DeepMD: Transforming How We Diagnose Heart Disease Using Convolutional Neural Networks

Viswajith Venugopal
Stanford University
viswa@stanford.edu

Swaroop Ramaswamy
Stanford University
swaroopr@stanford.edu

Abstract

Cardiovascular disease is an frighteningly common affliction in today's world, and heart failure can strike at any time, often with lethal consequences. This calls for effective methods to diagnose heart disease early, to enable those at high-risk to take steps to protect themselves. A feasible way to do this is using Magnetic Resonance Imaging (MRI) to take pictures of the heart, and measure the volume of the heart at different stages of a heart-beat, based on which the patient's risk for heart disease can be assessed. Measuring this volume is a difficult and time-consuming task, even for experienced cardiologists; with modern advanced image-processing tools like Convolutional Neural Networks, however, it is not overly optimistic to expect us to build machine learning models that can automate this task. We seek to build such a model.

We explore various preprocessing techniques and convolutional neural network architectures to solve this problem, and present their results. We find that our best model achieves a Continuous Ranked Probability Score of 0.0322 on the test set – which puts us in the top 20% in the Kaggle contest – and performs well for healthy hearts (those with a high ejection fraction).

1. Introduction

The problem we tackle is that of taking in MRI images of human hearts, and automatically measuring the end-systolic and end-diastolic volumes – that is, the volume of the heart when it is contracted, and after it is filled with blood. This is a regression problem with two outputs: the input to our algorithm is a time series of 30 images, which correspond to one entire cycle of a heart beat, and we are asked to output the end-systolic and end-diastolic volumes in millilitres.

There are strong reasons to pursue this problem. Cardiovascular illness is one of the biggest killers in today's society. In the United States alone, 1500 people are diagnosed with heart failure *every day*. Key to tackling this devastating malady is early diagnosis: declining heart function can be measured, and is a strong indicator of heart disease.

Currently, a method used by doctors to determine cardiac function is that of measuring the size of a chamber of a heart at the beginning and end of each heartbeat, and computing an associated metric called the ejection fraction. Both the volumes and the ejection fraction are predictive of heart disease. The gold standard test to accurately measure these volumes is Magnetic Resonance Imaging (MRI). However, currently, using MRI to measure cardiac volumes is manual and slow – it takes a skilled cardiologist up to 20 minutes, time which she could be spending with her patients. Thus, making this measurement process more efficient will enhance our ability to diagnose heart conditions early, and will lead to huge advances in heart disease treatment.

We work on a dataset released as part of the 2015 Kaggle Data Science Bowl [1], which consists of MRI images from more than 1,000 patients. This data set was compiled by the National Institutes of Health and Children's National Medical Center and is an order of magnitude larger than any cardiac MRI data set released previously. We have trained several models on this data, quantified how well we have done, and understood what works well and what doesn't. Our best model puts us in the top 20% of the Kaggle leaderboad for this contest.

The rest of this report is organized as follows. Section 2 discusses relevant work from existing literature on this subject. Section 3 explains the problem in more detail, in order to shed more light on what is involved in solving it. Section 4 presents a mathematical discussion of our methods. Section 5 talks about the key characteristics of this dataset. Section 6 details the experiments we performed, and the results obtained. Section 7 summarizes our conclusions and section 8 discusses future steps.

2. Related Work

There hasn't been much work per se on using Convolutional Neural Networks (CNNs) [16][15] to analyze MRI scans of the heart; presumably, this contest represents one

of the first attempts at this. However, there is a significant amount of literature on applying machine learning models on radiology problems. [26] serves as a good overview of these approaches, bringing up domain-specific issues that frequently come up. It talks about machine learning algorithms useful for medical image segmentation, brain function or activity analysis, and content-based image retrieval systems for CT or MRI images. [24] is also a less recent, but useful reading for this purpose.

Classically, most of the problems that computer visions have been used for have involved segmentation. [4] represents an early attempt, using a clever level-set formalism and robust evaluation models. [3] uses techniques involving probabilistic graphical models [14], using the EM algorithm [7] over a Markov-Gibbs random field (similar approaches have been used in image modelling before; for an example, see [19]). Segmentation is a good preprocessing step even for cardiac MRIs, so these papers are helpful for understanding the domain better.

There have been other instances of models involving specifically neural network being used for biological problems. A fairly old example is [28], which uses older techniques for extracting a 2D histogram from CT scans, and then using neural networks to find patterns in the histogram. Obviously, this is an old approach, and it is almost certain that their results can be improved using CNNs – this hypothesis is validated, for example, by [10], which uses simple CNNs to diagnose traumatic brain injuries, with results that significantly outperform other models.

