Convolutional Neural Networks for Left Ventricle Volume Estimation

Carol Hsin
Stanford University
cshsin [at] stanford.edu

Cheryl Danner
Stanford University
cdanner [at] stanford.edu

Abstract

In this study, we experimented with multiple approaches of using convolutional neural networks (CNNs) to determine the systole and diastole volumes of the left ventricle (LV) given input cardiac CINE MRIs mapped to volumes. The methods explored involved changing the inputs and outputs of the CNNs in four main approaches: 2D segmentation, 2D centroid coordinate and area regression, volume backpropagation, and 3D volume regression. In all our methods, we experimented with different CNN architectures and automated hyperparameter tuning. We also experimented with image processing and other techniques to standardize the inputs in order to improve accuracy while decreasing noise. Our results show that predicting the volumes directly from the Kaggle CINE MRIs produced lower relative errors than the methods relying on truth contours from a smaller study and that preprocessing the images to highlight important regions based on cardiac knowledge also improved our relative errors.

1. Introduction

Magnetic resonance imaging (MRI) has revolutionized the modern medical diagnostic process, and one of its many uses now is examining a patient's heart. The systolic and diastolic volumes of the left ventricle (LV) are clinically important, e.g. enlarged LV cavities may indicate cardiovascular disease. The usual process of analyzing these MRI results requires that trained personnel manually analyze the images, which can take up to 20 minutes to complete [11]. Automation would speed up anomaly detection and allow cardiologists more time with their patients.

Our project applies convolutional neural networks to the task of automatically calculating systolic and diastolic LV volumes. The input for each sample is four-dimensional – a stack of MRI slices is recorded at several time points, creating a picture of the entire heart volume as it goes through a cardiac cycle. We experimented with several ways of using CNNs to predict the LV volumes at systole and diastole.

2. Background/Related Work

A review of cardiac MRI segmentation methods in [11] breaks approaches from 1993 to 2010 into the following categories and common methods: 1) image-based, using theresholding and dynamic programming; 2) pixel classification, using Gaussian mixture models and clustering algorithms; 3) deformable parts models, including active contours and registration methods, 4) active shape and appearance models; and 5) atlas-guided methods. Outside of the pixel classificaiton cateogry, most of the approaches reviewed either require strong prior information or user interaction. Papers from the Medical Image Computing and Computer Assisted Interventions (MICCAI) 2009 LV Segmentation Challenge are in line with these methods. For example, [9] uses thresholding and comparison to a template mask, and [10] uses mathematical morphology.

Segmentation methods using neural networks, while nearly absent from [11] have grown in popularity recently, both for general image segmentation [8] and within the medical imaging field. In [3], deep neural networks are used for LV segmentation in ultrasound images, and [6] uses CNNs to predict a bounding box containing the LV.

3. Data Exploration and Understanding

3.1. Kaggle Dataset

Kaggle provides a dataset of 2D magnetic resonance images (MRIs) in DICOM format. The training set consists of MRIs from 500 patients and their associated systole and diastole volumes. Each patient's data consists of timeseries MRIs of short-axis (SAX) slices, or cross-sections, from the base to the apex of the heart. Figure 1 shows two sample images from the set of 480 images that comprise study 1, and Table 1 shows sample labels for a few studies.

Each slice consists of time frames across the cardiac cycle (one heartbeat, from start of systole to end of diastole) during one breath hold from the patient, e.g. if there are 12 slices, there were 12 separate breath holds. While there are usually about 12 slices per study and 30 time frames per slice, the data varies widely in both the number of slices per study (1 in study 499 to 22 in study 436) and the number of

Figure 1. Example of systole (left) and diastole (right) from Kaggle study 1. The LV is the light area bordered by dark grey at the center of each image, smaller at systole and larger with a thinner border at diastole.

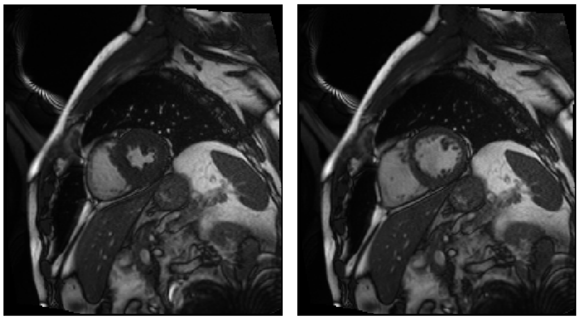
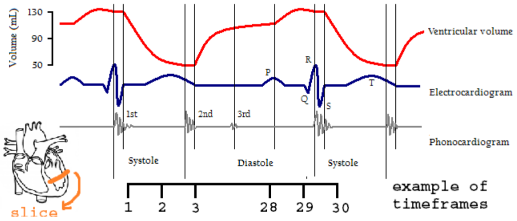


Table 1. Example labels for Kaggle training set give LV systole and diastole volumes in mililiters (mL).

