The Detection and Segmentation of the Left Ventricle of the Heart from Ultrasound Data using Deep Learning Architectures and Efficient Search Methods

Gustavo Carneiro*, Jacinto C. Nascimento, *Member, IEEE,* António Freitas, Ph.D.

This work was supported by project the FCT (ISR/IST plurianual funding) through the PIDDAC Program funds and by project PTDC/EEA-CRO/098550/2008. This work was also supported by project 'HEARTRACK' - PTDC/EEA-CRO/103462/2008. *This work was partially funded by EU Project IMASEG3D (PIIF-GA-2009-236173).

Gustavo Carneiro (corresponding author) and Jacinto C. Nascimento are with the *Instituto de Sistemas e Robótica*, *Instituto Superior Técnico*, 1049-001 Lisboa, Portugal. Email: gcarneiro@isr.ist.utl.pt and jan@isr.ist.utl.pt. **Phone:** +351-218418270, **Fax:** +351-218418291.

Abstract

We present a new pattern recognition model based on deep learning architectures for the automatic segmentation of the left ventricle of the heart in ultrasound images. Our model addresses the following problems inherent to pattern recognition approaches: 1) the need of a large set of training images, 2) robustness to imaging conditions not present in the training data, and 3) complex search process. We present extensive quantitative evaluation which shows that our method produces state-of-the-art results in a public database using two orders of magnitude less training data than current pattern recognition approaches. This is relevant given the difficulty in acquiring training data for medical image analysis applications. We also show that our method correlates well with user annotations ($r = 0.98, p < 10^{-5}$) and that the results produced by the system are within inter-user variations. Finally, we also show that efficient search methods reduce up to ten-fold the complexity of the method while it still produces state-of-the-art results.

I. INTRODUCTION

The structure and motion analysis of the left ventricle (LV) of the heart is of particular importance in the diagnosis of cardiovascular diseases [1]. Arguably, echocardiography has become the preferred medical imaging modality to visualize the LV of the heart due to the low cost and portability of the ultrasound imaging devices [2]. Typically, the ultrasound imaging of the LV is analyzed by an expert (e.g., a cardiologist), who segments the endocardial border of the LV at the end systole and end diastole phases which are then used to provide a quantitative functional analysis of the heart, such as the *Ejection Fraction* (EF) estimation. The automation of the LV segmentation (Fig. 1) is desirable in a clinical setting due to the following reasons: 1) it can improve the work flow by increasing the patient throughput; and 2) it can decrease the variability between user measurements. However, automatic LV segmentation systems have to handle several problems present in ultrasound imaging, such as: low signal-to-noise ratio, edge dropout, presence of shadows, no simple relation between pixel intensity and physical property of the tissue [3], and anisotropy of the ultrasonic image formation [3].

The solution for the automatic LV segmentation from ultrasound images has traditionally followed three approaches: 1) bottom-up, 2) model-based, and 3) pattern recognition. Bottom-up approaches [4,5] are usually based on morphological operators and standard image analysis techniques (e.g., edge detection, edge linking, Hough transform, etc.), but in general, they cannot deal with all the issues presented above. Model-based methods [6,7] focus on building a prior model of the LV shape and appearance. The shape model involves the assumption that the LV shape is smooth and can be represented by a pre-defined

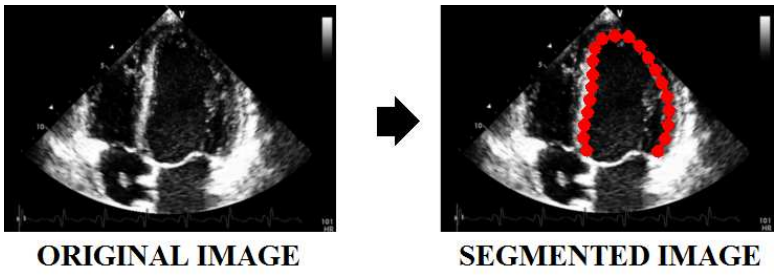


Fig. 1. The method proposed in this paper is a fully automatic system that takes an LV image (apical two or four-chamber views), such as the one on the left and produces an LV segmentation using a set of control points (or key points) and an interpolation method that fills the gaps between the control points (represented in red on the right).

template. In terms of appearance, it is assumed that the gray-value distributions inside and outside the LV are different from each other and that a strong edge is present in the LV border. Though successful, it is still unclear whether the priors developed in model-based methods can cover all possible variations of LV appearance and shape. Pattern recognition methods involve the use of a database of annotated LV images (i.e., a training set) to automatically build a statistical model of the LV appearance and shape [8]–[10]. In general, these methods do not assume any prior model of the LV and rely on a manually annotated training set to automatically determine the parameters of the statistical appearance and shape models. Pattern recognition approaches are robust to all imaging conditions present in the training set, which means that the success of such approaches relies on the size and richness of the training set. Moreover, the optimization process to detect and segment the LV consists of a non-convex problem, which increases the complexity of the search process. In summary, pattern recognition methods face the following challenges: 1) the need of a large and rich training set, 2) robustness to imaging conditions unseen in the training set, and 3) the run-time complexity of the search process. Lately, there has been a significant effort to reduce the search complexity [11,12], but the other two challenges are still open.

In this paper, we propose a new pattern recognition model for automatic LV segmentation that addresses the three problems listed above. In order to handle the robustness to imaging conditions and the need of large training sets, we rely on the use of deep learning architectures [13]. The new learning paradigm introduced by deep learning architectures has shown promising results in the sense of reducing the need for large and rich training sets. The complexity issue is addressed with the use of optimization algorithms of first (gradient descent) and second (Newton's method) orders [14]. The proposed method

for segmenting the LV works according to the following two stages: 1) a rigid classifier detects the rigid transform parameters of the LV in the image, 2) a non-rigid classifier adjusts the LV contour using boundary detection and a learned shape model. Note that both classifiers are learned with deep learning architectures.

Extensive quantitative evaluations show that our method achieves state-of-the-art performance with two orders of magnitude less training data than current pattern recognition approaches. We also show that our method correlates well with user annotations ($r = 0.98, p < 10^{-5}$) and that the segmentation results produced by our system are within inter-user variations using several statistical measures [15]. Finally, we also show that efficient search methods reduce up to ten-fold the complexity of the method while still producing state-of-the-art results [1].

A. Paper Organization

This paper is organized as follows. Sec. I introduces and motivates the work. We provide a literature review in Sec. II. The LV segmentation problem is formally defined in Sec. III. We explain in detail the training and segmentation tasks in Sec. IV. The experiments are shown in Sec. V, and we conclude the work in Sec. VI.

II. Literature Review

In this section, we provide a brief discussion on current solutions for the automatic LV segmentation approaches using ultrasound images. We first cover bottom-up methods, then we review model-based approaches, followed by a discussion on pattern recognition approaches.

A. Bottom-up Methods

The first LV segmentation methods were heavily based on bottom-up techniques [4,5]. In essence, these methods consist of a series of standard image processing techniques to detect the border of the LV. The techniques used include edge detection and linking, morphological operators (e.g., dilation or erosion), and Hough transform. Depending on the quality of the image, these methods can work satisfactorily, but they tend to fail in more challenging image conditions. Given the lack of robustness to imaging

[1]A preliminary version of this paper has been published before [16], but in this journal version, we extend considerably the literature review and explanation of the method. The experiments have been improved with the extension of the comparison with the state-of-the-art algorithms using more error measures and the box plots showing the performance of the algorithm with small training sets. Finally, the statistical comparisons with inter-user variations is completely new.

conditions, the field has explored model-based and pattern recognition methods, which are discussed below.

B. Model-based Methods

Model-based approaches use prior shape and appearance models of the LV. This type of approach has gained significant attention with the introduction of active contour models by Kass *et al.* [6]. The segmentation produced by such method consists of a contour generated by an optimization method that minimizes a cost function with two energy terms. The internal energy is minimized when the contour is smooth, while the external energy is minimized when the contour is located close to a border. The heavy dependence on the presence of edges motivated researchers to study more robust cost functions. For instance, current deformable models usually include a term that maximizes the difference between the texture and/or intensity values between the foreground (interior of LV) and background (exterior of LV) regions, and another term that is minimized when the final shape is close to a prior shape of the structure of interest [7,17]–[19]. It is interesting to note that these methods are usually based on level sets, which is known to work more effectively in magnetic resonance imaging (MRI) than ultrasound data [20].

The underlying prior model of the LV present in these cost functions is based on the following assumptions: the texture/intensity present in the imaging of the myocardium is different from texture/intensity of the blood pool [21]–[23], the LV border is smooth and well represented with edges, and the LV contour can be denoted with a pre-designed prior shape. The main problem with these models is that the violation of any of these assumptions may lead to an incorrect segmentation [24]. For instance, the presence or absence of edges may not correlate with the presence or absence of LV borders [3]. Also, the probabilistic models used to represent the distribution of texture/intensity values of the blood pool or the myocardium are usually too simple (e.g. Rayleigh distribution) and cannot capture all possible appearance variations. Another weakness present in this appearance term is the lack of a spatial distribution of the image features. Finally, the initialization of all these methods is critical because these optimization functions are usually non-convex, which means that the initial guess has to be close to a "good" local minimum to produce satisfactory results and converge sufficiently fast.