One particular medical imaging problem deep learning has proven to be particularly effective at is that of diagnosing Alzheimer's disease. [21], which uses CNNs with fairly standard architectures, along with sparse autoencoders [20], and predicts Alzheimer's from MRI scans with amazing accuracy. [17] is yet another example of using deep learning to diagnose Alzheimer's. [13] speaks of using similar models on the more ambitious problem of 'brain extraction', with methods that outperform the state-of-the-art.

Although there are few instances of neural networks being used on heart MRI scans themselves, there has been some work on using other computational techniques on these MRI scans. [18] talks about left ventricle segmentation, which is directly tied to our problem of estimating the left ventricle's volume. It does this segmentation from first principles, which makes the approach interesting and interpretable (and hence, we explore it further below). In a similar vein, [8] does surprisingly well using manually extracted features and a KNN classifier – but it is almost certain that deep learning techniques will do better automatically. There are also several common semi-automatic segmentation techniques: [22] compares several such techniques, and their effectiveness on determining cardiac function in rats.

3. Diagnosing Heart Disease with MRI Images

In this section, we take a closer look at the problem, in order to shed more light on the prediction problem, and the steps involved in solving it.

3.1. Ejection Fraction

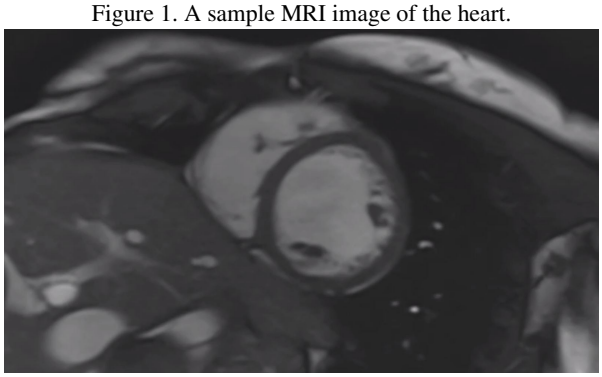
Cardiac function is diagnosed from MRI scans by measuring the volume of the left ventricle (lower left chamber of the heart) at two points in time: after systole, when the heart is contracted and the ventricles are at their minimum volume, and after diastole, when the heart is at its largest volume. The volume at systole, V_S, and the volume at diastole, V_D, are by themselves good predictors of heart disease. An even better predictor is the ejection fraction [2] which is computed as:

$$EF = 100 * \frac{V_D - V_S}{V_D} \tag{1}$$

This quantity represents the fraction of outbound blood pumped from the heart with each heartbeat. A low ejection fraction is a strong predictor for a wide range of cardiac problems.

3.2. MRI Images

Figure 1. A sample MRI image of the heart.



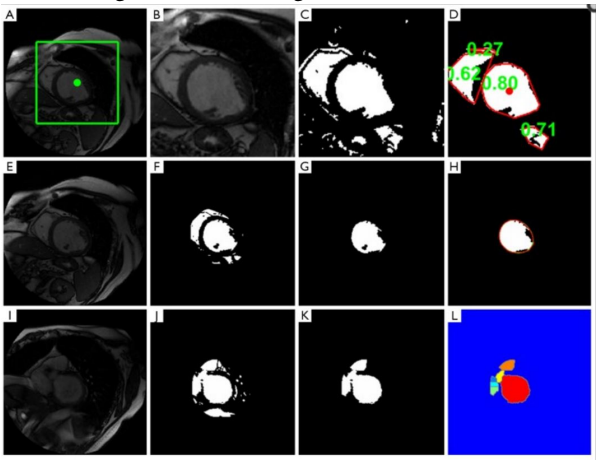
The primary tool for measuring these volumes is MRI scans. The dataset comprises of hundreds of images in DICOM[1] format. An example image is shown in Figure 1.

Variations in anatomy, function, image quality, and acquisition make this a challenging problem. The competition dataset provides a diverse representation of cases, containing patients from young to old, images from numerous hospitals, and hearts with normal to abnormal levels of cardiac function, which makes our problem even more challenging.

To understand what is involved in this analysis, it is helpful to understand how cardiologists tackle this problem. Figure 2, which is taken from the discussion in [18], gives a useful overview of one particular approach to this process. Boxes A-D show the process of localizing the left ventricle.

[1] https://en.wikipedia.org/wiki/DICOM

Figure 2. An MRI segmentation workflow.



Once it is localized, its contours need to be detected. This is called the endocardial contour, and this process is shown in boxes E-H. Boxes I-L are not directly relevant to our problem (for the interested reader, they show the identification and segmentation of the basal slice to find the outflow tract of the left ventricle).

Given the significant noise inherent in the production of these MRI images, the large patient-to-patient variance, and the fact that the end goal is a regression problem (that is, the estimation of the exact volume of a heart), it is not surprising that each individual step in this process is difficult. It takes experienced cardiologists 20 minutes to come up with measurements that are accurate to within 10 mL.