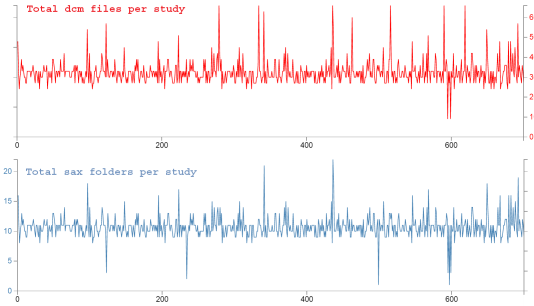
Id	Systole	Diastole
1	108.3	246.7
2	54.6	137.2
3	32.7	99.3
⋮	⋮	⋮
500	33.7	102.0

Figure 2. Our illustration describing the time frames in relation to the cardiac cycle and within a particular slice as explained on the bottom left. Here systole is frame 3 and diastole, 29.



images per slice (22 in study 416 to 330 in study 334).

Figure 3. Illustrating the variance in the Kaggle dataset. Studies 1 to 500 make up the training set, and studies 501 to 700 make up the validation set.



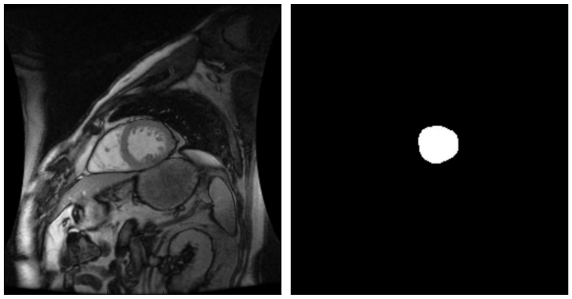
In addition to the dimensional variance illustrated in Figure 3, images across studies differed in size (height and width in pixels), pixel resolution, and slice thickness. Furthermore, the left ventricle had varying levels of brightness and contrast. All of these factors increased the difficulty of the learning process overall and the fundamental ability to present a consistently sized input to a network. These challenges are discussed further in Section 4.6.

3.2. Sunnybrook Dataset

While the Kaggle training data provides the truth values for the systolic and diastolic volumes, there is no ground truth labeling for LV segmentation. Therefore, in our segmentation experiments, we supplemented the Kaggle data with the 'Sunnybrook' dataset. The Sunnybrook data includes 45 sets of MRI images and 'truth' contours used for the MICCAI LV Segmentation Challenge held in 2009 [12].

A condition of this Kaggle challenge stipulates that, with the exception of the Sunnybrook data, no additional datasets may be used to train models. Therefore, we did not explore using pretrained models or any other datasets beyond the Kaggle and Sunnybrook datasets.

Figure 4. Example of image and label from Sunnybrook dataset



4. Methods and Approaches

Our methods, approaches, and overall view of the task evolved with each experiment. Results from each experiment led to ideas for new experiments. Thus, we ended up with about four main approaches, each building on knowledge derived from predecessors.

- Methods using truth contours
  - Segmentation
  - Coordinate and area regression
- Methods using volume labels
  - Backpropagation through volume calculation
  - Volume regression

The following experiments were implemented using Lasagne [5] and Keras [4] on top of Theano [1, 2].

4.1. Segmentation (2D inputs)

Our first approach to the task was to view it as a segmentation problem, in which we would predict the LV area based on per-pixel classification predictions and then use numerical integration to get the volumes. Since the Kaggle dataset did not provide truth labels for the LV location, we trained our segmentation model on the Sunnybrook dataset using a convolutional network with upsampling layers.

4.1.1 Network architecture and experimentation

The network features many layers of small convolutional filters with batch normalization and max pooling layers inserted at intervals. ReLu nonlinearities and Xavier initialization are used throughout. The output of the convolutional layers, now with much smaller spatial dimensions, is passed through a dense layer before the convolution and pooling operations are inverted to return to the original input size. Finally, the logistic function is applied element-wise to get each pixel's LV probability. Loss is calculated as the squared error between the predicted probability and the true LV probability provided by the binary mask. The loss L_i for the i^{th} sample, which has n^2 pixels, is shown in equation 1. The variable $y_{i,j}$ represents the label of the j^{th} pixel, $p_{i,j}$ is the predicted pixel LV probability, and $f(X_i)_j$ is the j^{th} pixel of the network output from input image X_i.

$$L_i = \sum_{j=1}^{n^2} (p_{i,j} - y_{i,j})^2, \quad p_{i,j} = \frac{1}{1 + e^{-f(X_i)_j}} \tag{1}$$

An initial proof-of-concept network (net1) had three convolution layers, each with 64 3x3 filters. After initial results were obtained, the network was expanded to net2 with seven convolution layers and then again to net3 with ten. Regularization was not implemented in net1, but small amounts of overfitting were observed in net2 and net3, leading to the addition of a dropout layer prior to the dense layer. Additional dropout layers were tested at points closer to the input to strengthen regularization, but validation loss increased greatly in training, so only the single dropout layer was retained. Results in Section 5 are not reported for intermediate networks net1 and net2.

