Diagnosis of Heart Disease via CNNs

Kaicheng Wang
Stanford University
kwang2@stanford.edu

Yanyang Kong
Stanford University
yanyangk@stanford.edu

Abstract

Our project predicts volume of heart by 2D MRI measurement. Combining pre-trained VGG [13] and self-trained networks, we build our Convolutional Neural Networks (CNNs) for prediction. Specific preprocessing methods are designed for our messy data. CNNs in various depths and regularization strength are tried for best validation result. In this Kaggle Challenge contest, our model beats the baseline CNN structure written by Marko Jocic [6].

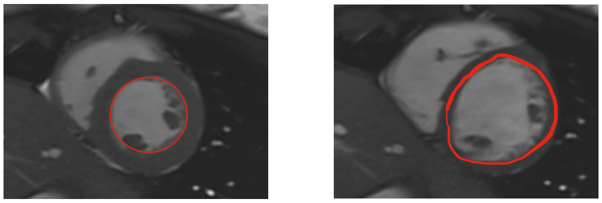
1. Introduction

Using MRI data to measure cardiac end-systolic (V_S) and end-diastolic (V_D) volumes (i.e., the size of one chamber of the heart at the beginning and middle of each heartbeat) (fig[1]) and then deriving the ejection fraction (EF) of heart is a standard process to assess the heart's squeezing ability. Declining EF is a key indicator of heart disease. However, the current process is manual and slow. The cardiologist could spend up to 20 minutes with one patient. Considering the huge amount of patients with potential heart failure, quick automatic measurement will help doctors to diagnose heart conditions more efficiently. [1]

$$EF = 100 \cdot \frac{V_D - V_S}{V_D}$$

Convolutional Neural Networks (CNNs) have been proved remarkably effective on neuroimaging data. As a powerful visual model, CNNs can yield many interesting hierarchies of features, which can be used to classification and segmentation. Contemporary CNN models like AlexNet[8], VGG-Net [13], GoogLeNet[14], ResNet[5] have been used and transferred to learn representations by fine-tunning weights in many different tasks.

In our project, we are going to apply deep Convolutional Neural Networks to predict the end-systolic volume V_S and end-diastolic volume V_D from MRI time series data in different axis views (the planes of slice).



(a) end-systolic volume

(b) end-distolic volume

Figure 1: Plots of of end-systolic, end-distolic volumes in one cardiac circle, circled in red curve

1.1. Problem Statement

Given one patient's several (10 axes in average) MRI measurements in different axes (sax_5,, 2ch_12, 4ch_32, etc), we are going to predict his or her left ventricle's end-distole volume and end-systole volume in one cardiac cycle. Each measurement contains 30 times series images in one cardiac cycle.

We aim to build two deep CNN regression models to predict these two volumes separately. The overview of our problem is shown in fig [2].

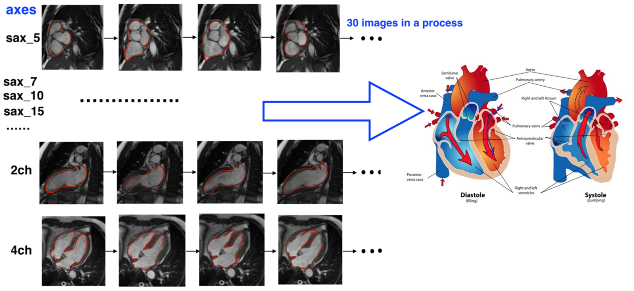


Figure 2: Using 30 MRIs during one cardiac cycle from different axis views to predict V_S and V_D

[1] This project is the Second Annual Data Science Bowl Challenge problem: https://www.kaggle.com/c/second-annual-data-science-bowl

1.2. Challenges

In this problem, variation in heart shape and image quantity of patients makes automated quantification of volume challenging. In the training dataset, we have a diverse representation of cases. Patients form different hospitals may be measured by MRI in various axes. Besides, normal and abnormal images of hearts are mixed together. In an extreme case, we have different poses and directions even in one cardiac cycle. Thus these images are highly distinct from each other. We need a robust model to validate and automate the cardiologists' manual measurement of ejection fraction.