3.3. Evaluation Metric

This is a regression problem, and the evaluation metric that was used for the Kaggle contest is the Continuous Ranked Probability Score (CRPS). In order to explicitly model the uncertainty in all these predictions, for each MRI, we must predict a cumulative probability distribution for both the systolic and diastolic volumes, and the CRPS is computed as:

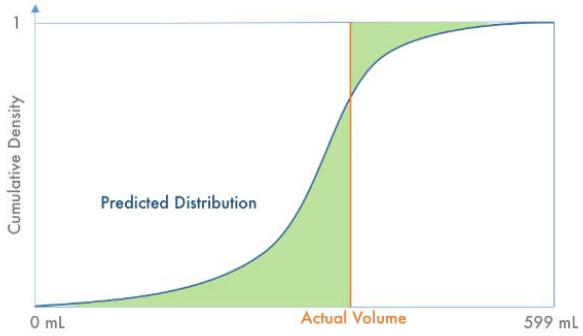
$$C = \frac{1}{600N} \sum_{m=1}^{n} \sum_{n=0}^{599} \left(P(y \leq n) - \mathbb{I}\left[(n - V_m) \geq 0\right]\right)^2 \tag{2}$$

where P is the predicted distribution, which is the predicted cdf of the volume, and $\mathbb{I}$ is the indicator function, which is 1 when its argument is true and 0 otherwise. We only need to provide the cdf value at integral values of the volume (in millilitres). The support of the distribution is taken to be from 0 to 600 mL – this range is wide enough to handle all the cases. Note that, although it is called a score, **the smaller the CRPS, the better**.

While direct visualization of this metric is difficult, an interpretation of this metric can be gleaned from the visualization in Figure 3. Basically, the smaller the highlighted area, the closer we are to the ground truth (which is a step function), and hence, the better. However, it should be noted that the shaded area is not exactly the CRPS. It is in fact, the non-squared version:

$$C = \frac{1}{600N} \sum_{m=1}^{n} \sum_{n=0}^{599} \left|P(y \leq n) - \mathbb{I}\left[(n - V_m) \geq 0\right]\right| \tag{3}$$

Figure 3. A visualization of the CRPS score.



4. Methods

As we have noted in the Related Work section, several methods have been proposed to perform left ventricle segmentation of the heart. Both due to the nature of the class we are doing this project for, and our belief that CNNs will outperform them, they form the core of our work. We assume a basic familiarity with CNNs in this section; the uninitiated reader is referred to a resource such as [11].

4.1. Augmentations

In this section, we describe some layers and techniques we used to enhance the performance of our networks.

4.1.1 Batch Normalization

Batch Normalization [9] is a technique to accelerate deep network training by explicitly normalizing the activations throughout a neural network to take on a unit Gaussian distribution. This reduces the internal covariate shift, the phenomenon where the input distribution to each layer changes as the parameters of its previous layers are updated. Thus, batch normalization layers are inserted between other layers in the network; to avoid becoming overly restrictive, the batch normalization layers have a scale and shift parameter (which default at 1 and 0, corresponding to a unit Gaussian) to make sure that the networks still have the power to learn any function. The key here is that this is a completely differentiable operation, and can be backpropagated through.

The normalization is done with respect to a minibatch of data. That is, if the input to a batchnorm layer is

the d-dimensional vector $(x_1, \dots x_d)$, the output is the d-dimensional vector $(y_1, \dots y_d)$ given by:

$$\hat{x}_k = \frac{x_k - \mathbb{E}[x_k]}{\sqrt{\text{Var}[x_k]}} \tag{4}$$

$$y_k = \gamma_k \hat{x}_k + \beta_k \tag{5}$$

where the mean and variance are taken for each dimension across the minibatch, and γ_k and β_k represent the scale and shift parameters respecitvely for dimension k.

4.1.2 Dropout

Dropout [23] is an innovative technique to deal with overfitting in neural networks. Here, at training time, some neurons are randomly dropped, along with their connections, during every forward pass. This prevents units from co-adapting too much, and forms an effective and simple method of regularization and model combination. That is, given a d-dimensional input vector $(x_1, \dots x_d)$, the output from the dropout layer is the d-dimensional vector $(y_1, \dots y_d)$ given by:

$$y_k = \text{Bern}(p) x_k \tag{6}$$

where $p \in [0, 1]$ is the dropout parameter, interpreted as the probability of the unit participating in the forward pass, and $\text{Bern}(p)$ is 1 with probability p and 0 with probability $1 - p$.