Figure 5 gives a high-level visual representation of net3, and Table 2 gives the corresponding layer specifications. All input images and filter sizes are square, so only a single spatial dimension is given in the input dimension and receptive field dimension columns.

4.1.2 Systole and diastole determination

After predicting the pixel-wise LV probabilities for the Kaggle images, we rounded the probabilities to get a mask and summed all pixels to get the predicted area for each image. To determine which times corresponded to systole and diastole, we used knowledge of the cardiac cycle and compared the areas within each slice, picking out the minimum as systole and maximum as diastole for that particular slice.

Figure 5. Conceptual depiction of segmenetation network

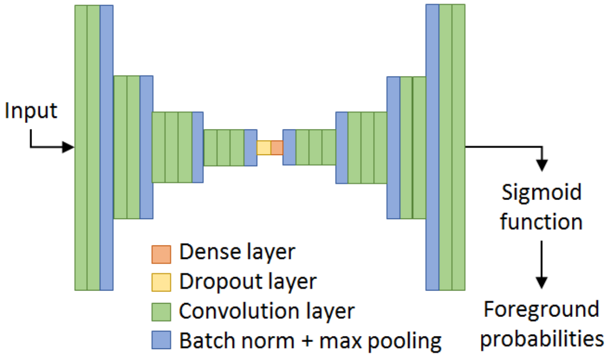


Table 2. Layer specifications for net3 up to the dense layer. Remaining layers invert the operations from Pool4 to Conv1-1.

Layer	Input dim	Field dim	Stride	# filters
Conv1-1	128	3	1	64
Conv1-2	128	3	1	64
Pool1	128	2	2	
Conv2-1	64	3	1	128
Conv2-2	64	3	1	128
Pool2	64	2	2	
Conv3-1	32	3	1	256
Conv3-2	32	1	1	128
Conv3-3	32	3	1	256
Pool3	32	2	2	
Conv4-1	16	3	1	256
Conv4-2	16	1	1	128
Conv4-3	16	3	1	64
Pool4	16	2	2	
Dropout	8			
Dense	8	8	1	4096

4.1.3 Volume calculation

The resulting systole and diastole areas are arranged by slice, ordered from base to apex of the heart. For each study, we used the DICOM metadata fields of the first image to get the pixel spatial values and the slice thickness. We used those values and the area vector from the previous step to calculate the volumes by numerical integration.

While some studies [13] have used a simple disk method in which pairs of areas are used to model a conic disk and the volume is an interpolation by frustum volume formula, we used Simpson's method of integration, which is more accurate. Simpson's method is equivalent to using a quadratic interpolation $P(x)$ for the function $f(x)$ giving the area at each point x. Below are the formulas for calculating the volume between a single pair of slices at adjacent locations a and b using the frustum method (equation 2) and Simpson's method (equation 3).

$$v_{\mathrm{f}} = \tfrac{b-a}{3}(f(a) + \sqrt{f(a)f(b)} + f(b)) \tag{2}$$

$$v_{\mathrm{s}} = \int_a^b P(x)\,\mathrm{d}x = \tfrac{b-a}{6}\left[f(a) + 4f(\tfrac{a+b}{2}) + f(b)\right] \quad (3)$$

4.2. Coordinate and Area Regression (2D input)

The application of a regression model to each image was explored for two purposes: 1) as a method of localizing the LV and 2) as an alternative means of determining LV area. Thus, we developed a model to predict the location of left ventricle centroid (x and y coordinates) and the LV area directly. Truth values for these three items could be derived from truth contours but not from volume labels, restricting viable training data to the Sunnybrook dataset. Area predictions for the Kaggle dataset were used to calculate systole and diastole volumes using the methods described in Section 4.1.3.

Use of the centroid predictions for online region proposals was dismissed due to the added complexity. A set of samples in a minibatch would have different centroid predictions and thus require disassembling the minibatch, cropping each sample, and concatenating to reform the minibatch, all within the network's layers.

4.2.1 Network architecture and experimentation

Modifying the network to perform regression instead of segmentation consisted of replacing the upsampling layers after the dense layer with two hidden dense layers and a final dense layer to produce the three output values. In the loss equation below, $f(X_i)$ is the output of the network for the input image X_i, and y_i is a vector containing true x and y coordinates for the LV centroid in the first two elements and true LV area in the third.

$$L_i = ||f(X_i) - y_i)||_2 \quad (4)$$

Figure 6 gives a visual representation of the modified network architecture, and Table 3 specifies details of the layers added for the regression objective. Dense layers are specified in the style of convolutional layers because they were implemented as convolutional layers, although the network could be equivalently built using true dense layers.

