}^{n} |y_i - f(x_i)|$$

where $S$ is the loss, $y_i$ is the ground truth pose vector of the $i$th sample, and $f(x_i)$ is the CNN output of the pose vector of the $i$th sample.[5]

### 3.4.1 Result and Shortcomings

Angle and position accuracy dramatically improved with the implementation of the L1-norm loss function. This iteration yielded a mean Euclidean loss of 4.00cm, a mean Euclidean angle loss of 14.42°, percent accuracy within 1 cm of 10.65%, percent accuracy within 2 cm of 34.75%, percent angle accuracy within 10 degrees of 50.92%, and a combined percent accuracy within 1cm and 10 degrees of 9.20%.

Compared to the previous iterations, these results were a great improvement in terms of positional accuracy. While angle accuracy slightly improved from the previous best (using batch normalization and unscaled L2 loss), the positional accuracy more than doubled from the previous best. One hypothesis as to why this may be is because L1 loss is more robust, and can thus generalize better to the dataset. The image quality is very poor, and a more general loss function would be needed.

### 3.4.2 L1 Loss with a Scaled Loss Function

Even though scaling the loss function in the previous iteration actually resulted in worse performance, one iteration of the network was trained with a scaled L1-norm loss function, as it would in theory improve the performance of our network.

Similar to the scaled L2-norm loss function, each absolute difference was multiplied by a scaling factor based on the vector element range. The scaled L1-norm function is given by:

$$S = \sum_{i=1}^{n} \frac{1}{\text{range}(y_i)} |y_i - f(x_i)|$$

The results of this loss function were more or less the same as the results of unscaled L1-norm loss. Both angle and position accuracy performance of the network slightly went down when the loss function was scaled. This iteration yielded a mean Euclidean loss of 4.09cm, a mean Euclidean angle loss of 15.63°, percent accuracy within 1 cm of 10.65%, percent accuracy within 2 cm of 35.56%, percent angle accuracy within 10 degrees of 45.08%, and a combined percent accuracy within 1cm and 10 degrees of 8.00%. This further confirmed that the scaled function did not contribute significantly to the network, and was thus not used in future iterations of the network.

## 3.5 Increasing Input Size

Ultrasound images are by nature very low quality. Scaling the ultrasound images from a 550px square image to 256px square image might have removed information from an image where this is not a lot of information. Therefore, an increased input size from `256x256` to `512x512` increases the amount of information going into the network. This increase did not require changes to the neural network as the 512px images are eventually downsampled to the same size as the 256px images when going through GoogLeNet.

### 3.5.1 Results using L2 Loss

Angle and position accuracy increased from the original pass of network using unscaled L2-norm loss. However, the network did not do as well as the pass with L1-norm loss on size

256px images. This iteration yielded a mean Euclidean loss of 4.16cm, a mean Euclidean angle loss of 12.72°, percent accuracy within 1 cm of 5.13%, percent accuracy within 2 cm of 28.13%, percent angle accuracy within 10 degrees of 52.87%, and a combined percent accuracy within 1cm and 10 degrees of 4.74%.

From these results, using the 512x512 image looked like a promising improvement for the network. Intuitively, the accuracy increased due to the increase of information going into the network. The next step was to combine the larger image input with L1-norm loss.

### 3.5.2 Results using L1 Loss

Both angle and position accuracy performance of the network unexpectedly decreased when using the 512px image input with L1-norm loss. This iteration yielded a mean Euclidean loss of 5.49cm, a mean Euclidean angle loss of 23.57°, percent accuracy within 1 cm of 2.37%, percent accuracy within 2 cm of 8.92%, percent angle accuracy within 10 degrees of 25.80%, and a combined percent accuracy within 1cm and 10 degrees of 1.10%.

This result was unexpectedly poor. Unscaled L1-norm loss yielded the best performace so far by a wide margin, and using $512x512$ images improved performance significantly on the L2-norm loss model. However, using L1-norm and $512x512$ yielded very poor results, doing only slightly better than the baseline and the scaled L2-norm model, and performing worse by far to the other iterations.

## 3.6 Best Model

The best model from the multiple iterations of the network is the GoogLeNet using batch norm and unscaled L1-norm loss on $256x256$ images. This model performed slightly better than same model with scaled L1 loss, and outperforms the other models by far in terms of positional accuracy and rotational accuracy. The model itself was one of the earlier iterations and still remains the model with the best accuracy by a large margin.

## 3.7 Analysis

From our results, L1-norm loss significantly performs better than L2-norm loss when using batch normalization and 256x256 image size. These results were unexpected and the final architecture ended up being surprisingly simple.