4.1.3 Optimization with Adam

Adam [12] is a stochastic gradient descent optimization algorithm which works very well in practice, and which we used as a default for our experiments. Concretely, the update from x_t to x_{t+1} is done using the gradient dx and the learning rate γ as follows:[2]

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) dx \tag{7}$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) dx^2 \tag{8}$$

$$x_{t+1} = x_t - \gamma \frac{m_{t+1}}{\sqrt{v_{t+1} + \epsilon}} \tag{9}$$

Here, β_1 and β_2 are decay parameters, m is a momentum vector (which can be interpreted as the "smoothened" gradient, and v is a vector that enables our updates to be per-parameter adaptive. (A more detailed explanation can be found at [25]).

4.1.4 Rotational and Shifting Augmentation

There is significant variance in not only the size, but also the shape and exact location of a patient's left ventricle in an MRI scan. To help our network combat this, we perform rotational and shifting augmentation: for each point in our training set, we create copies which are rotated and shifting by random amounts, and add it to our network with the same train labels. The idea is that this helps make our model more invariant to these transformations.

[2]Note: this is the update done in the middle of optimization, but during the beginning of optimization, the update is modified slightly to allow the vectors m and v to "warm up". For full details, refer to [12].

4.2. Loss Function Formulations

Since we are tackling a regression problem, the question of how to formulate the loss function becomes tricky. We tried several approaches.

4.2.1 Mean Squared Error

One approach is to make our network predict one real-valued output, and compute the loss (excluding regularization) as simply the mean of the sum of the squared deviations on all the data points in the mini-batch. That is:

$$L = \frac{1}{N} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{10}$$

We then compute a Gaussian CDF from using the prediction as the mean and the loss as the variance.

4.2.2 Cross-Entropy Loss

Another approach is to discretize our output into d-bins, and make our networks predict d-values, interpreting each output as the probability that the true value lies in the d-th bin. We compare this with the true probability distribution, and measure the deviation as:

$$L = \frac{1}{N} \sum_{i=1}^{n} \left(- \sum_{j=1}^{d} y_d \log(\hat{y}_d) \right) \tag{11}$$

We then construct a CDF from this PDF output.

4.2.3 Cosine similarity

This is similar to the approach using cross entropy loss, except instead of using the cross entropy loss, we use the cosine similarity measure.

$$L = - \frac{A.B}{||A|| ||B||} \tag{12}$$

Here A is the predicted probability vector and B is the true probability vector. Note that for both these approaches, we manually generate the true probability distribution by taking the true volume as a mean and choosing a small standard deviation.

4.2.4 Continuous Ranked Probability Score

Continuous Ranked Probability Score (CRPS) is the main evaluation metric for the Kaggle competition we participated in. It is described in detail in Section 3.3.

To directly minimize the CRPS, we treat the output of the neural network to be a CDF. We impose non-negativity and monotonicity by modifying the output when computing the loss. For instance, all negative values are set to 0 and if any value $v[i+1] < v[i]$ we set $v[i+1] = v[i]$. The hope is that the network will learn to predict values such that this loss is correctly minimized.

4.3. Preprocessing

In this section, we detail the preprocessing steps we undertook.

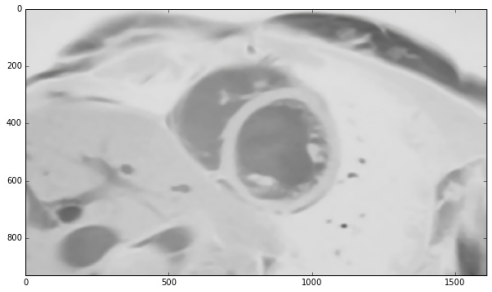
4.3.1 Resizing, Normalizing and Centering

For simplicity, we resized all images in our dataset to square 64x64 images. Note that they are greyscale, so they don't have three colour channels. We then normalize our inputs and center them, so that their magnitudes don't exceed 1, and they are centered at their mean.

4.3.2 Total Variational Denoising

We also apply Total Variational Denoising using the Chambolle algorithm [5]. This smoothens the image, removing small dark or bright spots that arise frequently in MRI images. An example is shown in Figure 4; this is the denoised version of the image in Figure 1.

Figure 4. A denoised DICOM image.



4.4. Incorporation of Metadata

The DICOM images have a number of metadata tags that incorporate some important data. Some of the the metadata tags include – gender, age, slice location, pixel spacing, manufacturer, UNIX time etc. Obviously many of these tags are unimportant and can be safely ignored. However, some like age, gender etc. are fairly important and it is possible to construct reasonable models for predicting the heart volumes using just the metadata.

We incorporate the metadata into our model by creating a 3-layer hidden neural net with 100 hidden neurons. The output layer of this neural net contains 10 neurons and it is concatenated with the output the the convolutional neural network, also containing 10 neurons. Finally, we place another hidden layer on top of this and predict the volume.