Figure 6. Conceptual depiction of regression network

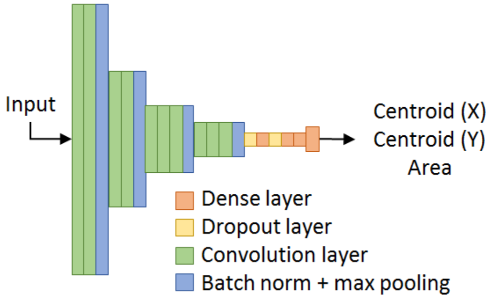


Table 3. Layer specifications for layers following the dense layer in net3. Prior layers are specified in Table 2.

Layer	Input dim	Field dim	Stride	# filters
Hidden1	1	1	1	1024
Dropout	1			
Hidden2	1	1	1	1024
Output	1	1	1	3

4.3. Backpropagation through Volume Calculation (2D input)

A major drawback of the methods described in Section 4.1 and 4.2 is that because the necessary truth label for each image is only provided with the Sunnybrook dataset, the Kaggle training data cannot be used for training. Given the relative sizes of the two sets (285 images for Sunnybrook to over 160,000 images for Kaggle training data), it is clear that finding a way to use the Kaggle training data could greatly improve performance.

Our approach was to embed the volume calculation into the computational graph of the network, allowing backpropagation through the numerical integration method to the parameters in a network from Sections 4.1 or 4.2. This calculation is feasible because the maximum number of slices per study is 22, with most studies containing fewer than 16. Equation 2 was implemented as a Theano function applied to the vectors of area values, and the loss function was converted to measure the squared error between the predicted volume and the true volume.

4.4. Volume Regression (3D input)

An alternative way to utilize the Kaggle training data is to adopt an end-to-end deep learning approach and predict volumes directly from sets of input images. In contrast to the previous methods, this method of using volume regression with 3D inputs breaks away from the lower-level information provided in the Sunnybrook dataset and relies solely on the Kaggle training data. Discarding the information from the Sunnybrook dataset is not a concern because the Kaggle dataset is much larger and more diverse.

Since the Kaggle dataset is over 200x bigger than the Sunnybrook dataset, as a sanity check, we sought to overfit and achieve zero loss on a small subset of the data before training on the full dataset. The test network features small convolutional filters and pooling layers ending with a dense layer using root mean squared error as its loss function so we can better interpret the losses. We trained separate models for systole and diastole volume predictions since systole and diastole images have different distinguishing features, each model can optimize for the features specific to their respective heart cycle and make lower error predictions.

Table 4. Layer specifications for net4.

Layer	Input dim	Field dim	Stride	# filters
Conv1-1	128	3	1	64
Conv1-2	128	3	1	64
Pool1	128	2	2	
Conv2-1	64	3	1	96
Conv2-2	64	3	1	96
Pool2	64	2	2	
Conv3-1	32	3	1	128
Conv3-2	32	1	1	128
Pool2	32	2	2	
Dense	16	16	1	1024
Output	1	1	1	1

4.5. Hyperparameter Tuning

To select hyperparameters, we developed a mechanism to automatically stop training a model if the loss has not improved within a set number of previous epochs. This feature accelerated the process of testing different learning rates. The Adam update rule was used with Lasagne's default parameters. Networks appeared to be learning well using Xavier weight initializations, so no adjustments were made beyond selecting the learning rate.

4.6. Data Preprocessing Experiments

Working with the full Kaggle training dataset presented computational time and memory issues, illustrating the need to decrease the size of the inputs without losing the most relevant information in the images. We also investigated methods for handling the irregularities discussed in Section 3.1 with image sizing, LV location, and image brightness and contrast [14].

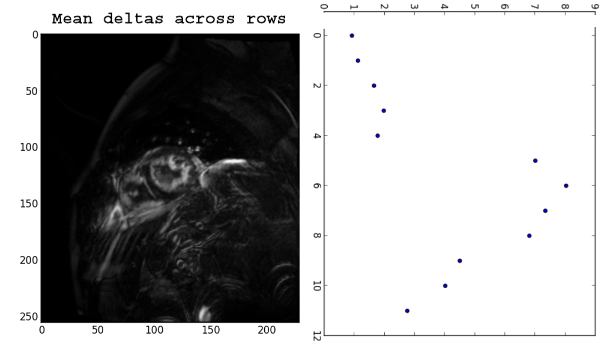
4.6.1 Real world scaling

To achieve a standard scale, images were scaled in both directions by the pixel spacing values (mm per pixel) recorded in the DICOM metadata as described in [14]. Each pixel in the rescaled version represents an area of 1mm × 1mm. Since the average heart is about 9cm × 6cm, we decided the standardized image size would be 128 × 128 pixels, large enough to contain the heart while also being a convenient power of 2 for the CNN. However, just cropping the images from the center point runs the risk of cropping out part or all the LV, which is not guaranteed to be in the center.