2. Related Work

Deep learning, especially Convolutional Neural Networks have been applied to medical imaging recognition in recent years. Sergey Plis, et al. [12] applied deep neural networks to learn structural and functional brain imaging data and showed that deep learning methods are able to learn physiologically important representations and detect latent relations in neuroimaging data. Payan et al. [11] used 3D convolutional neural network to predict Alzheimer's disease based on brain MRI images. They first initialized the filters for CNN by sparse auto-encoder and then built a CNN whose first layer using the filters learned with auto-encoder. Their model proved to be a success in discriminating between healthy and diseased brains. However, their neural network model was too simple and they didn't go on to try more sophisticated neural network models. Liu and Shen [9] also applied deep CNN models on raw MRI images. They found the regions of interest (ROI) that may be correlated with Alzheimer's disease, which spares the effort of manual ROI annotation process. Brebisson et al. [4] constructed a deep convolutional neural network called `SegNet` for anatomical brain segmentation.

In cardiac study area, N Kannathal, et al. [7] have implemented neural networks for classification of cardiac patient states using electrocardiogram (ECG) signal. Yaniv Bar et al. [3] used a CNN trained by ImageNet and obtained area under curve (AUC) of 0.93 for Right Pleural Effusion detection, 0.89 for Enlarged heart detection, 0.79 for classification between healthy and abnormal chest x-ray, which shows that deep learning with large scale non-medical image database may be sufficient for general medical image recognition tasks.

In terms of our specific problem, there is a baseline CNN structure written by Marko Jocic [6]. He simply treated all images equally and trained a ConvNet with 6 CONV-layers and 2 FC-layers. Each group of 30 images is an input and the corresponding volume is output. This model is easy to compute but not precise. We will show how we beat the baseline model by treating images differently and more accurate models.

3. Data [1] and Preprocess

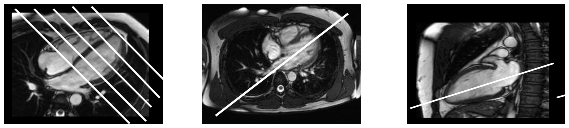
We will examine hundreds of cardiac MRI images in DICOM format for each patient. This dataset was compiled by the National Institutes of Health and Children's National Medical Center. It is an order of magnitude larger than any cardiac MRI data set released previously. We only utilize pixel information of DICOM file.

3.1. Details of Data

In the training dataset, we have 500 patients undergoing about 10 experiments (measurements) from different axes planes. Each experiment observes one slice of heart, which leads to 30 images across the cardiac cycle. Different experiments are acquired from separate breath holds. This is important since the registration from slice to slice is expected to be imperfect. Besides, there are another 200 patients in the validation dataset. In short, We have about $500 \times 10 \times 30$ raw training images and $200 \times 10 \times 30$ raw validation images.

3.2. Axis-based Preprocessing

Notice that the amount of measurements of cardiac MRI varies from patient to patient (not necessarily 10). These images are taken from different views, or axis planes in medical imaging terminology, like `2ch_16, 4ch_17, sax_5, sax_6, sax_7, sax_8`,etc. Each patient has different planes for his or her heart. According to [10], `ch` represents left ventricular long axis acquisition planes. The `2_ch` (fig [3b]) and `4_ch` (fig[3c]) views are used to visualize different regions of the left atrium, mitral valve apparatus. `sax` represents short axis acquisition planes(fig [3a]). These stacks are oriented parallel to the mitral valve ring, and are acquired regularly spaced from the cardiac base to the apex of the heart.



(a) short axis stack (b) 2-chamber view (c) 4-chamber view

Figure 3: Views of sax, 2_ch, 4_ch.

Since `sax` views are excellent in volumetric measurements, as fig[4] shows, we divide these planes into four regions, with three regions for continuous `sax` stacks, called **sax-1**, **sax-2**, **sax-3** correspondingly and another region for `ch` views called **ch**. The **sax-1** corresponds to the first third of `sax` views of that patient, **sax-2** corresponds to the second third of the `sax` views and **sax-3** corresponds to the last third of the `sax` views. Region `ch` represents for `2-ch` and `4-ch` views, which are additional views and less important than `sax` views.