We believe this to be more robust that using the neural network output as a feature along with the metadata tags to predict the volume. The reason is, building such a model will assign a larger-than-optimal weight for the neural network predictions since predictions of the neural network on the training set will be better than the predictions of the neural network on the validation and test sets.

4.5. Ensembling Techniques

It is well-known that ensembling boosts the performance of neural networks. [27] We explored various techniques of ensembling, combining our models with different weights. In order to predict a probability distribution from a regression network, a good way of estimating the variance of our prediction while simultaneously reducing the error by ensembling is to train three networks, all with the same data, but one with the labels $y +$ offset, another with the labels $y -$ offset, and one with the label y, and combining the predictions of these on test data.

5. Dataset Information and Statistics

As we mentioned earlier, we work on a dataset released as part of the 2015 Kaggle Data Science Bowl [1]. An example can be found in Figure 1. The training set consists of 500 patient cases, with many slices of images available for each case. Each slice is a time series of 30 DICOM images. Most relevant to us are the short axis views, which cut through the heart in a way that makes the left ventricle most visible. We are also given the ground truth values of the systolic and diastolic volume of each case. The validation set consists of 200 such cases; very recently, as the contest came to a close, we were also provided with labels on the validation data. On disk, the training data is 25 GB, and the validation data is 10 GB.

In addition, as we discussed in Section 4.4, the images have a number of metadata tags which could potentially be useful. We carried out all the preprocessing steps we detailed in Section 4.3 on these images.

Just to get a sense of the numbers, the mean systolic volume in our dataset was 71.96 mL, with a standard deviation of 43.2 mL; the mean diastolic volume was 165.86 mL, with

a standard deviation of 59.3 mL. Histograms presenting the distributions of this volume in the training data are shown in Figures 5 and 6.

Figure 5. Histogram of systole volumes in our dataset.

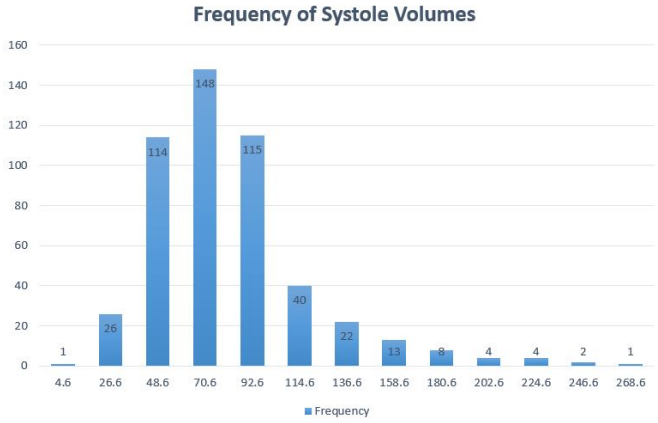
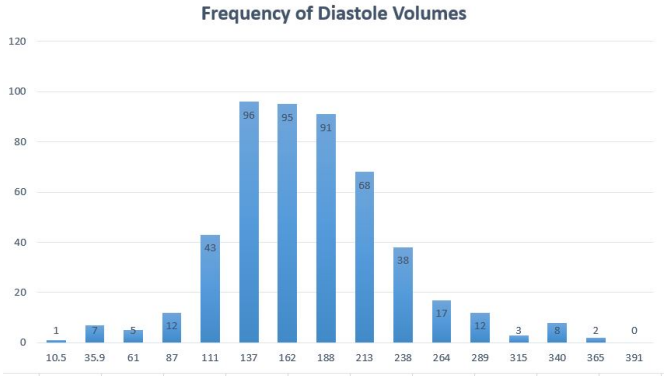


Figure 6. Histogram of diastole volumes in our dataset.



6. Experiments and Results

In this section, we present different experiments we ran, and the results obtained.

6.1. Overview of Workflow

After preprocessing our images as described in section 4.3, we feed the time series into our convolutional network as a volume of $30 \times 64 \times 64$. We tried several targets, with different loss formulations, as described in section 4.2. We used the Adam update, with a learning rate of 10^{-4}. We did all this with a mini-batch size of 32.

To implement this, we used the Keras [6] framework on an Nvidia Tesla K40 GPU.

6.2. Single Images

We attemped to feed in single 64×64 images and predict volumes from the single images. However, we found this to be highly infeasible owing to the large computational time required per epoch. This takes 30 times longer per epoch to run. Moreover, we did not see any performance improvements after running 5 epochs.

6.3. Number of layers

We tried 3 architectures – a 6 layer net, a 7 layer net and a 10 layer net. We found that the 7 layer net achieved better results that both the 6 layer and 10 layer nets. It is possible that if we played with the learning rate, we would have achieved good results with the 10 layer net as well. To enable ourselves to perform more experiments and iterate quickly, we chose to go ahead with the 7 layer net.