4.6.2 Mean deltas for LV localization

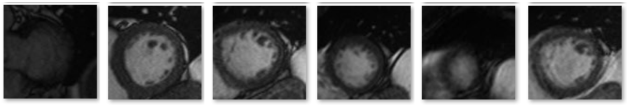
We experimented with multiple approaches to LV localization with the most successful method being localization by the average deltas across the images of a given slice. Since each slice provides a timeseries through the cardiac cycle and the heart is the main source of motion in these images, the difference (delta) between each image from the mean of the timeseries within a slice would be correlated with the location of the LV. While a single delta would not provide much information depending on the cardiac stage it was derived, the average of these deltas is clearly correlated with the LV as seen in Figure 7

Figure 7. Average of the deltas from the mean slice image (left) of the timeframes in a slice and the block pixel row averages (right)



After computing the mean delta image, we located the center (x, y) of the brightest region so we can perform LV localized cropping. To decrease computation time, we averaged pixels per row in steps of 20px, see plot in Figure 7, and took the location of the maximum value as the estimated LV y-coordinate, repeating the procedure with the columns for the estimated LV x-coordinate, resuling in the bounding box $[(x-64, y-64), (x+64, y+64)]$. While the LV might not be perfectly centered, we hypothesized that, given the average heart dimensions, over 60mm in all directions would be enough for most cases. This hypothesis appeared to be valid based on our test images in Figure 8.

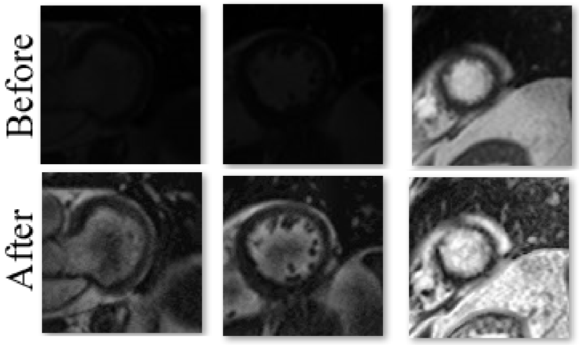
Figure 8. Sample results from our average delta method for LV localized cropping.



4.6.3 Pixel Value Standardization

The top row of images in Figure 9 illustrates the drastic differences in image brightness and contrast within the Kaggle dataset. In many images, the area containing the LV had values so low that the LV was barely visible. Although CNNs are able to learn features with a level of invariance to lighting conditions [7], we expected the large disparities would impede learning. To address the problem, we scaled pixel values to the range 0 to 255 and applied OpenCV's contrast-limited adaptive histogram equalization with example results in the bottom row of Figure 9.

Figure 9. Cropped left ventricle region samples before and after contrast equalization.

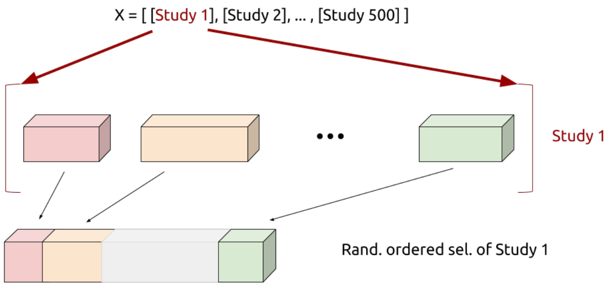


4.7. Handing Dataset Irregularities

Due to the wide variations in number of slices per study and the number of dicom images per slice, we needed a method to standardize the dimensions of the 3D tensors entering the CNNs. In our earlier experiments, our method of dealing with the irregularities was to ignore them by enforcing a channel size of 30 by repeating or ignoring images depending on the number of timeframes in the original slice since the vast majority of slice folders have 30 time frames. Thus, each slice would result in a 3D tensor with a channel size of 30, which would then be mapped to the systole and diastole volumes.

Since our earlier methods did not take into account the variation nor did it utilize the relationship between image slices that are adjacent in time or in space, we came up with several ideas on how to standardize the input 3D tensors. We decided to experiment with standardizing the channel size (e.g. C=30) and on every training iteration, randomly select C images from each of the study's preprocessed slices and then order them by slice location and time in the final 3D tensor. The result is a time and spatially ordered 3D tensor in addition to a regularization from the random samplings, see Figure 10 for illustration.