Lately, researchers studying model-based approaches have worked on reducing the dependence on good initial guesses (note that the initial guess is usually provided manually). Zagrodsky *et al.* [25] propose an automatic segmentation of the LV in 3-D ultrasound data. Specifically, a global alignment of the LV is obtained via mutual information, and this initial shape is used to initialize the deformable

model. Nevertheless, note that this deformable model uses the same assumptions above and as a result is susceptible to the same weaknesses. Jacob *et al.* [26] demonstrated empirically how it is possible to achieve a certain degree of robustness against the issues of deformable models with the use of a relatively simple statistical learning of a shape model. The problem is that to automatically learn the model parameters, the method requires a few user annotations on the test images every time a new sequence is presented. It is interesting to see the use of a learning procedure for the shape model, but this approach would be more interesting if the authors have used a separate set of images for training the model "off-line". In addition, the authors use quite weak appearance models (based only on the presence of image transitions that explain the epi/endocardium) for the LV, which is compensated with robust shape and motion models.

C. Pattern Recognition Methods

In pattern recognition methods, the shape and appearance models are represented by statistical models which are automatically learned from a manually annotated training set. In general, pattern recognition methods can be divided into generative and discriminative models [27]. Assume that the observed data can be denoted as $\mathbf{x}$, and the classification of the data is represented by the label y. The generative model learns the joint probability $p(\mathbf{x}, y)$ and in order to calculate $p(y|\mathbf{x})$ one has to use the Bayes rule, where the likelihood $p(\mathbf{x}|y)$ is necessary. On the other hand the discriminative model learns the posterior $p(y|\mathbf{x})$ directly. Ng and Jordan [27] compared these two models and reached the conclusion that discriminative models have a lower asymptotic error than generative models, but the convergence (as a function of the size of training data) of generative models is faster.

The most notable generative models designed for the problem of LV segmentation are the active shape model (ASM) [28,29] and the active appearance model (AAM) [30,31]. These models use manually annotated training data to build a shape model (ASM) or a joint model of shape and appearance (AAM). The lack of a prior LV model made this approach quite successful, but the need of relatively large collections of training data has been an issue because the effectiveness of the method is correlated with the size and richness of the training data. The AAM and ASM have been widely studied [9], and it has been noted that ASM is faster and achieves more accurate segmentation, but AAM can be more robust [32]. Given the advantages of AAM and ASM, there have been works that advocate the combination of the two models in the problem of LV segmentation from ultrasound and MRI data [24,33]. Finally, a common issue with ASM and AAM is that both approaches need a relatively good initial guess to work properly [34]. As mentioned before, generative models, such as AAM and ASM, should be used

whenever the training set is relatively small (at the expense of relatively higher asymptotic errors). For larger training sets, one should focus on the use of discriminative models.

Discriminative models for the LV segmentation have been intensively exploited by Comaniciu and colleagues [8,10], who have a quite large set of annotated LV images (in the order of thousands) and take advantage of the lower asymptotic errors described above. Essentially, their discriminative model is based on boosting classifiers [35] and the process of segmenting the LV is broken into two stages. The first stage searches for the rigid transformation of the LV and the second stage estimates the non-rigid deformation by finely adjusting the result from the first stage. This approach holds quite competitive segmentation results [36].

In general, methods based on pattern recognition methods rarely address the training set size issue, which is a limiting factor to the widespread study of such models. Furthermore, the solutions to handle the robustness to imaging conditions have been limited to artificially enlarge the training set by introducing slight variations (i.e., noise) to the annotated images [37,38]. On the other hand, there has been a significant effort to reduce the search complexity of pattern recognition models. Recall that the complexity of the search process resides in the non-convexity of the optimization function used during the detection process. For instance, the marginal space learning (MSL) [11] partitions the search space into sub-spaces of increasing complexity and achieves a significant complexity reduction. Zhou and Comaniciu [12] also propose a method to reduce the search complexity consisting of a statistical method which outputs a gradient vector given a search region. This vector optimizes the LV segmentation function by indicating the preferred direction for the search. This approach is likely to work as long as the search region is sufficiently close to a local optimum of the objective function. In addition, the training procedure is likely to need a larger training set due to the much higher number of parameters to be learned in the gradient vector.

III. Problem Definition

The main problem we wish to solve in this paper is the delineation of the left ventricle in an ultrasound image I. This delineation is denoted by a vector of points $\mathbf{s} = [\mathbf{x}_i^\top]_{i=1..N}$, with $\mathbf{x}_i \in \Re^2$. Note that this set of points is formed by a parametric B-spline curve with uniform parametrization [39], which guarantees the same number of points for each delineation, and the same geodesic distance between points. We assume that $\mathcal{D} = \{(I, \theta, \mathbf{s})_i\}_{i=1..M}$ is the training set containing training images I_i of the imaging of LV using ultrasound, a respective manual annotation $\mathbf{s}_i$ and the parameters of a rigid transformation $\theta_i \in \Re^5$ (position $\mathbf{p} \in \Re^2$, orientation $\vartheta \in [-\pi, \pi]$, and scale $\sigma \in \Re^2$) that aligns the two base points and apical

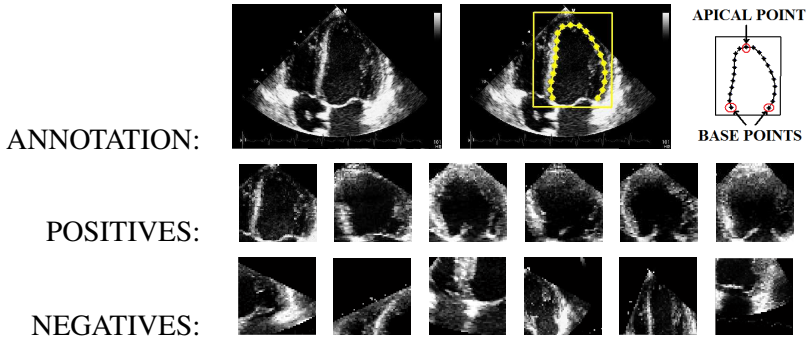


Fig. 2. Original training image (top left) with the manual LV segmentation in yellow line and star markers (top middle) with the rectangular patch representing the canonical coordinate system for the segmentation markers. The top-right image shows the reference patch with the base and apical points highlighted and located at their canonical locations within the patch. The images on the second row display several patches extracted from the annotated training images (i.e., the positive set), and the third row shows examples of negative patches extracted from places outside the annotation parameters. Note that both positive and negative patches will be used to train the rigid classifier.

point to a canonical coordinate system (see Fig. 2). Our objective is to find the LV contour with the following decision function:

$$\mathbf{s} = E[\mathbf{s}|I, y = 1, \mathcal{D}] = \int_{\mathbf{s}} \mathbf{s} p(\mathbf{s}|I, y = 1, \mathcal{D}) d\mathbf{s}, \tag{1}$$

where $y = 1$ is a labeling variable indicating the presence of LV in image I. Notice that the usual goal in pattern recognition methods is to find the parameter $\mathbf{s}$ that maximizes the probability function $p(\mathbf{s}|I, y = 1, \mathcal{D})$, but the use of expectation $E[.]$ in (1) provides a more robust decision process. Equation 1 can be expanded in order to decouple the rigid and non-rigid detections,

$$p(\mathbf{s}|I, y = 1, \mathcal{D}) = \int_{\theta} p(\mathbf{s}|\theta, I, y = 1, \mathcal{D}) p(\theta|I, y = 1, \mathcal{D}) d\theta. \tag{2}$$

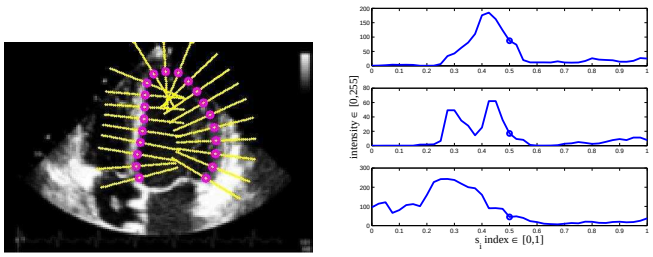
The first term in (2), representing the non-rigid part of the detection, is defined as follows:

$$p(\mathbf{s}|\theta, I, y = 1, \mathcal{D}) = \prod_{i=1}^{N} p(\mathbf{x}_i|\theta, I, y = 1, \mathcal{D}), \tag{3}$$

where $p(\mathbf{x}_i|\theta, I, y = 1, \mathcal{D})$ represents the probability that the point $\mathbf{x}_i \in \Re^2$ is located at the LV contour. Assuming that ψ denotes the parameter vector of the classifier for the non-rigid contour, we compute

$$p(\mathbf{x}_i|\theta, I, y = 1, \mathcal{D}) = \int_{\psi} p(\mathbf{x}_i|\theta, I, y = 1, \mathcal{D}, \psi) p(\psi|\mathcal{D}) d\psi. \tag{4}$$