L1 norm may have suited this problem better than L2 norm as the sparsity property of the L1 norm was ideal for this problem, and to a lesser extent the robustness property. The data itself does not have many outliers, since all of the data is contained within a small volume of 3D space (the volume of the body part that was scanned). The sparsity property of L1-norm is valuable as it helps in feature selection. Feature selection is a further-involved form of sparsity: instead of shrinking coefficients near to 0, feature selection is taking them to exactly 0, and hence excluding certain features from the model entirely. In a model where features are less defined, having a stricter feature selection property is important to filter out less important features that may affect the network regression.

The surprising result that we reached was that using the 512x512 image input with L1-norm loss actually lowered accuracy dramatically. This was surprising because, intuitively, using a bigger image would mean having more information as input, which would lead to better regression in the network. This is possibly a result of the Hughes effect, also known as the Hughes phenomenon or the curse of dimensionality[1], which states that as the dimensionality of the problem increases (in this case, the size of the input), the volume of the feature dimension space increases so fast that data actually becomes sparse[11]. To combat this problem, an incredibly large amount of training data is needed to cover the possibilities of the input space. This is very difficult with our limited amount of data. As a result, increasing the dimensionality of our problem decreases the predictive power of the network, which gives a possible explanation as to why the network performed so poorly on the 512x512 images. It seems like increasing the input size was only detrimental to the result

[1]The Hughes effect describe a series of phenomena in highly dimensional feature space. The curse of dimensionality refers to the problems that arise with these phenomena, including the exponential need for more data when the dimensionality increases. Complementary to the curse of dimensionality is the blessing of dimensionality. This is less widely noted but includes the concentration of measure phenomenon, in which certain random fluctuations are very well controlled in high dimensions and the success of asymptotic method. These are used widely in mathematical statistics and statistical physics, and suggest that statements about very high-dimensional settings may be made where moderate dimensions would be too complicated.[11]

of the network using L1 loss, and actually increased the accuracy of the network using L2 loss. This is an interesting problem, and it seems that the combination of the loss function and input size can greatly affect the resulting performance of the network. This may be because L2 loss is much more stable than L1 loss, and as such the loss scales better with higher dimensions. However, working with higher dimensions overall did not seem to significantly enhance performance, the input size was scaled back to 256x256 pixels. The final network performed much better than the baseline, which was the PoseNet implementation, but the final performance was still rather poor in terms of total accuracy. The following chapter will discuss the network's performance on different datasets, and the last chapter will discuss these results, the persistent challenges encountered during this project and future work to overcome the aforementioned challenges and further optimize the network.

# Chapter 4

# Performance on Different Datasets

The ultrasound data collected on the forearm was relatively uniform, as there are not many features within forearm ultrasounds to track. Two more datasets were collected to see how the network would perform on data with more features and data with constrained output. One dataset was taken of an ultrasound phantom, with features that are clearly defined and mapped out in the phantom. The second dataset was taken of the same forearm as the original data, except movement was constrained to only one direction along the forearm, with no rotation.

## 4.1 Ultrasound Phantom

### 4.1.1 Dataset

The network was trained on ultrasound images taken from an ultrasound phantom to observe how the network performed on images with clearly defined features. The dataset was taken from a CIRS General Purpose Ultrasound Phantom Model 054GS.[3] Figure 4-1 shows a diagram of the layout of features within the phantom, as well as an ultrasound image showing some of the features. Each feature (e.g. dot, cyst) is clearly defined in the ultrasound layout and the features are positioned in a clear unique pattern. For this data collection, there was minimal variation in rotation as the phantom is flat and rigid, thus making it difficult to maintain contact if the ultrasound probe is rotated away from the phantom surface. Data

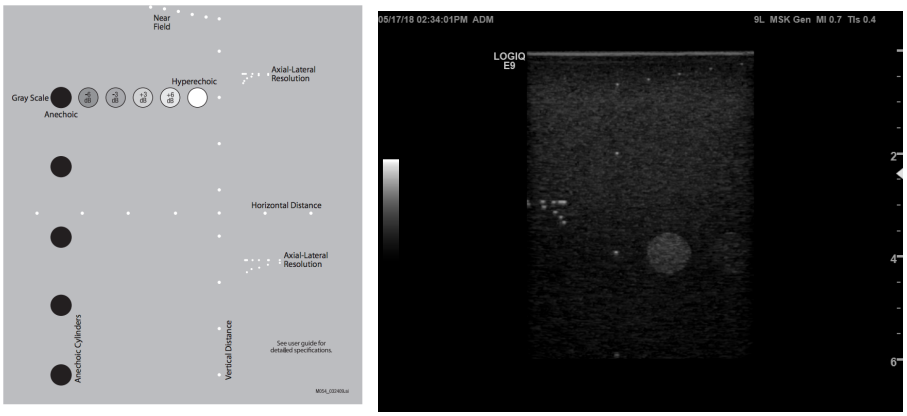



Figure 4-1: (Left) Diagram of the layout of features within the ultrasound phantom.[3] (Right) Ultrasound image of the phantom. Features shown in the layout diagram can be seen in the ultrasound.