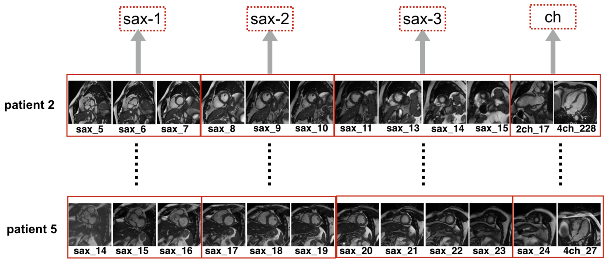


Figure 4: Example of patient 2 and patient 5 with their axis based regions.

After partitioning the MRIs into 4 regions for each patient, we first preprocess the images by resizing or zero-padding to (224,224) dimensions and replicate the gray MRI image to 3 channels to fit VGG-net. We also augment the data set by randomly rotate and shift slightly before putting to CNN.

We extract features of MRI images via pre-trained CNN (we use VGG-19 [13] in our model) in 4 regions by randomly sampling one slice in every region. Then we use these 4 region features as input for our own self-trained networks. In this way, not only do we keep the input dimension constant for every patient, but also we enlarge our data by random combination of slices in different regions.

3.3. Minimum and Maximum Image Selection

Since we only need to predict end-systolic volume V_S and end-diastolic volume V_D for each patient, and these two volumes are corresponding to the minimum and maximum volume of heart. From 30 images in a cardiac cycle, there is one image records the maximum volume and another image records the minimum volume, so the other 28 out of 30 images are noises to our regression models. Since the heart region in MRI has brighter pixels, which means the values in those pixels are larger than other pixels, so end-diastole status corresponds to more bright pixels in MRI and end-systole status corresponds to less bright pixels in MRI, see fig[1]. In practice, based on these two criteria: sum of pixels of image and the number of pixels with value above some threshold or below some threshold, we pick out 2 images in one time series indicating end-systolic and end-diastolic status. This automatic selection method is verified by hand. simple to implement and effective in our experiments.

4. Methods

We treat our problem as supervised regression and tackle this problem by convolutional neural network models. In this section, we discussed our models in details and also talked about evaluation criteria.

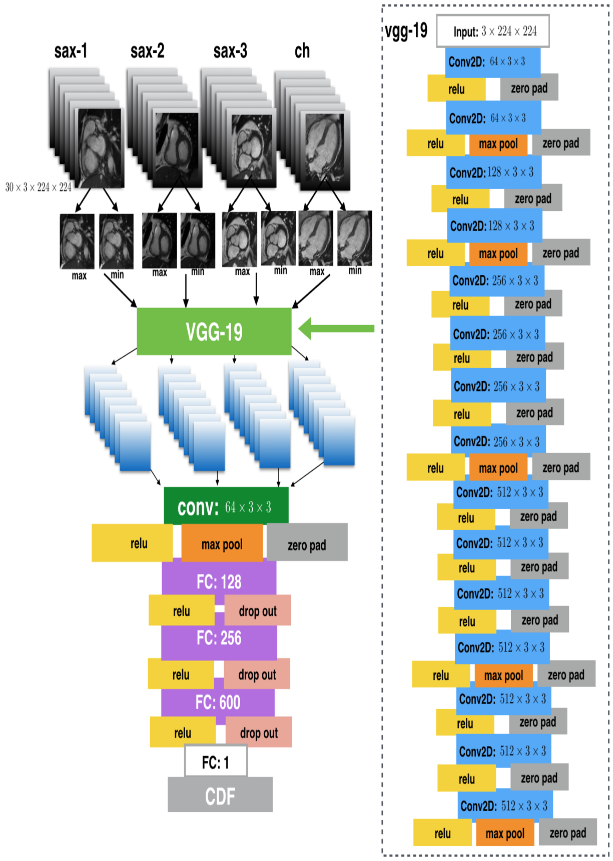


Figure 5: CNN Structure

According to study of [3], the pretrained CNNs on other image dataset like ImageNet are still powerful to extract useful hierarchical information for medical images, we choose VGG-19 [2] as pretrained CNN layers to extract features, which are built in our whole model, see fig[5].

4.1. CNN Structure

As mentioned in section 3, the current input for each patient are 2 groups of 4 images sizing in (3,224,224) dimensions. One is for end-systolic model and the other is for end-diastolic model. Taking one group of 4 images for example, we extract features of them separately using pre-trained VGG-19 [10] network, their outputs are concatenated and reshaped into 3 dimensions before being imported to our self-trained networks. The output of these two modes is one number, which is the predicted volume. Then we transform this predicted volume to CDF using Gaussian distribution with mean as the predicted value and standard deviation as squared root error. Our general CNN structure is shown in fig [5].