In practice, we made a few simplifications in (3-4). First, as explained later in Sec. IV-A, we run a maximum a posteriori learning procedure of the classifier parameters, which produces ψ_{MAP}, meaning



(a) (b)

Fig. 3. Lines drawn perpendicularly to the annotation points (a) and their typical profile (b). Notice that each line is divided into 41 equidistant points (see text for details).

that in the integral (4) we have $p(\psi|\mathcal{D}) = \delta(\psi - \psi_{\text{MAP}})$. Second, instead of computing the probability $p(\mathbf{x}_i|\theta, I, y = 1, \mathcal{D}, \psi)$, we train a regressor that, given a set of 41 inputs $\mathbf{z}_i(j)$ for $j = \{1, 2, .., 41\}$, it indicates the index j of the most likely input where the edge is located (see Fig.3). These 41 points are located on the lines perpendicular to the contour $\mathbf{s}$ at each of the $\mathbf{x}_i$ points. This roughly means that $p(\mathbf{x}_i|\theta, I, y = 1, \mathcal{D}, \psi) = 1$ for only the most likely candidate of inputs [36].

The second term in (2) represents the rigid detection, which is denoted by

$$p(\theta|I, y = 1, \mathcal{D}) = Zp(y = 1|\theta, I, \mathcal{D})p(\theta|I, \mathcal{D}), \tag{5}$$

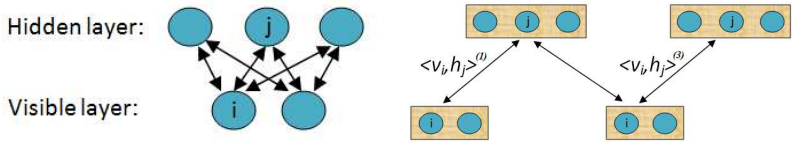
with

$$p(y = 1|\theta, I, \mathcal{D}) = \int_{\gamma} p(y = 1|\theta, I, \mathcal{D}, \gamma)p(\gamma|\mathcal{D})d\gamma, \tag{6}$$

where γ is the vector containing the classifier parameters, and Z is a normalization constant. We make the same approximation as in (4), where we run a maximum a posteriori learning procedure of the classifier parameters, producing γ_{MAP}, which means that in the integral (6) we have $p(\gamma|\mathcal{D}) = \delta(\gamma - \gamma_{\text{MAP}})$. Finally, in (5) the term $p(\theta|I, \mathcal{D}) \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$, with $\mu_\theta = \frac{1}{M}\sum_{i=1}^{M}\theta_i$ and $\Sigma_\theta = \frac{1}{M}\sum_{i=1}^{M}(\theta_i - \mu_\theta)(\theta_i - \mu_\theta)^\top$, and $\mathcal{N}(\mu_\theta, \Sigma_\theta)$ denotes the multivariate normal distribution.

IV. Training and Segmentation Procedures

In this section, we first discuss the deep learning architecture used to build our classifiers, then we examine how the training and testing data sets have been set up. We also explain the training and detection procedures with the efficient search procedures.



(a) RBM (b) RBM training

Fig. 4. Restricted Boltzmann machine (RBM) model (a) and the training procedure (b) [40]. Notice that the bias is not represented, but it is an extra node in each layer with a unidirectional connection with the other layer.

A. Deep Learning Architectures

One of the goals in the development of artificial neural networks (ANN) was to emulate central nervous system with several layers of neurons. The implementation of such ANN with several layers is known as deep learning architectures. However, it has been noticed that the training of such deep ANN is difficult because of two limitations: 1) slow convergence and 2) failure to reach "good" local optima. The main culprit for both issues is the back-propagation algorithm [41] used to find the values for the network parameters by maximizing the posterior probability. Nevertheless, researchers have also noticed that if the initial guess for the network parameters is sufficiently close to a local optimum, then back-propagation manages to converge relatively fast to this local optimum even in deep learning architectures. Hinton and colleagues [13,42] studied a method to produce such initial guess close to a local optimum with the use of multiple layers of restricted Boltzmann machines (RBM).

An RBM is represented with a hidden and a visible layer of stochastic binary units with the energy configuration [13] (Fig. 4):

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in\text{visible}} b_i v_i - \sum_{j\in\text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}, \tag{7}$$

where v_i and h_j are the visible and hidden binary states of units i and j, w_{ij} are the weights and b_i and b_j are the bias terms. The probability of a configuration in the visible units is then defined as [43]

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\tilde{\mathbf{v}}} e^{-E(\tilde{\mathbf{v}},\mathbf{h})}}. \tag{8}$$

Due to the lack of connections within layers, the conditional distributions $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$ are reduced to simple Bernoulli distributions with the following properties [43]:

$$p(\mathbf{h}|\mathbf{v}) = \prod_j p(h_j = 1|\mathbf{v}) = \prod_j \frac{1}{1+e^{-b_j - \sum_i w_{ij} v_i}}, \tag{9}$$

$$p(\mathbf{v}|\mathbf{h}) = \prod_i p(v_i = 1|\mathbf{h}) = \prod_i \frac{1}{1+e^{-b_i - \sum_j w_{ij} h_j}}. \tag{10}$$

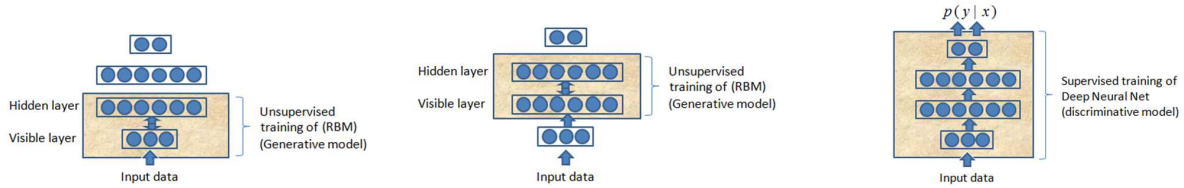
The learning of parameters b_i, b_j, and w_{ij} is based on the contrastive divergence [44], which is an efficient approximation of the log-likelihood gradient used as an update rule for training RBMs. Specifically, the training of RBMs involves maximizing $p(\mathbf{v})$ in (8) by following the gradient $\frac{\partial p(\mathbf{v})}{\partial \eta}$, where η are the parameters b_i,b_j, and w_{ij}. Essentially, as depicted in Fig. 4, the training of an RBM can be summarized in Algorithm 1, where the operator $< . >$ denotes the expected value using a distribution of samples from running a Gibbs sampler, $\varepsilon = 0.1$, and $\max_t = 50$. Hence, this training only finds parameter values for the network that are effective at reconstructing input data in the visible layer, thus forming a generative model trained with unlabeled data (i.e., unsupervised training). Note that the RBM described above works with binary inputs in the visible layer, which has to be adapted to real-valued inputs (our input images are real-valued), as follows:

$$E(\mathbf{v}, \mathbf{h}) = \sum_i \frac{(v_i - b_i)}{2\sigma_i^2} - \sum_j b_j h_j - \sum_i \frac{v_i}{\sigma_i} \left(\sum_j h_j w_{ij} \right), \tag{11}$$

where σ_i^2 is the variance of the visible unit i.

Algorithm 1 RBM training [43].

1: **input**: set $t = 0$ and randomly generate $b_i^{(t)}$, $b_j^{(t)}$, and $w_{ij}^{(t)}$ for all nodes i in visible layer and nodes j in hidden layer.
2: **repeat**
3: $t = t + 1$
4: for each input image $\{\mathbf{v}_k^{(1)}\}_{k \in 1..K}$ applied in the visible layer compute $p(\mathbf{h}_k^{(1)}|\mathbf{v}_k^{(1)})$ (9), which represents the probability of firing each h_j given the input $\mathbf{v}_k^{(1)}$
5: update stochastically the binary hidden units $\mathbf{h}_k^{(2)} \sim p(\mathbf{h}_k^{(1)}|\mathbf{v}_k^{(1)}) \in \{0,1\}^{\#\text{hidden nodes}}$, and generate a reconstruction of the data using $p(\mathbf{v}_k^{(2)}|\mathbf{h}_k^{(2)})$ (10)
6: update stochastically the binary visible units $\mathbf{v}_k^{(3)} \sim p(\mathbf{v}_k^{(2)}|\mathbf{h}_k^{(2)}) \in \{0,1\}^{\#\text{visible nodes}}$, and generate the new probabilities of firing each h_j as in $p(\mathbf{h}_k^{(3)}|\mathbf{v}_k^{(3)})$ (9)
7: update weights and biases for the nodes i of visible layer and j of the hidden layer as follows:
8: $w_{ij} = w_{ij} + \Delta w_{ij}^{(t)}$, with $\Delta w_{ij}^{(t)} = \varepsilon(< v_i, h_j >^{(1)} - < v_i, h_j >^{(3)})$,
9: $b_i = b_i + \Delta b_i^{(t)}$, with $\Delta b_i^{(t)} = \varepsilon(< v_i >^{(1)} - < v_i >^{(3)})$,
10: $b_j = b_j + \Delta b_j^{(t)}$, with $\Delta b_j^{(t)} = \varepsilon(< h_j >^{(1)} - < h_j >^{(3)})$
11: **until** $t > \max_t$



(a) (b) (c)

Fig. 5. Training of deep belief network consists of unsupervised learning of several layers of RBMs (a-b) followed by a supervised learning of the whole network (c).