was collected with different probe orientations perpendicular to the surface of the phantom, and there was minimal fanning. There were 4059 images taken from the phantom, and this data was split into 70% for training and 30% for testing. There was less freedom of movement with the phantom, and the surface of the phantom is relatively small (17.8cm x 12.7cm), so less data was needed to take images that encompassed the volume of the phantom.

### 4.1.2 Results

As shown in Table 4.1, the Ultrasound CNN model achieved very high positional accuracy, most likely due to the defined features in the dataset. The rotational accuracy is less than expected given that there was minimal rotation within the dataset, but the accuracy is still rather high. Having multiple defined features drastically increases the performance of the network, which is expected as convolutional neural networks optimize on features. However, a dataset with well defined features will not always be available in ultrasound data collected on patients, so the Ultrasound CNN model will need to be improved to handle ultrasound images with less structure. These results also imply that the rotational accuracy of the network is still relatively poor. The data set contained very defined features, yet rotational accuracy is still less than 80%. One focus for future work in the network is to improve the network's prediction for the probe's rotation. Improving rotation might also boost accuracy for the probe position.

Table 4.1: Performance Accuracies of Ultrasound CNN on Ultrasound Phantom

| Average Distance Loss (cm) | Average Angle Loss (degrees) | % Accuracy within 1cm | % Accuracy within 2cm | % Accuracy within 10° | % Accuracy within 1cm and 10° |
|---|---|---|---|---|---|
| 0.16[1] | 15.40[2] | 100.00% | 100.00% | 79.00% | 79.00% |

## 4.2 Constrained Movement Dataset

### 4.2.1 Dataset

Another set of ultrasound images was taken of a forearm, except movement was constrained to one axis along the forearm. There is no rotational movement in this dataset, except for small fluctuations when moving the ultrasound probe (the dataset was taken manually with the probe). When constraining movement to one direction along the arm, the layers of muscle and veins create enough features that the Ultrasound CNN model can track. Without the ambiguity given by multiple different orientations of the probe, the problem of identifying the probe's position becomes a lot easier. There were 1686 images taken from the phantom, and this data was split into 70% for training and 30% for testing. The probe's movement in this data was severely constrained, so there was a lot less unique data to encompass the forearm.

### 4.2.2 Results

The model had very high positional and rotational accuracy on the constrained forearm dataset, as shown in Table 4.2. This was a significant improvement to the results of the model on the original dataset, which was collected on the same forearm. The model predicts the probe position with significantly higher accuracy when there is no variation in rotation. Rotating the probe can introduce many ambiguities, because the subject can have many features from one angle and be completely uniform from another angle. With the forearm,

[1]Standard Deviation: 0.027
[2]Standard Deviation: 24.290

the images are sometimes completely uniform when taking longitudinal cross section images, so the ultrasounds would look the same even taken at different locations. By constraining rotation and only having the model learn on position, the model becomes very accurate in predicition position.

Table 4.2: Performance Accuracies of Ultrasound CNN on Constrained Forearm Data

| Average Distance Loss (cm) | Average Angle Loss (degrees) | % Accuracy within 1cm | % Accuracy within 2cm | % Accuracy within 10° | % Accuracy within 1cm and 10° |
|---|---|---|---|---|---|
| 0.45[3] | 6.90[4] | 97.70% | 97.70% | 95.40% | 95.40% |

### 4.2.3 Analysis

The Ultrasound CNN model yielded very high positional accuracy when trained on datasets with defined features and limited rotational movement. Having defined features such as dark cysts and bright dots can give the network clear information about the probe's location. Constraining rotation also limits the amount of ambiguous data the network is exposed to during training. Depending on the orientation of the probe, the ultrasound images can be completely uniform and identical at different locations. In practice, however, there might not be the luxury of being able to constrain the data or having clear features in the images. For future iterations of the network, it is important to optimize performance on more ambiguous and uniform datasets so that the network can be flexible with imperfect data.