In experiments, we built 8 models for systolic voluem V_S and diastolic volume V_D respectively. Their pre-trained

VGG and self-trained network are different from each other. We optimize our model according to the evaluation result. Details of models are introduced in Section 4.3. The reason for building such models are explained in Section 5.

4.2. Evaluation

Since our outputs are 2 volumes, V_S and V_D, we use Root Mean Squared Error (RMSE) as our loss function. We implement CNNs in **Keras** with Adam optimizer.

Based on our volume prediction μ and loss σ^2, we first calculate CDF (cumulative probability distribution) for V_S and V_D by fitting normal distribution with mean as predicted class value and variance as loss $N(\mu, \sigma^2)$. Intuitively the predicted distribution curve will be sharper when we have lower loss, since we are more confident about our prediction. Then our model is evaluated on the Continuous Ranked Probability Score (CRPS) as mentioned in Kaggle, see fig[6]:

$$C = \frac{1}{600N} \sum_{m=1}^{N} \sum_{n=0}^{599} (P(y \leq n) - H(n - V_m))^2$$

where $H(x)$ is the Heaviside step function ($H(x) = 1$ for $x \geq 0$ and 0 otherwise).

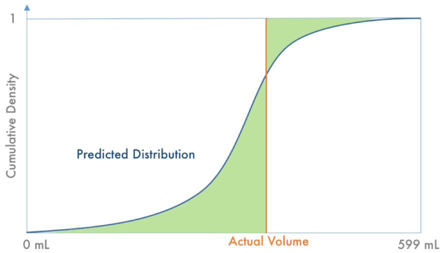


Figure 6: plot of predicted distribution, with error measured as the area of green region.

4.3. Models

Our model is divided into 2 parts. As for the pre-trained VGG part, we compare results of 3 models (*Model-1*, *Model-2*, *Model-3*), the depths of which increase as more CONV-layers are added. As for the self-trained network part after VGG, we explored optimal parameters of 1 conv-layer and 2 fully-connected (FC) layers (*Model-4* to *Model-8*). See the table [1] for more details.

8 models are experimented for V_S and V_D respectively. Since *Model-3* performs better than *Model-1* and *Model-2* in either cases, *Model-4* to *Model-8* are built based on *Model-3*. In the end, we choose the best model for predicting V_S and V_D independently according to CRPS on validation data. *Model-1* to *Model-3* utilize same self-trained FC-layers while adding more pretrained Conv-layers. Based on *Model-3*, *Model-4* removes the last Conv-layer in VGG and adds one Conv-layer to the pre-trained network. *Model-5* to *Model-8* keep the same structure of pre-trained VGG as in *Model-4*, and the only difference between them is the strength of regularization. The schematic diagrams of all models are showed in the table above[1].

Model	Structure
1	Conv-vgg:8 Conv-Layers(fixed) + FC(128, 256, 600, 1)(to be trained)
2	Conv-vgg:12 Conv-Layers(fixed) + FC(128, 256, 600, 1)(to be trained)
3	Conv-vgg: 16 Conv-Layers(fixed) + FC(128, 256, 600, 1)(to be trained)
4	Conv-vgg: 15 Conv-Layers(fixed) + 1 Conv-Layer(64, 3, 3)(to be trained) + FC(128, 256, 600, 1) (to be trained)
5	Conv-vgg: 15 Conv-Layers(fixed) + 1 Conv-Layer(64, 3, 3)(to be trained) + FC(128, 256, 600, 1) (to be trained + reg(l-2:1e-3))
6	Conv-vgg: 15 Conv-Layers(fixed) + 1 Conv-Layer(64, 3, 3)(to be trained) + dropout(0.5) + FC(128, 256, 600, 1) (to be trained + reg(l-2:1e-3))
7	Conv-vgg: 15 Conv-Layers(fixed) + 1 Conv-Layer(64, 3, 3)(to be trained) + FC(128, 256, 600, 1) (to be trained + reg(l-2:1e-1))
8	Conv-vgg: 15 Conv-Layers(fixed) + 1 Conv-Layer(64, 3, 3)(to be trained) + dropout(0.5) + FC(128, 256, 600, 1) (to be trained + reg(l-2:1e-1))