Table 1. Results with different number of layers after 100 epochs

	Train CRPS	Val CRPS
6 layers	0.0344	0.0378
7 layers	0.0258	0.0297
10 layers	0.0306	0.0367

6.4. Batch Normalization

Batch normalization had little impact on the speed at which our model trained and but it doubled the amount of time needed per epoch. Therefore, we decided to enable batch normalization only for the last FC layer.

6.5. Dropout

As expected, adding dropout to our model helped us reduce overfitting. Therefore for all further experiments, we enabled dropout.

Table 2. Results with and without dropout after 100 epochs

	Train CRPS	Val CRPS
With Dropout	0.0258	0.0297
Without Dropout	0.0244	0.0312

6.6. Predicting PDF

Predicting the CDF from the PDF as in 4.2.2 and 4.2.3 did not work well at all. The CRPS did not decrease in any significant way. We did not run many iterations as it was not producing results in the right direction.

6.7. Predicting CDF

Predicting the CDF directly as in 4.2.4 produces results that are a little worse than predicting volumes and constructing a CDF from the volumes.

6.8. Including metadata

We include the metadata as in 4.4 and quite surprisingly our results did not show any improvement.

Table 3. Results with direct CDF prediction and CDF construction after 300 epochs

	Train CRPS	Val CRPS	Test CRPS
Direct CDF	0.03975	0.03715	0.0410
Volume to CDF	0.0197	0.0212	0.0326

Table 4. Results with and without metadata after 300 epochs

	Train CRPS	Val CRPS	Test CRPS
With metadata	0.0240	0.0305	0.0332
Without metadata	0.0197	0.0212	0.0326

6.9. Error Analysis

To perform error analysis, we take predictions of systolic and diastolic volumes, identify cases in which the predictions are good/poor and attempt to analyze the reason behind good predictions and poor predictions.

In our analysis, we identified 3 broad cases in which the predictions are poor.

6.9.1 Low Ejection Fraction

In the training set, only 6% of the patients have an ejection fraction that is lower than 40. The percentage of patients that have a low ejection fraction is similar in the validation set. However, because the number of patients with a low ejection fraction is so low, the model does a poor job of predicting volumes for such patients.

Correlation between Systolic MSE and EF = -0.435.
Correlation between Diastolic MSE and EF = -0.26

This negative correlation is actually quite undesirable. A positive correlation is desirable because a positive correlation implies a high false positive rate and a low true negative rate. That is, if the ejection fraction is low, then the model should definitely catch it but if the ejection fraction is normal, then it is okay to report that as low. A low ejection fraction will likely lead to further investigation by the cardiologist. .

6.9.2 Bright patches

We found that the model does quite poorly only images that have bright patches in them. These bright patches are artifacts caused by the MRI machine operator. Better preprocessing should get rid of these artifacts and improve our results on such images. We attempted to solve this issue restricting the pixel values to the middle 90 percentile and rescaling the images. However, this did not give us a significant improvement. Any gains we made on the images with bright patches were offset by the losses we made on the other images.

Figure 7. Error in Systolic Volume vs Ejection Fraction

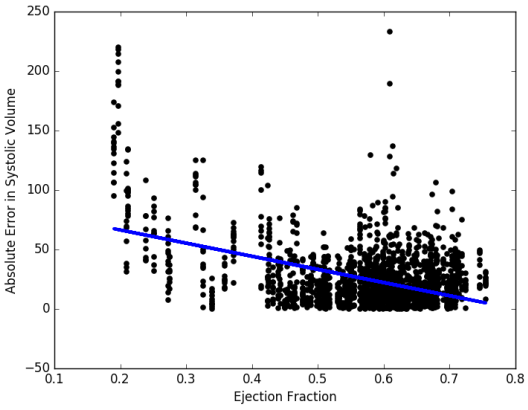
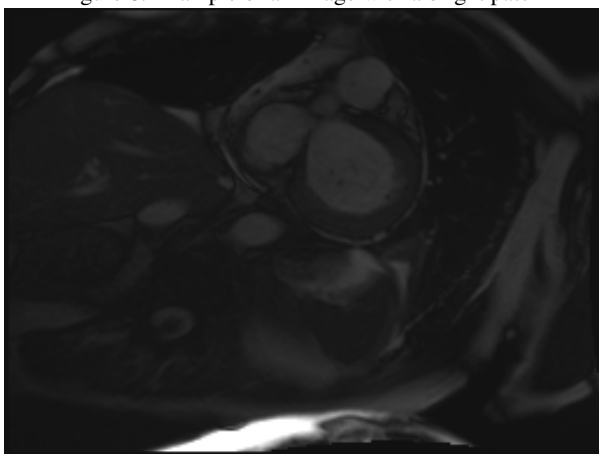


Figure 8. Example of an image with a bright patch



6.9.3 Shaky Images

Sometimes, the images are distorted due to movements of the patient during the MRI Scan. This jittering in the image causes our model to output poor results.