Figure 10. Illustration of random selection method to handle dataset irregularities


X = [[Study 1], [Study 2], ... , [Study 500]]
...
Study 1
Rand. ordered sel. of Study 1

5. Experiments, Results and Discussion

Average relative error in systole and diastole volume predictions for Kaggle studies 1 through 25 is shown in Table 5, for experiments run using inputs that were preprocessed as described in Section 4.6 ("preprocessed" inputs). Table 6 shows average relative error rates using the original images with minimal preprocessing (referred to as "raw" inputs). The "raw" inputs were zero padded if smaller than 256 pixels or cropped from the center if larger than 256 pixels, then resized by a factor of 0.5 and normalized. Although overall performance on volume prediction is poor in both cases, there is clear improvement using the preprocessed inputs.

Table 5. Average relative error for volume predictions on Kaggle studies 1 through 25 with full image preprocessing.

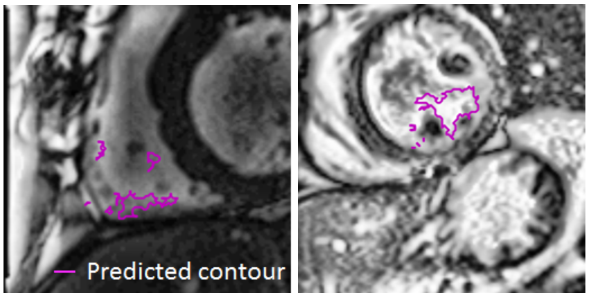
Approach	Architecture	Systole	Diastole
Segmentation	net3	0.53	0.50
Centroid area regr.	net3	0.41	0.52
Volume regression	net4	0.46	0.42

Table 6. Average relative error for volume predictions on Kaggle studies 1 through 25 with minmal preprocessing.

Approach	Architecture	Systole	Diastole
Segmentation	net3	1.17	0.60
Centroid area regr.	net3	1.09	0.55

Examination of examples that contributed most to the error rate identifies ways in which the preprocessing could be improved to achieve better results. Figure 11 shows sample slices from Kaggle studies 6 (left) and 19 (right) with segmentation contour predictions overlaid. In the image from study 6, the LV is off-center, with a significant portion cropped out of the frame. The image from study 19 shows exaggerated noise resulting from the automatic contrast adjustment.

Figure 11. Examples of preprocessed image slices with poor centering (left) and exaggerated noise (right).

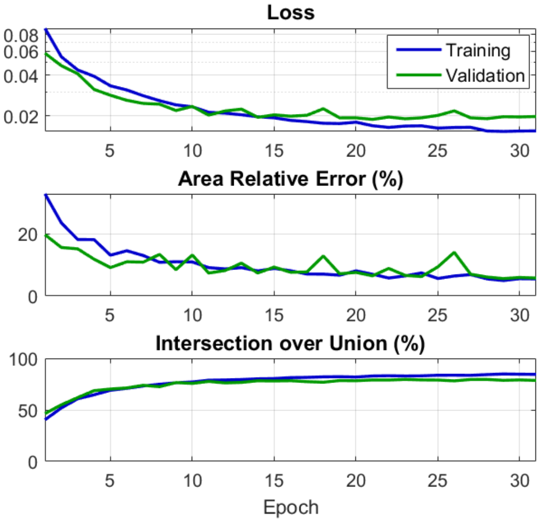


5.1. Segmentation Method

The CNN segmentation model was trained using a 90-10 training-validation split of the Sunnybrook dataset. A round of training and predictions was performed using raw inputs and again using preprocessed images. In both cases, trials with different learning rates showed good performance using 0.00001. Curves for loss, area relative error, and intersection over union for 31 epochs of training on the preprocessed data are shown in Figure 12. Sample prediction and truth contours from the validation set used for the training process are shown in the top row of Figure 14, with sample predictions on the Kaggle dataset in the bottom row. The training history suggests good generalization within the Sunnybrook dataset, with validation relative error and intersection over union at approximately 0.06 and 0.79 respectively. However, the model did not generalize well to the Kaggle dataset, as shown by relative error on volume calculations near 0.5 in Table 5. Example results suggested that the network expected the LV to have little deviation from the center of the image; bright areas near the center were likely to be identified as the LV even when the true LV was off-center. Adding online data augmentation in the form of random shifts of ±20 pixels in vertical and horizontal directions did not improve results.

Compared to the model trained on raw input images, the model trained on preprocessed inputs performed better on the Kaggle dataset. One example of clear improvement in can be seen by comparing the predicted LV contours for the lower right image in Figure 13, where the LV is correctly identified, to the contours for the same case in Figure 14, where the LV is misidentified.

Figure 12. Segmentation training history (preprocessed inputs)



5.2. Coordinate and Area Regression Method

The coordinate and area regression network was trained similarly to the segmentation network, using a 90-10 training-validation split of the Sunnybrook dataset. Trials of different learning rates showed better performance at a slightly higher learning rate of 0.00005.