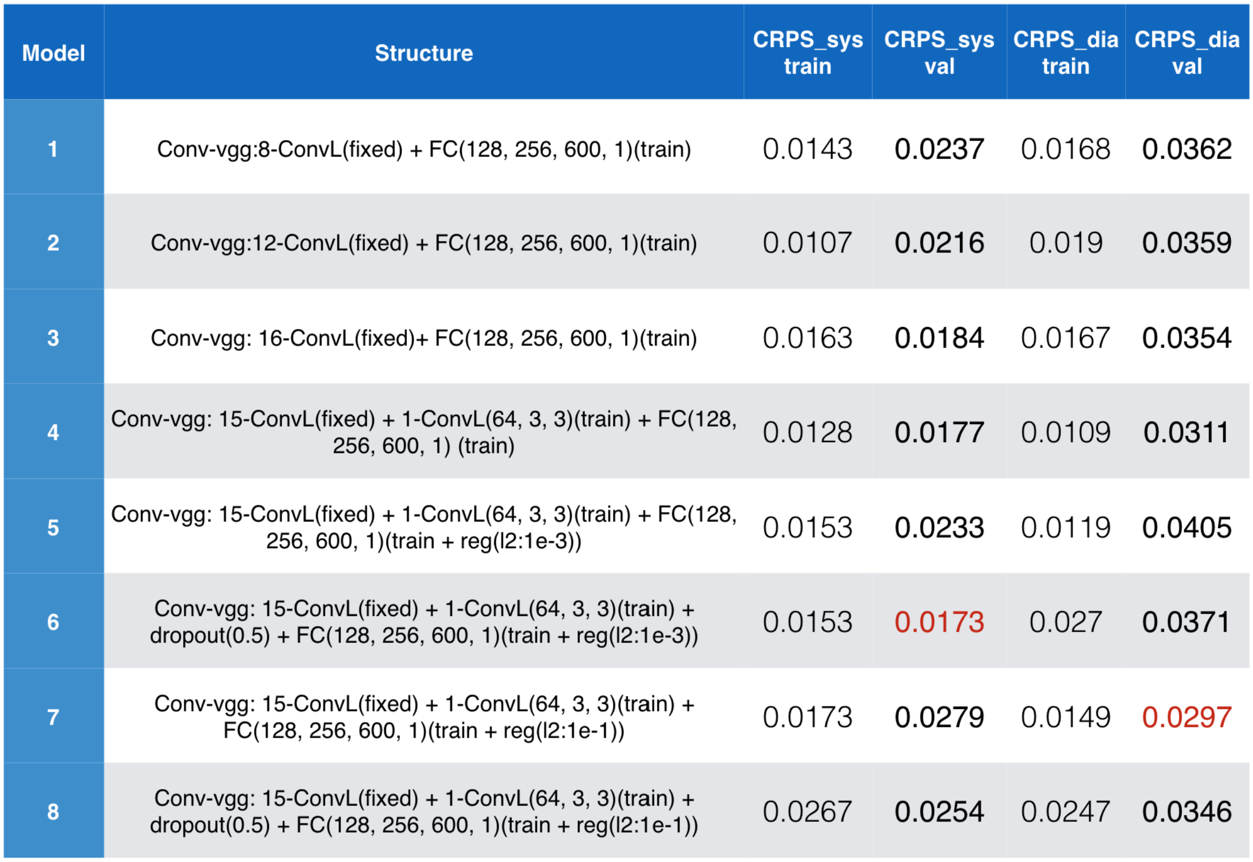
Table 1: The structures of 8 models in our experiments

5. Results and Discussion

We run our models on Stanford Rye01 with configuration as 8 core (2x E5620) cpu, 48GB ram, 250GB local disk, 6x C2070, Ubuntu 13.10 and CUDA 6.0.

In each model, we use mini batch gradient descent to train the weights and the batch size is 100. We train the models over 400 iterations and get the CRPS for 8 models in the table [7].