We do not expect our model to do very well for such images, but it would be useful to discard these images in or assign a lower weight to these images when making predictions. A clever way of identifying and discarding shaky images will help mitigate this issue.

6.10. Best Model

Our best model was a 7-layer convolutional network, which takes a $30 \times 64 \times 64$ volume as input, and has the following architecture: `Conv64,3x3 - ReLu - Conv64,3x3 - MaxPool2x2 - Dropout0.25 - Conv96,3x3 - ReLu - Conv96,3x3 - ReLu - MaxPool2x2 - Dropout0.25 - Conv128,2x2 - ReLu - Conv128,2x2 - ReLu - MaxPool2x2 - Dropout0.25 - FC1024 - BatchNorm - ReLu - Dropout0.5 - Out.`

Figure 9. Example of a shaky image

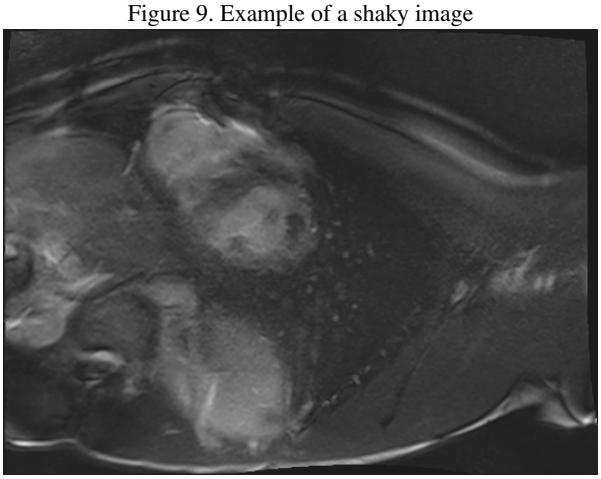


Figure 10. History of Training Loss Versus Epoch.

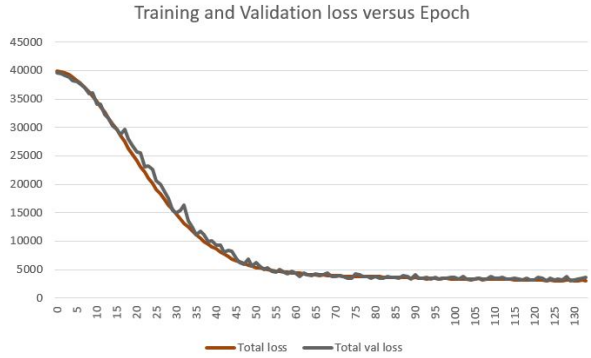
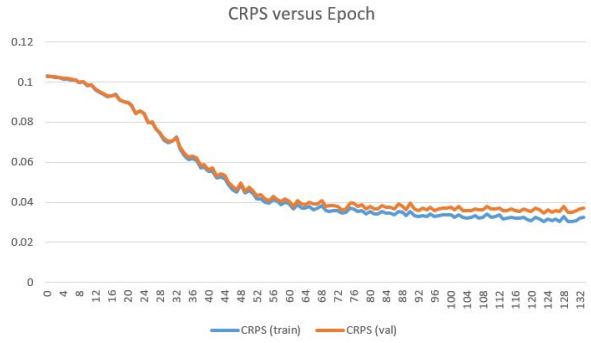


Figure 11. History of Training Loss Versus Epoch.



This model, augmented with the ensembling described in Section 4.5, gave us our best performance, reaching the top 20% of the Kaggle contest. It achieved a CRPS of 0.0322 on the test set. The history of CRPS and training loss respectively versus epoch are shown in Figures 11 and 10.

7. Conclusions

After running several experiments, we have a model which is reasonably successful in predicting the systolic and diastolic volume of a heart from MRI images. We found, however, that the problem is even harder for patients with a low ejection fraction, which is unfortunate. The best model we obtained was on running a 7-layer convolutional network on denoised images; this put us in the top 20% of the Kaggle competition.

How good is our model? What do these CRPS numbers mean? The average cardiologist is usually off by around 10 mL in computing systolic and diastolic volumes. The median error of our model is 17 mL for the systolic volume and 30 mL for the diastolic volume. While this is not something we would trust our lives with, we believe this shows that with some tuning and more data, it is possible to achieve expert-like performance.

8. Future Work

This section details how we plan to take things forward from here.