By stacking several layers of RBMs, and training them in sequence (*i.e.*, the hidden layer values of the trained layer is used as an input of the next visible layer), one can create a deep belief network (DBN) with, in principle, no limit in the number of hidden layers (see Fig. 5). The number of layers is related to the representation capability of the classifier [45]. After several RBMs have been trained as described above, we add a top layer representing the posterior probability to be learned. For the problem of non-rigid detection, the top layer has only one node that outputs $p(\mathbf{x}_i|\theta, I, y = 1, \mathcal{D}, \psi)$ in (4). In practice, we have:

$$p(\mathbf{x}_i|\theta, I, y = 1, \mathcal{D}, \psi) = \begin{cases} 1, & \mathbf{x}_i = \mathbf{z}_i[1 + \lfloor r_i \times 40 \rfloor] \\ 0, & \text{otherwise} \end{cases}, \tag{12}$$

where $r_i = \min\{1, \max\{0, b + \sum_j w_j v_j\}\}$ from the last layer, and 41 is the number of points dividing the line perpendicular to the contour (4). Therefore, we train a regressor that outputs $r_i \in [0, 1]$, which translates into indices as $(1 + \lfloor r_i \times 40 \rfloor]) \in \{1, 2, ..., 41\}$. For the rigid detection, the top layer has two nodes, representing the posterior probability $p(y|\theta, I, \mathcal{D}, \gamma)$ (6), where node one denotes $y = 1$ and node two means $y = 0$. Therefore, we have:

$$p(y = 1|\theta, I, \mathcal{D}, \gamma) = \frac{\exp(b_1 + \sum_i w_{i1} v_i)}{\sum_{j=1}^{2} \exp(b_j + \sum_i w_{ij} v_i)}, \tag{13}$$

and $p(y = 0|\theta, I, \mathcal{D}, \gamma) = 1 - p(y = 1|\theta, I, \mathcal{D}, \gamma)$. For both cases, after adding the top layer, the whole network is trained using a supervised learning process based on backpropagation [41]. This training procedure finds the maximum posterior as follows: $\psi_{\text{MAP}} = \arg\max_\psi p(\{\mathbf{s}_i\}_{i=1..N}|\{I_i\}_{i=1..N}, \psi)$ in (4), and $\gamma_{\text{MAP}} = \arg\max_\gamma p(y = 1|\{(I, \theta)_i\}_{i=1..N}, \gamma)$ in (6), where $(I, \theta, \mathbf{s})_i \in \mathcal{D}$.

1) Discussion: Notice that deep learning architectures introduce a new training paradigm, as shown in Fig. 6. In this figure, the concept of "stuff" is introduced as anything that can be represented with an image. For instance, the LV is the "stuff" that can be imaged through an ultrasonic device, and the label

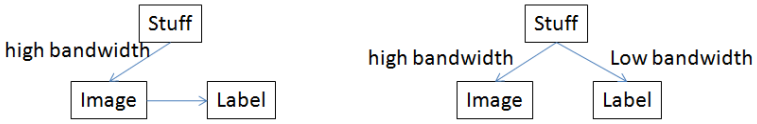


Fig. 6. New learning paradigm explored in deep learning architectures. On the left, it is displayed the current paradigm of machine learning, where it is assumed that all labeling information to an image (e.g., the LV segmentation) is independent of the original cause (the imaging of the left ventricle of the heart) given the image. On the right, it is shown the deep learning paradigm, where an unsupervised generative model learns image generation process, and then a discriminative model is trained based on this generative model [40].

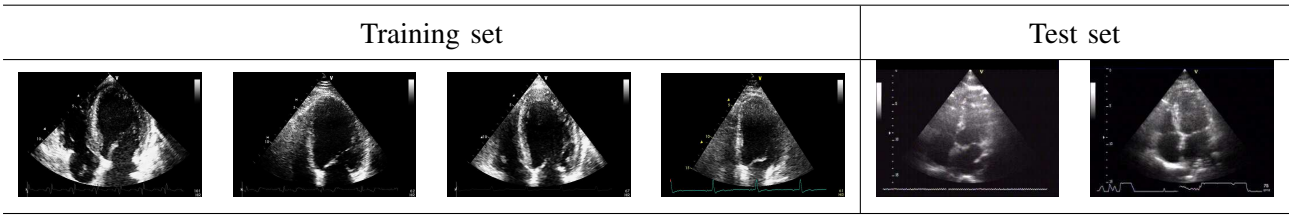


Fig. 7. First images of a subset of the sequences used as training and test sets.

is the LV segmentation. Note that the link between "stuff" and image has a high bandwidth, which means that there are too many ways that the "stuff" can be imaged. Current learning paradigm assumes that label is independent of "stuff" given the image. Therefore, current learning models need to collect a large training set in order to confidently learn the parameters of the statistical model representing the probability of label given image. On the other hand, as explained in this section, deep learning architectures first learn a generative model (trained with unlabeled data) representing the probability of image given "stuff", followed by a discriminative learning (trained with labeled data) of label given "stuff" using the induced LV model obtained during the training process of the first stage. Leveraging the generative model in the learning of the discriminative model is the key that makes deep learning architectures less dependent on large training sets.

B. Training and Testing Data sets and Manual Annotation Protocol

We have two sets of annotated data. The first set contains 400 ultrasound images of left ventricle, which have been taken from 12 sequences (12 sequences from 12 healthy subjects with no overlap), where each sequence contains an average of 34 annotated frames. Let us denote this set as $\mathcal{T}_1$, and each sequence is represented by a letter from A to L. This set contains images using the apical two and four-chamber views. The second set, used exclusively for testing (i.e., it is *never* used for training),

contains two sequences of 80 images, where each sequence has 40 annotated images (2 sequences from 2 healthy subjects with no overlap). This set is denoted by $\mathcal{T}_2$ with sequences A and B. Note that there is no overlap between subjects in sets $\mathcal{T}_1$ and $\mathcal{T}_2$. We worked with two cardiologists, where the first one annotated all images in the database (i.e., sets $\mathcal{T}_1$ and $\mathcal{T}_2$). The other cardiologist annotated the sequences $\mathcal{T}_{1,\{A,B,C\}}$. This additional annotation will help us verify whether the system segmentations are within inter-user variations. The first image of four sequences from $\mathcal{T}_1$ and two sequences from $\mathcal{T}_2$ are shown in Fig. 7.

For the manual annotation, the experts could use any number of points to delineate the LV, but they had to explicitly identify the base and apical points in order for us to determine the rigid transformation between each annotation and the canonical location of such points in the reference patch (see Fig. 2). This vector of points was then interpolated and the final contour has a fixed number of points N with the same distance between points [39].

C. Training Procedure

For the rigid classifier, we follow the multi-scale implementation of Carneiro et al. [37] and build an image scale space $L(\mathbf{x}, \sigma)$ produced from the convolution of the the Gaussian kernel $G(\mathbf{x}, \sigma)$ with the input image $I(\mathbf{x})$, as follows:

$$L(\mathbf{x}, \sigma) = G(\mathbf{x}, \sigma) * I(\mathbf{x}), \tag{14}$$

where σ is the scale parameter, $\mathbf{x}$ is the image coordinate, $*$ is the convolution operation, and $G(\mathbf{x}, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{\mathbf{x}^2}{2\sigma^2}}$. We train three separate classifiers (6) for the following image scales $\sigma = \{4, 8, 16\}$, where the images $L(.)$ are down-sampled by a factor of two after each octave. The values for these scales have been empirically determined using a validation set, from which we observe that larger values for σ in the coarser scale prevents the detection process to converge, and lower values for σ in finer scales did not improve the accuracy of the method. In practice, the original patches (see Fig. 2) have size 56×56 pixels, but the sizes used for scales $\sigma = \{4, 8, 16\}$ were $14 \times 14, 7 \times 7,$ and 4×4 pixels, respectively.