[3]Standard Deviation: 0.293
[4]Standard Deviation: 14.822

# Chapter 5

# Discussion and Future Work

## 5.1 Discussion and Challenges

The final network architecture of Ultrasound CNN is an altered version of Kendall *et al.* 's PoseNet with batch normalization and L1 loss. This network performed much better than the baseline, which was the original PoseNet implementation. However, the final performance was still rather poor on the forearm data with all six degrees of freedom. The network resulted in a percent accuracy within 1 cm of 10.65% (34.75% within 2 cm), and percent angle accuracy within 10 degrees of 50.92%. In order for this network to be effective in practice it will need to perform at much higher accuracy.

When trained on highly defined data and constrained data, the network performed at a very high positional and rotational accuracy. For the phantom data, the network achieved 100% positional accuracy on the test set and 79% rotational accuracy, which is a great improvement to the previous less defined dataset. On the constrained forearm dataset, the model achieved 97.70% positional accuracy and 95.40% rotational accuracy. This dataset was collected from the same subject as the original forearm dataset, but the movement was severely constrained, which resulted in a much higher performance. Training the network on different datasets stresses the importance of how training data impacts the performance of the network. However, when doing data collection on actual patients, there may not be the luxury of having clear or constrained data. If this model were to be used in practice, then collecting data with constrained rotation will be very important.

The problem of doing any sort of regression on ultrasound images is that the images are by nature of very poor quality, so it would be difficult to isolate and track features on the images. As a result, many of the images with very different pose vectors end up look very similar, and it might be challenging for a simple CNN to effectively regress the pose vector.

Other issues that are common to machine learning problems is being able to optimize the network within reasonable computing resources. Throughout the project, there were difficulties procuring cost effective GPU computing power to train the network on. Even though GoogLeNet uses inception modules to reduce the total number of parameters needed for different convolutions, it is still a highly parameterized deep network. The network thus took very long to train on a normal GPU setup. The AWS instance trained 20-30 epochs in one day for 256x256 pixel input size images. However, these instances can quickly become expensive, so further computing optimizations will be needed for this network.

The following section describes future optimizations and directions this project can take if this project were to continue.

## 5.2 Future Work

### 5.2.1 Using Images with More Structure

One suggestion that came about in meetings was to use ultrasounds from a different body part with more structure. The forearm was originally selected for convenience and easy control of probe movement, but there were suggestions of imaging the throat, leg, or liver, due to there being more structure and thus more features. It is very possible to get different network performance on different body parts, and which may further give a proof of concept that the Ultrasound CNN can regress the 6DoF pose vector. There is proof from the phantom data results that the network does perform very well with more defined data, so it would be of interest to observe how network performance changes with different body parts.

### 5.2.2 Multiple-Image Input

In practice, when doctors take ultrasounds they would fan the ultrasound probe or compress the body part with the probe to give a better idea of what they were looking at. This would give them multiple images to better estimate where the ultrasounds are in the patient's body. Following this practice, another idea is to take multiple images as input to the network, either of sweeps from the same position on the body part or of images taken at different compression levels. The first option gives more information on surrounding tissue and the second gives information on tissue softness.

Using multiple image input was actually attempted at the later stages of this project, but was not completed due to time constraints and limited access to computing resources.

### 5.2.3 Classification Problem

There have been other applications of turning regression tasks into classification tasks in machine learning by making the output space discrete instead of continuous. This can be done by putting the output space into "bins" and classifying each sample to a single bin. There are a number of benefits to doing classification as opposed to regression:

- There are powerful loss functions that can be used on the fully connected layer's probabilities for each class, which include cross entropy loss and hinge loss.
- The output space is severely decreased and constrained to a finite number of discrete classes
- Can train well with limited data
- Easier measurement for accuracy

Of course, there are disadvantages to putting the output space into bins, including losing a lot of information in the predictors. If this model were to be turned into a classification problem instead of regression problem, more research would need to be done on the use case of this network and how much information can afford to be lost.

### 5.2.4 3D Reconstruction

Kendall *et al.* expanded upon their PoseNet implementation to use 3D reconstruction, and were able to reconstruct the streets of Cambridge, UK using the image data they received.[16] This would be a great tool to visualize the inner structure of the scanned subject, and how well the network can predict the probe's location.

## 5.3 Conclusion

Although the model performed better than the baseline of PoseNet in terms of accuracy, but there is still a lot of work to be done to refine the network enough to use in practice. The network performed very well when the dataset was "cleaned": the data contained clear features, and the output space was constrained so there was little probe rotation. However, the network still performed very poorly on more ambiguous data, stressing the importance of the constraints training data. If the network were to be used in practice as is, there would need to be several restrictions on the training data. There are different routes this project can take in terms of improving network performance, including scanning different body parts to see difference in performance, turning this regression problem into a classification problem, and using multiple images as input. Using machine learning to learn poses on ultrasound probes is a novel idea, and we believe we are slowly breaking into the field with the advances in this project.