8.1. Changes to the input layer

Currently, we predict the volume based on 30 images from a single slice. Clearly, it is not a great idea to attempt to predict the volume based on the output of a single slice. It would be much better to construct a 3-D image by concatenating these images and attempt to predict the volume using that. The biggest hurdle in this regard is that the position of the slices is fairly arbitrary and the number of slices per person is variable.

8.2. Better incorporation of metadata

As highlighted earlier, incorporation of metadata, surprisingly, did not give us any performance improvements. This may partly be due to the way in which we incorporate the metadata. We could look at either using the metadata along with the FC-7 features, or using the metadata to preprocess the inputs fed into the neural network.

8.3. Different Approaches for predicting CDF

Is there a natural way of predicting a CDF from a neural network by restricting the outputs to be in [0, 1] and non-decreasing? The approaches which we took incorporate these in the loss function, but this is not very effective and our predictions are worse than if we directly predict a volume and construct a CDF.

Similarly, is there a simple way of minimizing the CRPS score directly by allowing the neural network to output a PDF? When predicting PDFs and constructing a CDF from it, we used a cross-entropy or cosine similarity loss but these do not directly translate to minimizing CRPS, which is the evaluation metric we care about. Is there is a better proxy for CRPS than cosine similarity or cross entropy error?

References

[1] Data science bowl cardiac challenge data.

[2] Ejection fraction heart failure measurement. http://www.heart.org/HEARTORG/Conditions/HeartFailure/DiagnosingHeartFailure/Ejection-Fraction-Heart-Failure-Measurement_UCM_306339_Article.jsp.

[3] A. M. Ali and A. A. Farag. *Advances in Visual Computing: 4th International Symposium, ISVC 2008, Las Vegas, NV, USA, December 1-3, 2008. Proceedings, Part I*, chapter Automatic Lung Segmentation of Volumetric Low-Dose CT Scans Using Graph Cuts, pages 258–267. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[4] C. Baillard, P. Hellier, and C. Barillot. Segmentation of brain 3d mr images using level sets and dense registration. *Medical image analysis*, 5(3):185–194, 2001.

[5] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1-2):89–97, 2004.

[6] F. Chollet. keras. https://github.com/fchollet/keras, 2015.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[8] M. M. Hadhoud, M. I. Eladawy, A. Farag, F. M. Montevecchi, and U. Morbiducci. Left ventricle segmentation in cardiac mri images. *American Journal of Biomedical Engineering*, 2(3):131–135, 2012.

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] K. Kamnitsas, C. Ledig, V. Newcombe, J. Simpson, A. Kane, D. Menon, D. Rueckert, and B. Glocker. Segmentation of traumatic brain injuries with convolutional neural networks, 2015.

[11] A. Karpathy. Cs231n convolutional networks notes. http://cs231n.github.io/convolutional-networks/, 2016.

[12] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] H. S. M.-H. B. B. Kleesiek, Urban. Deep mri brain extraction: A 3d convolutional neural network for skull stripping.

[14] D. Koller and N. Friedman. Probabilistic graphical models.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] Y. LeCun, K. Kavukcuoglu, C. Farabet, et al. Convolutional networks and applications in vision.

[17] R. Li, W. Zhang, H.-I. Suk, L. Wang, J. Li, D. Shen, and S. Ji. Deep learning based imaging data completion for improved brain disease diagnosis. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*, pages 305–312. Springer, 2014.

[18] Y.-L. Lu, K. A. Connelly, A. J. Dick, G. A. Wright, and P. E. Radau. Automatic functional analysis of left ventricle in cardiac cine mri. *Quantitative imaging in medicine and surgery*, 3(4):200, 2013.

[19] R. M. Mohamed, A. El-Baz, and A. A. Farag. Image modeling using gibbs-markov random field and support vector machines algorithm. *International Journal of Information Technology*, 1(4).

[20] A. Ng. Sparse autoencoder.

[21] A. Payan and G. Montana. Predicting alzheimer's disease: a neuroimaging study with 3d convolutional neural networks. *arXiv preprint arXiv:1502.02506*, 2015.

[22] J. Riegler, K. K. Cheung, Y. F. Man, J. O. Cleary, A. N. Price, and M. F. Lythgoe. Comparison of segmentation methods for mri measurement of cardiac function in rats. *Journal of Magnetic Resonance Imaging*, 32(4):869–877, 2010.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[24] K. Suzuki. Special issue on machine learning for medical imaging. *Algorithms*, 2:3.

[25] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[26] S. Wang and R. M. Summers. Machine learning and radiology. *Medical image analysis*, 16(5):933–951, 2012.

[27] Z.-H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1):239–263, 2002.

[28] D. Zukić, A. Elsner, Z. Avdagić, and G. Domik. Neural networks in 3d medical scan visualization. *arXiv preprint arXiv:0806.2925*, 2008.