In order to form the positive and negative training sets (Fig. 1), we define a vector denoting a margin variable for each scale $\mathbf{m}_\sigma \in \Re^5$, which increases by a factor of two after each octave [37]. Specifically, considering an interval variable, computed as (the value 400 has been empirically determined based on the trade off between run time detection and training efficacy):

$$\mathbf{t}_\theta = \frac{\max(\{\theta_i\}_{i=1..M}) - \min(\{\theta_i\}_{i=1..M})}{400} \in \Re^5, \tag{15}$$

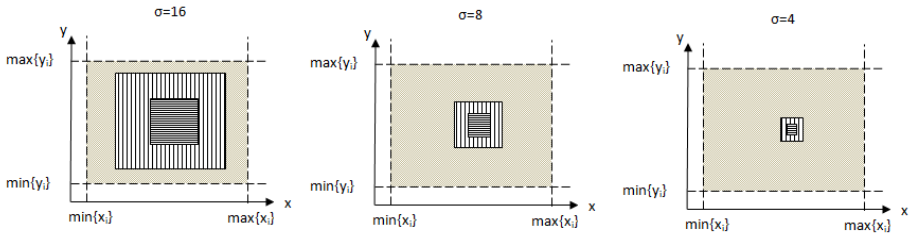


Fig. 8. Multi-scale training. The graphs represent the first two dimensions of the rigid parameter space θ, and the gray square represent the region where negatives are sampled for training, the square represented with vertical lines represent the margin and the square with horizontal lines denotes the region where positives are sampled for training. The ground truth is located at the center of the square represented with horizontal lines.

we set

$$\mathbf{m}_\sigma = 2 \times \sigma \times \mathbf{t}_\theta. \tag{16}$$

This means that coarser scales will have larger margins than finer scales resulting in a trading of robustness for accuracy (see Fig. 8). Positives for each training image I at scale σ are randomly generated within the parameter range $[\theta - \mathbf{m}_\sigma/2, \theta + \mathbf{m}_\sigma/2]$, and negatives are randomly generated *outside* the range $[\theta - \mathbf{m}_\sigma, \theta + \mathbf{m}_\sigma]$, where θ is the parameter vector representing the rigid transformation of the LV annotation. In total we produce 100 positive and 500 negative patches per training image using the margins defined above. This unbalance on the number of positive and negative samples can be explained by the much larger volume covered by the negative region explained above [46].

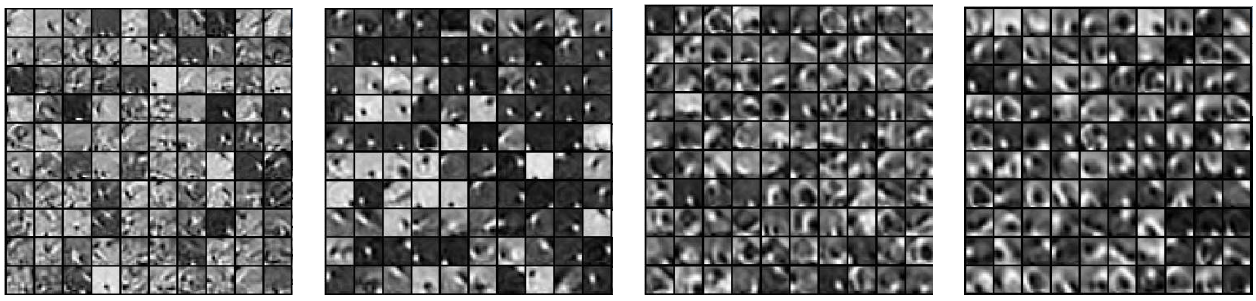
For each scale, we train a DBN using 80% of the data set available for training and 20% for validation. These two sets are mutually exclusive and are selected randomly from the training set. The validation set is necessary to select the following two parameters of the DBN: a) number of nodes per hidden layer, and b) number of hidden layers. The number of nodes per hidden layer varies from 50 to 500 in intervals of 50. The number of hidden layers varies from 1 to 4 (we did not notice any boost in performance with more than 4 layers). Finally the output layer contains only two nodes representing the posterior probabilities of presence/absence of LV given input data. Using the 400 annotated images from training set, we achieved the configurations displayed in Table I (notice that for $\sigma = \{8, 16\}$, only two hidden layers have been selected using the cross validation procedure explained above).

It is worth verifying the types of features learned for the rigid detector. Let $\mathbf{W}_i$ for $i = 1..4$ represent the matrices of weights for each of the four layers of the DBN learned at $\sigma = 4$. From Tab. I, we see that

TABLE I

LEARNED CONFIGURATION FOR THE DEEP BELIEF NETWORKS.

σ	Visible Layer	Hidden Layer 1	Hidden Layer 2	Hidden Layer 3	Hidden Layer 4	Output Layer
4	196 (14×14 pix.)	100	100	200	200	2
8	49 (7×7 pix.)	50	100	-	-	2
16	16 (4×4 pix.)	100	50	-	-	2

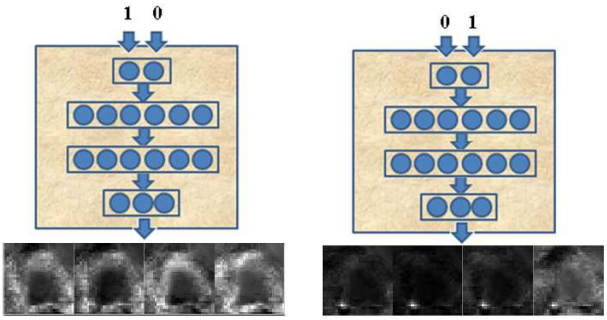


(a) Layer 1 (b) Layer 2 (c) Layer 3 (d) Layer 4

Fig. 9. First 100 features for each layer of the rigid classifier at $\sigma = 4$. From left to right the features learned for each one of the four layers are shown, and as expected, the features tend to be more global for higher layers.

$\mathbf{W}_1 \in \Re^{196\times100}$, $\mathbf{W}_2 \in \Re^{100\times100}$,$\mathbf{W}_3 \in \Re^{100\times200}$,$\mathbf{W}_4 \in \Re^{200\times200}$. The features shown in Fig. 9 depicts the first 100 columns of the following matrices (notice that each 196 dimensional vector is reshaped to a 14×14 matrix): (a) $\mathbf{W}_1$, (b) $\mathbf{W}_1\mathbf{W}_2$, (c) $\mathbf{W}_1\mathbf{W}_2\mathbf{W}_3$, and (d) $\mathbf{W}_1\mathbf{W}_2\mathbf{W}_3\mathbf{W}_4$. It is interesting to see that the features in higher layers tend to be more global than features in lower layers (resembling wavelet features), which demonstrates intuitively the abstraction capabilities of the DBN (similar observations have been noticed by Hinton et al. [43] in other types of experiments). Finally, the DBN can also be used as a generative model by setting the output of LV accordingly (that is, in order to generate positive samples, set the probability of LV to 1, and to generate negative samples, set the probability of LV to 0). Inverting the direction of the weights, one can stochastically generate positive and negative samples. We show a few positive and negative samples generated by the DBN trained at $\sigma = 4$ in Fig. 10. Notice that the positive samples resembles a blurry version of trained LV while the negative samples show significantly less structure.

The non-rigid regressor (12) is trained only at $\sigma = 4$ because it is run only at the finest scale, as explained below in Sec. IV-D.1. In order to increase the robustness of the regressor, we also included



(a) Positive samples (b) Negative samples

Fig. 10. Positive and negative samples generated from the deep belief network trained. Note that the two numbers in the top layer represents the probability of LV (positive sample) given data and the probability of background (negative sample) given data, respectively.

in the training set 100 detections (per training image) within the margin $[\theta - \mathbf{m}_4/2, \theta + \mathbf{m}_4/2]$, which means that the output contour from the rigid classifier might not be correctly aligned (in terms of the rigid transformation) with the LV. Using 80% for training and 20% for validation, the following configuration has been selected: 1) visible layer with 41 nodes, 2) two hidden layers, each with 50 nodes, and 3) output layer with one node.

We also build a shape model based on principal component analysis (PCA) [47,48] that is used to project the final result from the non-rigid regressor. The goal of this last stage is to suppress noisy results from the regressor. Assuming that $\mathbf{X} = [\mathbf{s}_1, ..., \mathbf{s}_M] \in \Re^{2N \times M}$ is a matrix that contains in its columns all the annotations in the training set $\mathcal{D}$, where the mean shape $\mu_{\mathbf{s}} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbf{s}_i$ has been subtracted from each column. Then we can decompose $\mathbf{X}$ using eigenvalue decomposition, as follows: $\mathbf{X}\mathbf{X}^\top = \mathbf{W}\mathbf{\Sigma}\mathbf{W}^\top$. Given a new annotation, say $\tilde{\mathbf{s}}$, we obtain its new value by first projecting it onto the PCA space $\mathbf{y}^\top = (\tilde{\mathbf{s}}^\top - \mu_{\mathbf{s}}{}^\top)\tilde{\mathbf{W}}\tilde{\mathbf{\Sigma}}^{-0.5}$, where $\tilde{\mathbf{W}}$ contains the first L eigenvectors (we cross validated L with the validation set, and selected $L = 10$), and $\tilde{\Sigma}$ is a diagonal matrix containing the first L eigenvalues in the diagonal. Then the final shape $\mathbf{s}^*$ is obtained by re-projecting $\mathbf{y}$ onto the original shape space and adding back the mean shape, as in $\mathbf{s}^* = \mathbf{y}^\top \tilde{\mathbf{\Sigma}}^{0.5} \tilde{\mathbf{W}}^\top + \mu_{\mathbf{s}}$.