We choose the optimal systolic model and diastolic model by CRPS on validation dataset. According to the table[7], *Model-3* does perform the best in first 3 models in either cases, which implies that deeper pre-trained lay-



Model	Structure	CRPS_sys train	CRPS_sys val	CRPS_dia train	CRPS_dia val
1	Conv-vgg:8-ConvL(fixed) + FC(128, 256, 600, 1)(train)	0.0143	**0.0237**	0.0168	**0.0362**
2	Conv-vgg:12-ConvL(fixed) + FC(128, 256, 600, 1)(train)	0.0107	**0.0216**	0.019	**0.0359**
3	Conv-vgg: 16-ConvL(fixed)+ FC(128, 256, 600, 1)(train)	0.0163	**0.0184**	0.0167	**0.0354**
4	Conv-vgg: 15-ConvL(fixed) + 1-ConvL(64, 3, 3)(train) + FC(128, 256, 600, 1) (train)	0.0128	**0.0177**	0.0109	**0.0311**
5	Conv-vgg: 15-ConvL(fixed) + 1-ConvL(64, 3, 3)(train) + FC(128, 256, 600, 1)(train + reg(l2:1e-3))	0.0153	**0.0233**	0.0119	**0.0405**
6	Conv-vgg: 15-ConvL(fixed) + 1-ConvL(64, 3, 3)(train) + dropout(0.5) + FC(128, 256, 600, 1)(train + reg(l2:1e-3))	0.0153	0.0173	0.027	**0.0371**
7	Conv-vgg: 15-ConvL(fixed) + 1-ConvL(64, 3, 3)(train) + FC(128, 256, 600, 1)(train + reg(l2:1e-1))	0.0173	**0.0279**	0.0149	0.0297
8	Conv-vgg: 15-ConvL(fixed) + 1-ConvL(64, 3, 3)(train) + dropout(0.5) + FC(128, 256, 600, 1)(train + reg(l2:1e-1))	0.0267	**0.0254**	0.0247	**0.0346**

Figure 7: CRPS for 8 models

ers are better for our transferable learning. This indicates some similarities between our dataset and ILSVRC dataset, where the VGG is trained.

At present, only 2 FC-layers are trained in our models. We then take a further step to add a trainable CONV-layers on top of our self-trained network at the same time we remove the last CONV-layer in pretrained VGG just to be fair. That is how *Model-4* is built. We notice overfitting problem in the first 4 models. By adding dropout and choosing different l-2 regularization strength, we experiment 4 more models to fight against overfitting. Since *Model-4* performs best in the first 4 models, we set *Model-5* to *Model-8* inherit identical structure from *Model-4*.

According to the final results, We choose *Model-6* to predict systolic volume V_S and *Model-7* to predict diastolic volume V_D. Notice that for every model, we trained end-systolic and end-diastolic model separately. In other words, they share the same structure but are different in weights.

In experiment, the appropriate learning rates differ from model to model. We tune all 16 models manually. End-systolic models are trained at learning rate = $1e-5$ approximately. End-diastolic models' learning rates are usually two magnitudes higher. We present these two best models' learning process in the first 100 iterations in fig[8].

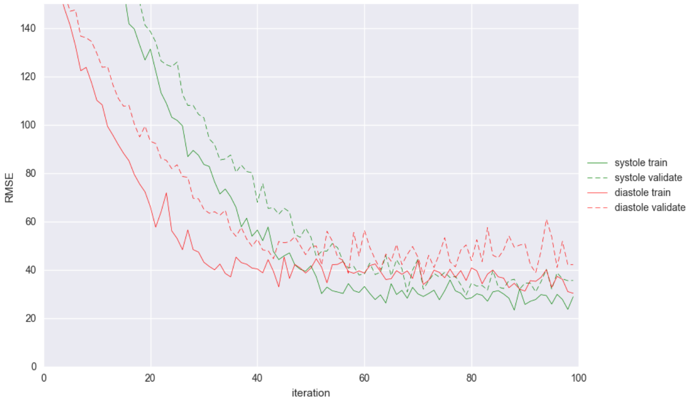


Figure 8: RMSE of systole and diastole models in 100 iterations.

As shown in fig[8], end-systolic model has lower loss than end-diastolic. This is reasonable since the variation of V_S is less than that of V_D, making V_S easier to predict. CRPS drops extremely slowly after the first few hundreds iterations, we finally reach our best result after over 1500 iterations.