Although its prediction mechanism is fundamentally different, the coordinate and area regression model had performance very similar to that of the segmentation model. One difference is that this model had a much better relative validation error rate during training on the raw input images (0.13) compared to the preprocessed images (0.61). Results on the first 25 studies in the Kaggle dataset were comparable to results for the segmentation model in Tables 5 and

Figure 13. Segmentation predictions on preprocessed image samples from the Sunnybrook dataset (top), Kaggle study 1 (botom left), and Kaggle study 4 (bottom right).

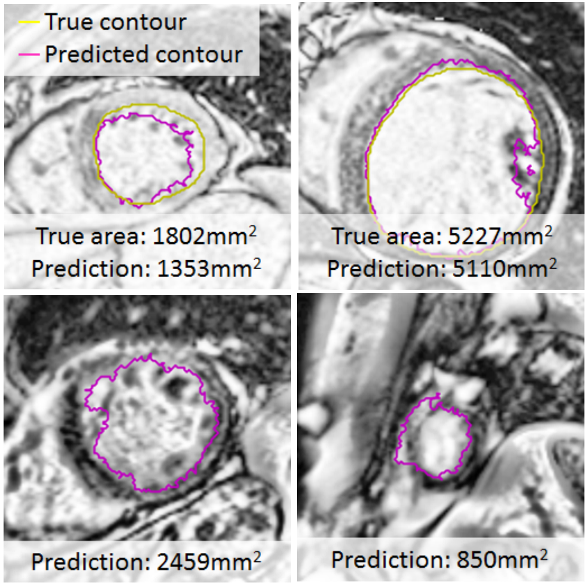
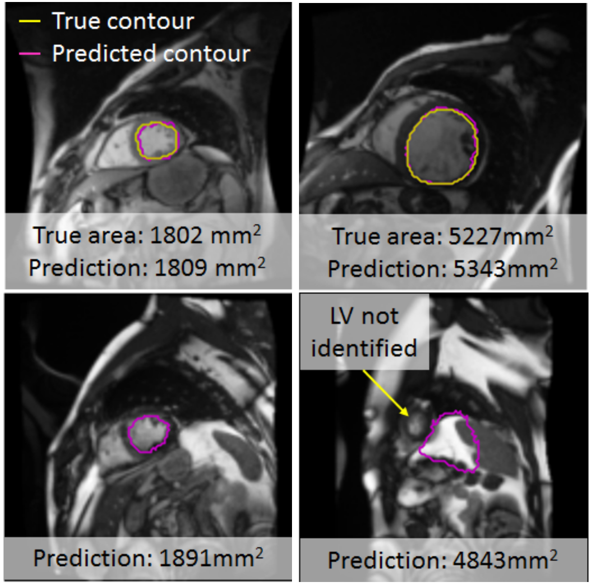


Figure 14. Segmentation predictions on raw inputs from the Sunnybrook dataset (top) and Kaggle dataset (bottom). Samples are analogous to those in Figure 14.



6. Commonality can be seen also in Figure 16 where again the network using raw images misidentifies the LV, but the network using preprocessed inputs correctly locates it.

Figure 15. Centroid and area regression training history (preprocessed inputs)

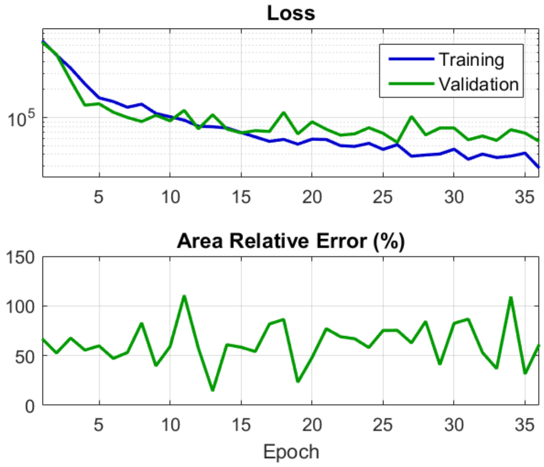
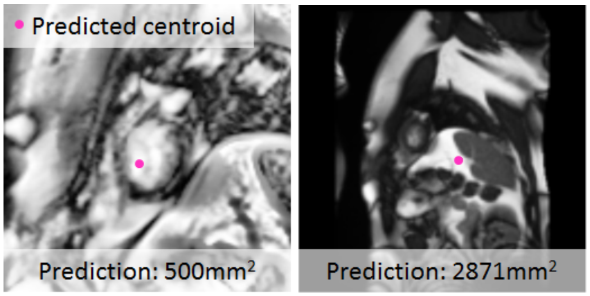


Figure 16. Predicted LV centroid locations and areas for Kaggle study 4 using preprocessed (left) and raw (right) inputs.