D. Detection Procedure

The detection procedure (see Alg. 2) consists of running the rigid classifier at scale $\sigma = 16$ on the K_{coarse} initial hypotheses. These initial hypotheses are determined by sampling from the function $\text{Dist}(\{\theta_i\}_{i=1..N} \in \mathcal{D})$ which describes the distribution of the θ parameters in the training set. Given the

K_{coarse} samples of the function described by the DBN at $\sigma = 16$, we build a new distribution function $\text{Dist}(\{p(\theta_i|I, y = 1, \mathcal{D})\}_{\theta_i \in \text{hypothesis from DBN at } \sigma=16})$, which will be sampled K_{fine} times. These samples are used to initialize a search procedure (explained later in this section) based on the DBN trained at $\sigma = 8$. The output of this search is a new set of samples $\{\theta_i\}_{i=1..K_{\text{fine}}}$. The same procedure is repeated at $\sigma = 4$ and the final K_{fine} samples from $\text{Dist}(\{p(\theta_i|I, y = 1, \mathcal{D})\}_{\theta_i \in \text{hypothesis from DBN at } \sigma=4})$ are used to produce the final contour (1), which is projected onto the PCA space explained in Sec. IV-C. Note that we set $K_{\text{coarse}} = 1000$ and $K_{\text{fine}} = 10$ empirically based on the performance of the detection algorithm on the validation set.

We propose three ways to build such distribution from the K_{fine} points, and they are based on the following three different search approaches, which are explained below: *1) full search, 2) gradient descent, and 3) Newton's method* [14].

Algorithm 2 Detection Procedure.

1: sample $\{\theta_j\}_{j=1..K_{\text{coarse}}} \sim \text{Dist}(\{\theta_i\}_{i=1..M} \in \mathcal{D})$
2: compute $\{p(\theta_j|I, y = 1, \mathcal{D})\}_{j=1..K_{\text{coarse}}}$ using DBN at $\sigma = 16$
3: sample $\{\theta_j\}_{j=1..K_{\text{fine}}} \sim \text{Dist}(\{p(\theta_i|I, y = 1, \mathcal{D})\})_{i=1..K_{\text{coarse}}}$ using DBN at $\sigma = 16$
4: run one of the following search procedures: full, gradient descent, or Newton's method using DBN at $\sigma = 8$
5: sample $\{\theta_j\}_{j=1..K_{\text{fine}}} \sim \text{Dist}(\{p(\theta_i|I, y = 1, \mathcal{D})\})$ using DBN at $\sigma = 8$
6: run one of the following search procedures: full, gradient descent, or Newton's method using DBN at $\sigma = 4$
7: sample $\{\theta_j\}_{j=1..K_{\text{fine}}} \sim \text{Dist}(\{p(\theta_i|I, y = 1, \mathcal{D})\})$ using DBN at $\sigma = 4$
8: run the non-rigid classifier at $\sigma = 4$, and for each rigid parameter $\theta \in \{\theta_j\}_{j=1..K_{\text{fine}}}$ produced in the step above, generate a respective contour $\mathbf{s} \in \{\mathbf{s}_j\}_{j=1..K_{\text{fine}}}$
9: $\tilde{\mathbf{s}} = \frac{1}{\sum_{i=1}^{K_{\text{fine}}} p(\mathbf{s}_i|I, y=1, \mathcal{D})} \sum_{j=1}^{K_{\text{fine}}} \mathbf{s}_j \times p(\mathbf{s}_j|I, y = 1, \mathcal{D})$
10: $\mathbf{y}^\top = (\tilde{\mathbf{s}}^\top - {\mu_{\mathbf{s}}}^\top)\tilde{\mathbf{W}}\tilde{\mathbf{\Sigma}}^{-0.5}$
11: $\mathbf{s}^* = \mathbf{y}^\top \tilde{\mathbf{\Sigma}}^{0.5}\tilde{\mathbf{W}}^\top + \mu_{\mathbf{s}}$.

For the full search, we run the DBN classifier at $\sigma \in \{8, 4\}$ at all the 243 points in $\theta_i + [-\mathbf{m}_\sigma, 0, +\mathbf{m}_\sigma]$ for $i = 1..K_{\text{fine}}$ (note that $243 = 3^5$, that is the five dimensional parameter space of the rigid classifier with three points per dimension). The gradient descent algorithm for each scale in the pyramid is explained in Alg. 3. The gradient $\nabla p(y = 1|\theta_i, I, \mathcal{D}, \gamma_{MAP})$ is computed numerically using central difference, with

Algorithm 3 Gradient Descent.

input: given sample points $\Theta = \{\theta_i\}_{i=1..K}$ in the search space
repeat
  for each $\theta_i \in \Theta$
  $\triangle\theta_i = -\nabla p(y = 1|\theta_i, I, \mathcal{D}, \gamma_{MAP})$
  line search: $t = \arg\max_{t>0} p(y = 1|\theta_i + t \times \triangle\theta_i, I, \mathcal{D}, \gamma_{MAP})$
  update $\theta_i = \theta_i + t \times \triangle\theta_i$
until $\|\nabla p(y = 1|\theta_i, I, \mathcal{D}, \gamma_{\text{MAP}})\|_2 \leq \epsilon$

the step size $\mathbf{m}_\sigma$ (16). A better precision can be achieved with the Newton's method, where the price is the computation of the Hessian matrix and its inversion (see Algorithm 4).

Algorithm 4 Newton's Method.

input: given sample points $\Theta = \{\theta_i\}_{i=1..K}$ in the search space
repeat
  for each $\theta_i \in \Theta$
  Compute Newton step and decrement
    $\triangle\theta_i = -(\nabla^2 p(y = 1|\theta_i, I, \mathcal{D}, \gamma_{MAP}))^{-1} \nabla p(y = 1|\theta_i, I, \mathcal{D}, \gamma_{MAP})$
    $\lambda^2 = \nabla p(y = 1|\theta_i, I, \mathcal{D}, \gamma_{MAP})^\top (\nabla^2 p(y = 1|\theta_i, I, \mathcal{D}, \gamma_{MAP}))^{-1} \nabla p(y = 1|\theta_i, I, \mathcal{D}, \gamma_{MAP})$
  line search: $t = \arg\max_{t>0} p(y = 1|\theta_i + t \times \triangle\theta_i, I, \mathcal{D}, \gamma_{MAP})$
  update $\theta_i = \theta_i + t \times \triangle\theta_i$
until $\frac{\lambda^2}{2} < \epsilon$

The prior distribution to be used by the coarse classifier at $\sigma = 16$ can be sampled using Monte-Carlo sampling techniques, but if this distribution is poorly represented because of limited number of annotations in the training set, then we can partition the parameter space into a grid with the limits given by the training set, and use these K_{coarse} grid points to run the classifier. Specifically in our search approach, we have not noticed much improvement with the use of Monte-Carlo sampling, so we use the computationally simpler grid sampling. Finally, the K_{fine} points are determined from the distribution produced by the classification results of the previous coarser scale through a k-means procedure with K_{fine} clusters. The points to be searched in the finer scale consists of the K_{fine} means of these clusters.

1) Complexity of Search Approaches: The bottleneck of our method in terms of run-time complexity is the execution of the DBN classifier, so it is important to reduce as much as possible the number of times the classifier has to run during a detection procedure. For the run-time complexity presented below, we show the figures in the number of times the classifier is run for each of the search approaches considered.

The *full search* approach explained above has a search complexity of $K_{\text{coarse}} + (\#\text{scales} - 1) \times K_{\text{fine}} \times 3^5 + K_{\text{fine}} \times N$, where K_{coarse} is $O(10^3)$, K_{fine} is $O(10)$, and for the non-rigid classifier, we assume that the detection of each contour point is independent of the detection of other contour points (see Eq. 3), but the dependence between points is restored with the projection of the final result onto the PCA shape model space (Sec. IV-C). From Table I, we notice that the complexity of the classifier at $\sigma = 16$ is $O(16 \times 100 \times 50 \times 2) = O(8 \times 10^4)$, at $\sigma = 8$ is $O(49 \times 50 \times 100 \times 2) = O(2.45 \times 10^5)$, at $\sigma = 4$ is $O(196 \times 100 \times 100 \times 200 \times 200 \times 2) = O(1.56 \times 10^{11})$, and the regressor is $O(42 \times 50 \times 50 \times 1) = O(1 \times 10^5)$. This means that the full search method (using 243 samples in fine scale for each of the K_{fine} samples) needs roughly the following number of multiplications: $1000 \times 8 \times 10^4 + 10 \times 3^5 \times 2.45 \times 10^5 + 10 \times 3^5 \times 1.56 \times 10^{11} + 10 \times 21 \times 1 \times 10^5 \approx 3.8 \times 10^{14}$.