The training takes about 70 hours and reaches optimal CRPS at 0.0235, which is an average of these 2 models. We beat the baseline model[6], whose CRPS is 0.0359.

We also plot the best models fitting results on 200 patient test dataset, shown in fig[9]. The true volumes are aligned in a line with increasing order to make observe convenient, and the dots are predicted volumes for that patient. The vertical distance between the dot and the line represents the prediction error for that patient.

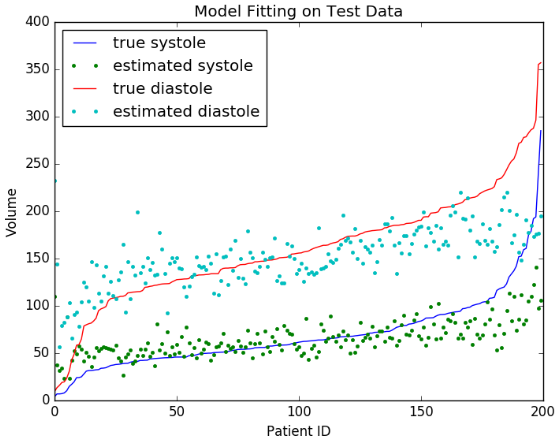


Figure 9: Systole and diastole model fitting on test data

6. Conclusion and Future Work

In our project, CNNs are used to extracted features from 2D images and combine information from different axes views to predict volume. These two roles of CNNs somehow corresponds to our pre-trained VGG and self-trained network.

Preprocessing images and optimizing models are two keys to the final result. Preprocessing is extremely important bacause of messy data. The dimension of input images from each patient should be constant. The importance of different axis views are different. The time when heart's volume reaches maximum or minimum varies even in one patient's records. All these facts lead to our axis-based preprocessing and selection of minimum/maximum image. We manage to keep the dimension of input constant and eliminate noises. Model optimization is essential in transferable learning. It is hard to quantify the resemblance between our data and the data where VGG is trained.Thus different numbers of CONV-layers have to be tested. There is also the tradeoff between minimizing training loss and fighting against overfitting. The strength of regularization depends on the data. Even though end-systolic and end-diastolic model have same amount of data, the choices of regularization strength are different.

In fact, we have only tried very few models due to our limited computation resource. We expect to implement cross validation for potwntial result. Aside from choosing images with minimum and maximum volumes by program, we can also select them by hand for higher accuracy. Besides VGG-Net, we shall try other pre-trained CNN models like AlexNet, GoogLeNet, etc. What is more, if we take other information in the DICOM file into consideration, like gender and age, we might get a more precise model.

Acknowledgment

We would like to thank the CS231N instructors, Andrej Karparthy and Justin Johnson and staff for their help and guidance for our project. We also want to thank Marko Jocic [6] for his baseline model structure posted in Kaggle forum.

References

[1] Data science bowl cardiac challenge data.

[2] Vgg-19 pretrained weights.

[3] Y. Bar, I. Diamant, L. Wolf, and H. Greenspan. Deep learning with non-medical training used for chest pathology identification. In *SPIE Medical Imaging*, pages 94140V–94140V. International Society for Optics and Photonics, 2015.

[4] A. Brebisson and G. Montana. Deep neural networks for anatomical brain segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–28, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[6] M. Jocic, 2016. https://github.com/jocicmarko/kaggle-dsb2-keras/.

[7] N. Kannathal, U. R. Acharya, C. M. Lim, P. Sadasivan, and S. Krishnan. Classification of cardiac patient states using artificial neural networks. *Experimental & Clinical Cardiology*, 8(4):206, 2003.

[8] 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.

[9] F. Liu and C. Shen. Learning deep convolutional features for mri based alzheimer's disease classification. *arXiv preprint arXiv:1404.3366*, 2014.

[10] J. Margeta, A. Criminisi, D. C. Lee, and N. Ayache. Recognizing cardiac magnetic resonance acquisition planes. In *MIUA-Medical Image Understanding and Analysis Conference-2014*, 2014.

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

[12] S. M. Plis, D. R. Hjelm, R. Salakhutdinov, and V. D. Calhoun. Deep learning for neuroimaging: a validation study. *arXiv preprint arXiv:1312.5847*, 2013.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.