# Appendix A

# Code

The code for this project can be found at github.mit.edu/eyxue/ultrasound-cnn. This repository includes all of the models mentioned in this paper, as well as scripts for data processing and data analysis.

# Bibliography


[1] Amazon ec2 p2 instances. https://aws.amazon.com/ec2/instance-types/p2/. Accessed: 2018-04-20.

[2] Cs231n: Convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks/. Accessed: 2018-04-20.

[3] General purpose ultrasound phantom. http://www.cirsinc.com/products/all/80/general-purpose-ultrasound-phantom/. Accessed: 2018-05-20.

[4] Imagenet large scale visual recognition competition (ilsvrc). http://www.image-net.org/challenges/LSVRC/. Accessed: 2018-04-30.

[5] L1 and l2 as loss function and regularization. http://www.chioka.in/. Accessed: 2018-04-30.

[6] OpenCV open source computer vision. http://opencv.org/. Accessed: 2018-04-20.

[7] Optitrack - optical motion tracking. http://optitrack.com/. Accessed: 2018-04-20.

[8] Running tensorflow with gpu support on aws. http://mortada.net/. Accessed: 2018-04-20.

[9] Tensorflow: Api documentation. https://www.tensorflow.org/api_docs/. Accessed: 2018-04-20.

[10] Ultrasound images of diseases of the liver. http://www.ultrasound-images.com/liver/. Accessed: 2018-04-20.

[11] María Alonso, José A Malpica, and Alex Martinez-Agirre. Consequences of the hughes phenomenon on some classification techniques. 05 2011.

[12] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian D. Reid, and John J. Leonard. Simultaneous localization and mapping: Present, future, and the robust-perception age. *CoRR*, abs/1606.05830, 2016.

[13] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *CoRR*, abs/1606.03798, 2016.

[14] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[16] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Convolutional networks for real-time 6-dof camera relocalization. *CoRR*, abs/1505.07427, 2015.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[18] Pencilla Lang, Petar Seslija, Damiaan F. Habets, Michael W. A. Chu, David W. Holdsworth, and Terry M. Peters. Three-dimensional ultrasound probe pose estimation from single-perspective x-rays for image-guided interventions. In Hongen Liao, P. J. "Eddie" Edwards, Xiaochuan Pan, Yong Fan, and Guang-Zhong Yang, editors, *Medical Imaging and Augmented Reality*, pages 344–352, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[20] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Relative camera pose estimation using convolutional neural networks. *CoRR*, abs/1702.01381, 2017.

[21] Pagoulatos N, Edwards WS, Haynor DR, and Kim Y. Interactive 3-d registration of ultrasound and magnetic resonance images based on a magnetic position sensor. *IEEE Trans. Information technology in biomedicine*, 1999.

[22] A. Ourahmoune, S. Larabi, C. Hamitouche-Djabou, S. Idri, M. Belhadj, and M. Benallegue. Kinect-based ultrasound probe pose estimation to build an affordable knee ultrasound learning database. In *2015 8th International Conference on Biomedical Engineering and Informatics (BMEI)*, pages 489–494, Oct 2015.

[23] Onur Özyesil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey on structure from motion. *CoRR*, abs/1701.08493, 2017.

[24] Bronislav Pribyl, Pavel Zemcík, and Martin Cadík. Pose estimation from line correspondences using direct linear transformation. *CoRR*, abs/1608.06891, 2016.

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[26] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312, May 2016.

[27] J. W. Trobaugh, W. D. Richard, K. R. Smith, and R. D. Bucholz. Frameless stereotactic ultrasonography: Method and applications. *Computerized Medical Imaging and Graphics*, 1994.

[28] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. *CoRR*, abs/1612.02401, 2016.

[29] Bo Zheng, Ryo Ishikawa, Takeshi Oishi, Jun Takamatsu, and Katsushi Ikeuchi. 6-dof pose estimation from single ultrasound image using 3d ip models. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, June 2008.

[30] Bo Zheng, Ryo Ishikawa, Jun Takamatsu, Takeshi Oishi, and Katsushi Ikeuchi. A coarse-to-fine ip-driven registration for pose estimation from single ultrasound image. *Computer Vision and Image Understanding*, 117(12):1647 – 1658, 2013.