For the *gradient descent* search procedure, each iteration above (at $\sigma \in \{8, 4\}$) represents a computation of the classifier in 10 points of the search space (five parameters times two points) plus the line search in around 10 points as well. The gradient descent search needs roughly the following number of multiplications: $1000 \times 8 \times 10^4 + 10 \times [20, 100] \times 2.45 \times 10^5 + 10 \times [20, 100] \times 1.56 \times 10^{11} + 10 \times 21 \times 1 \times 10^5 \in [3.0 \times 10^{13}, 1.5 \times 10^{14}]$. Notice above that we can limit the number of iterations between one and five, meaning that the complexity of this specific step for one hypothesis θ_i is between 20 to 100, which is smaller than $3^5 = 243$, representing a reduction of the search space between 2 and 10 times.

For the *Newton's method*, the computation of the Hessian, gradient and line search requires 25+10 runs of the classifier. The Newton step search needs roughly the following number of multiplications: $1000 \times 8 \times 10^4 + 10 \times [35, 175] \times 2.45 \times 10^5 + 10 \times [35, 175] \times 1.56 \times 10^{11} + 10 \times 21 \times 1 \times 10^5 \in [5.4 \times 10^{13}, 2.7 \times 10^{14}]$. Limiting the number of iterations between one and five means that the complexity of this step for one hypothesis θ_i is between 35 to 175, which is smaller than $3^5 = 243$, representing a reduction of the search space between around 1.5 and 7 times.

A criticism for the use of the more efficient search procedures of gradient descent and Newton's method is the fact that the function $p(y = 1|\theta, I, \mathcal{D}, \gamma_{MAP})$ in (6) has no guarantee to be convex. However, recall that we sample the initial coarse search space using either a grid or a Monte-Carlo sampling procedure, and then we follow each of the initial hypotheses. It is possible to show that as this initial number of

hypotheses increases, the likelihood of following the path for a "good" local optimum also increases, but this is out of the scope of this paper.

V. EXPERIMENTS

In this section, we first explain the error measures used to assess the performance of our algorithm. Then we present comparisons with state-of-the-art model-based and pattern recognition methods. We also show that our method produces results that agree reasonably well with respect to inter-user variations. Finally, we also discuss the run-time complexity of the method.

A. Error Measures

In this section we describe the following six error measures used for the evaluation of our algorithm: Hammoude distance (HMD) (also known as Jaccard distance) [49], average error (AV) [36], Hausdorff distance (HDF) [50], mean sum of square distances (MSSD) [51], mean absolute distance (MAD) [51], and average perpendicular error (AVP) between the estimated and ground truth contours.

Let $\mathbf{s}_1 = [\mathbf{x}_i^\top]_{i=1..N}$, and $\mathbf{s}_2 = [\mathbf{y}_i^\top]_{i=1..N}$, with $\mathbf{x}_i, \mathbf{y}_i \in \Re^2$ be two vectors of points representing the estimated and reference LV contours, respectively. The smallest distance from a point $\mathbf{x}_i$ to the curve $\mathbf{s}_2$ is

$$d(\mathbf{x}_i, \mathbf{s}_2) = \min_j ||\mathbf{y}_j - \mathbf{x}_i||_2, \tag{17}$$

which is known as the distance to the closest point (DCP). The average error between the vectors $\mathbf{s}_1$, $\mathbf{s}_2$ is

$$d_{\mathrm{AV}}(\mathbf{s}_1, \mathbf{s}_2) = \frac{1}{N} \sum_{i=1}^{N} d(\mathbf{x}_i, \mathbf{s}_2). \tag{18}$$

The Hausdorff distance between both sets is defined as the maximum of the DCPs between the two curves

$$d_{\mathrm{HDF}}(\mathbf{s}_1, \mathbf{s}_2) = \max\Big(\max_i\{d(\mathbf{x}_i, \mathbf{s}_2)\}, \max_j\{d(\mathbf{y}_j, \mathbf{s}_1)\}\Big). \tag{19}$$

The Hammoude distance is defined as follows [49]:

$$d_{\mathrm{HMD}}(\mathbf{s}_1, \mathbf{s}_2) = \frac{\#((R_{\mathbf{s}_1} \cup R_{\mathbf{s}_2}) - (R_{\mathbf{s}_1} \cap R_{\mathbf{s}_2}))}{\#(R_{\mathbf{s}_1} \cup R_{\mathbf{s}_2})}, \tag{20}$$

where $R_{\mathbf{s}_1}$ represents the image region delimited by the contour $\mathbf{s}_1$ (similarly for $R_{\mathbf{s}_2}$), and $\#(.)$ denotes the number of pixels within the region described by the expression in parenthesis. The error measures MSSD [52] and MAD [53] are defined as follows:

$$d_{\mathrm{MSSD}}(\mathbf{s}_1, \mathbf{s}_2) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{y}_i\|_2^2, \tag{21}$$

and

$$d_{\mathrm{MAD}}(\mathbf{s}_1, \mathbf{s}_2) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{y}_i\|_2. \tag{22}$$

Note here that MSSD and MAD are defined between corresponding points (i.e., we do not use the DCP in this case).

Finally, the average perpendicular error (AVP) between estimated (say $\mathbf{s}_2$) and reference ($\mathbf{s}_1$) contours is the minimum distance between $\mathbf{y}_i \in \mathbf{s}_2$ and $\mathbf{x}_{i^\star} \in \mathbf{s}_1$ using a line perpendicular to the contour at $\mathbf{y}_i \in \mathbf{s}_2$. Let us represent the line tangent to the curve at the point $\mathbf{y}_i \in \mathbf{s}_2$ as $\mathcal{L} = \{\mathbf{y}_{i-1} + t(\mathbf{y}_{i+1} - \mathbf{y}_{i-1}) | t \in \Re\} = \{\mathbf{y} | \mathbf{a}^\top \mathbf{y} + b = 0\}$ with $\mathbf{a}^\top(\mathbf{y}_{i+1} - \mathbf{y}_{i-1}) = 0$ and $b = -\mathbf{a}^\top \mathbf{y}_{i-1}$. Let us also denote the curve sampled at points $\mathbf{s}_1 = [\mathbf{x}_i^\top]_{i=1..N}$ with the following implicit representation: $f(\mathbf{x}, \theta_{\mathbf{s}_1}) = 0$, where $\theta_{\mathbf{s}_1}$ denotes the parameters of this representation. Hence, we can find the point $\mathbf{x}_{i^\star} = \arg\min_{\mathbf{x} \in \mathbf{s}_1} (\|\mathbf{x} - (s^* \mathbf{a} + \mathbf{y}_i)\|_2$, where $s^* = \arg\min s$ subject to $f(s\mathbf{a} + \mathbf{y}_i, \theta_{\mathbf{s}_1}) = 0$. The AVP error measure is defined as

$$d_{\mathrm{AVP}}(\mathbf{s}_1, \mathbf{s}_2) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_{i^\star} - \mathbf{y}_i\|, \tag{23}$$

B. Comparison with the State of the Art

The performance of our system is compared against the performance of two state-of-the-art approaches on the test set described in Sec. IV-B with respect to the error measures defined in (18)-(23). The method proposed by Nascimento and Marques [36] is a model-based approach that uses common assumptions on the LV segmentation, such as that the borders are well represented with edge information, the contour is smooth and the LV has a prior shape. The method proposed by Comaniciu and colleagues [8,10] is a pattern recognition approach relying on a quite large annotated training set (they have in the order of thousands of annotated images), using discriminative classifier based on boosting techniques. Note that the original algorithms from both authors have been run on the test set.