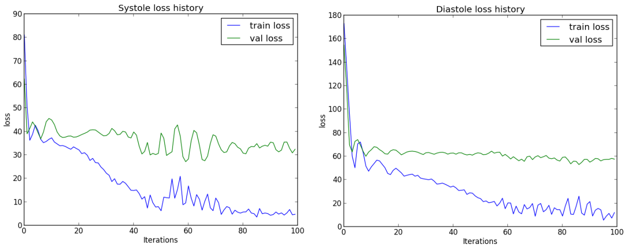
5.3. Backpropagation through Volume Calculation Method

In separate trials, the saved parameters for each of the trained area-prediction networks from Sections 5.1 and 5.2 were used to initialize the network that was modified to incorporate the volume calculation. Throughout many attempts to train the network using various learning rates, the network was not able to learn, as indicated by completely flat training and validation loss curves over the course of 10 epochs on a subset of the Kaggle training data. Due to the lack of training progress, results are not available for this method.

5.4. Volume Regression Method

While there was clear overfitting, we were not able to get near 0 loss on a small subset (studies 1 to 25) of the data. However, this is most likely caused by the method we used to sample the dataset in order to fix the size of the input tensors while accounting for irregularities as described in Section 4.7. Since the random sampling is an inherent feature in this method and is not removable, we believe the loss achieved did pass the sanity check to some extent and we could consider looking at more data.

Figure 17. Training history for systole and diastole volume regression models on data from studies 1 to 25 using method described in Section 4.7



When we tried our preprocessing method on the full dataset, we ran into memory and storage challenges. We were previously storing our data as npy, but we now have a more complicated structure, so we decided to use protobufs to facilitate serialization and deserialization. While this worked with a small dataset of 25 studies, when we tried to scale up, we ran into the protobuf 2 GB limit. It is also a possibility that our data sampling method, which requires holding a copy of the full data in memory along with what is fed into the CNN, would result in memory crashes within the program even if we were able to serialize the data with protobufs. If we had more time, we would experiment with using HDF5 and other storage techniques along with other data accessing techniques.

6. Conclusion/Future Plans

Without access to an extensive set of labeled images, area prediction methods are not viable for predicting LV volumes. Although it would be interesting to examine why backpropagation through the volume equation did not work, the strategy of using 3D inputs to directly predict volumes appears to be more promising. We believe that additional work to implement the volume regression method with random sampling could produce excellent results by reworking the data loading mechanism to avoid memory issues and tuning the network design. Image preprocessing also proved to be a significant factor in performance since we were able to distill the more pertinent information for the CNN. Given more time, we would also experiment with using the LV localized images' deltas directly rather than the images since the deltas would not contain the background noise apparent in the images.

References

[1] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[3] G. Carneiro, J. C. Nascimento, and A. Freitas. Robust left ventricle segmentation from ultrasound data using deep neural networks and efficient search methods. *Biomedical Imaging: From . . .*, pages 8–11, 2010.

[4] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[5] S. Dieleman, J. Schlter, C. Raffel, E. Olson, S. K. Snderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, diogo149, B. McFee, H. Weideman, takacsg84, peterderivaz, Jon, instagibbs, D. K. Rasul, CongLiu, Britefury, and J. Degrave. Lasagne: First release., Aug. 2015.

[6] O. Emad, I. A. Yassine, and A. S. Fahmy. Automatic Localization of the Left Ventricle in Cardiac MRI Images Using Deep Learning. pages 683–686, 2015.

[7] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 2:97–104, 2004.

[8] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2014.

[9] Y. Lu, P. Radau, K. Connelly, A. Dick, and G. a. Wright. Segmentation of left ventricle in cardiac cine mri: An automatic image-driven method. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5528:339–347, 2009.

[10] L. Marak and J. Cousty. 4D Morphological segmentation and the MICCAI LV-segmentation grand challenge. *. . . 2009 Workshop on . . .*, 2009.

[11] C. Petitjean and J.-N. Dacher. A review of segmentation methods in short axis cardiac {MR} images. *Medical Image Analysis*, 15(2):169 – 184, 2011.

[12] P. Radau, Y. Lu, K. Connelly, G. Paul, A. J. Dick, and G. A. Wright. Evaluation Framework for Algorithms Segmenting Short Axis Cardiac MRI. *The MIDAS Journal - Cardiac MR Left Ventricle Segmentation Challenge*.

[13] V. Tran. A fully convolutional network for left ventricle segmentation. `https://gist.github.com/ajsander/b65061d12f50de3cef5d#file-fcn_tutorial-ipynb`.

[14] P. VanMaasdam. Image preprocessing: The challenges and approach. `http://www.datasciencebowl.com/image-preprocessing-the-challenges-and-approach/`, Jan. 2016.