The objectives of our comparison against these two methods is threefold. The first goal is to show that our method achieves competitive results in this quite challenging test set (displayed in Fig. 7), especially when compared with the training images, which present rather different image statistics (in terms of texture, views, etc.). Recall that we have 400 annotated images for training our system, which is an order of magnitude smaller than what is available for Comaniciu's method [8,10], but in spite of that, our system presents comparable or better results with respect to most of error measures. The second goal is to demonstrate the ability of deep belief networks to generalize well with quite small data sets; that is, the performance of the method is quite stable with small training sets. For instance we randomly used only a subset of the 400 training images to train the DBN classifier, where the subset size varies from

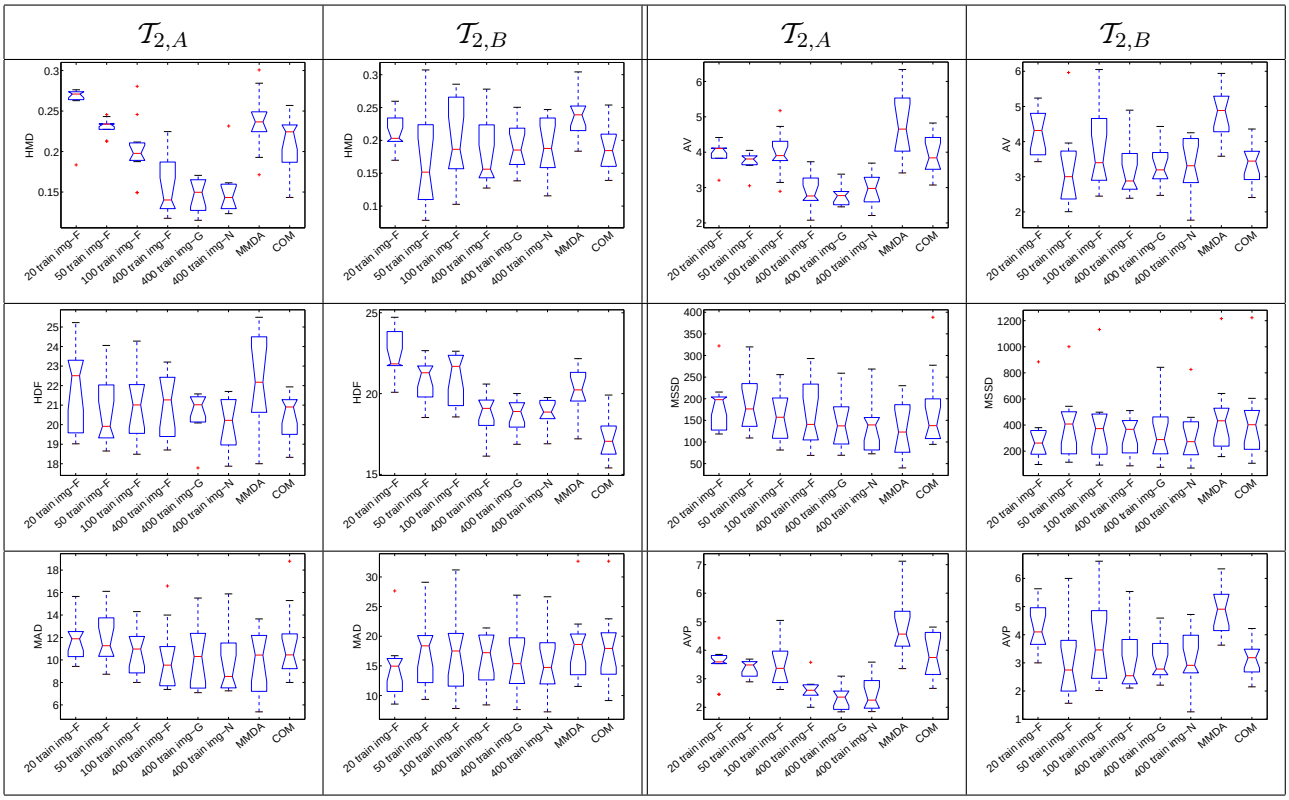


Fig. 11. Box plot results for all error measures explained in Sec. V-A (the measures are denoted in the vertical axis of each graph). Using the test sequences $\mathcal{T}_{2,A}$ (columns 1 and 3) and $\mathcal{T}_{2,B}$ (columns 2 and 4), we compare the results of our method with varying training set sizes ('$\{20, 50, 100, 400\}$ train img-F') using the full search method, and with efficient search methods ('400 train img-G' for gradient descent and '400 train img-N' for Newton's method), Nascimento and Marques' approach [36]('MMDA'), and Comaniciu and colleagues' method ('COM') [8,10].

$\{20, 50, 100, 400\}$, which means *one to two orders of magnitude* less training images than Comaniciu's method [8,10]. The last goal is to verify the performance of the gradient descent and Newton's method when compared with the full search approach. For this last experiment, we use the DBN classifier trained with 400 images.

Figure 11 shows the error measures (18)-(23) in sequences $\mathcal{T}_{2,\{A,B\}}$ using box plot graphs of Comaniciu's method [8,10] (see label 'COM'), Nascimento's approach [36] (label 'MMDA'), and our approach with different training set sizes (labels '$\{20, 50, 100, 400\}$ train img') and the three different search approaches (labels 'F' for the full search, 'G' for the gradient descent search and 'N' for the Newton's method search). Figure 12 displays two results for each sequence produced by our method (using the '400 train img-F'). For all error measures, we can see that the method '400 train img-F' is either comparable or

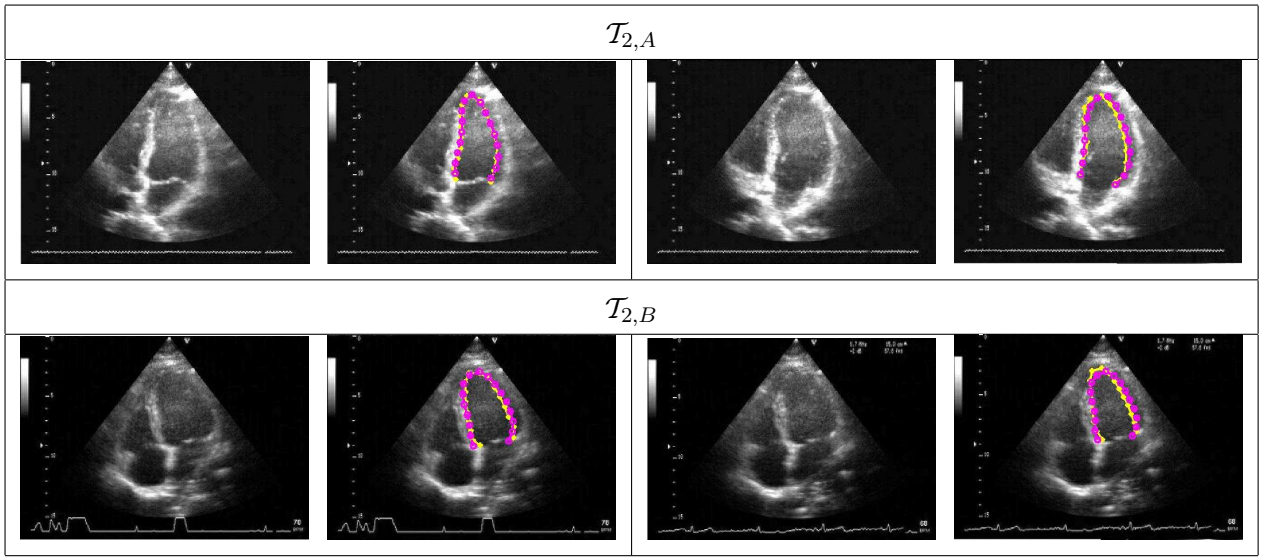


Fig. 12. Examples of detection on test sequences. The yellow, solid annotation denotes the manual annotations, while the magenta dashed line shows the results from our system.

better than the best of 'COM' and 'MMDA'. For the '$\{20, 50, 100\}$ train img-F' methods, the results are always comparable to best results of '400 train img-F', 'COM, and 'MMDA'. Finally, the more efficient search methods of '400 train img-G' and '400 train img-N' show slightly worse results than '400 train img-F', but they are still comparable to the best results in the experiment.

C. Comparison with Inter-user Statistics

For the quantitative comparison we follow the methodology proposed by Chalana and Kim [15], and revised by Lopez *et al.* [54]. We briefly show how the gold standard is calculated based on the manual segmentations, then we describe the *modified Williams index* and the *Percent statistics*. We show additional results with scatter plots and the Bland-Altman graph [55]. These evaluations compare the computer generated segmentation to the multiple users' segmentations. Note that in all these comparisons we used the DBN classifier '400 train img-F' classifier from Fig. 11.

1) Averaging Curves: The gold standard contour is based on the average curve as proposed by Chalana and Kim [15]. Let $\{\mathbf{s}_1, ..., \mathbf{s}_U\}$ be the annotations from U users. The averaging process is as follows:

i) Set $t = 0$, and compute the average curve $\bar{\mathbf{s}}^{(t)}$ in the set $\{\mathbf{s}_1, ..., \mathbf{s}_U\}$

ii) From each point $\mathbf{x}_i \in \bar{\mathbf{s}}^{(t)}$, a normal to the curve at that point is drawn. The intersection of this normal with each of the U input curves is determined.

iii) The points of intersection define a new set of correspondences between the U curves.

iv) Set $t = t + 1$.

v) The new correspondences are averaged again to give another averaged curve, say $\overline{\mathbf{s}}^{(t)}$.

vi) Repeat steps 2-4 until $\|\mathbf{s}^{(t)} - \mathbf{s}^{(t-1)}\|_2$ is below a certain threshold.

2) Training and testing for Inter-user Statistics: Recall from Sec. IV-B that for sequences $\mathcal{T}_{1,\{A,B,C\}}$ we have two LV manual annotations produced by two different Cardiologists. In this experiment, we have an average of 17 images annotated for each sequence, so in total we have 50 images annotated by two experts. We then train three separate DBN classifiers using the following training set for each one: 1) $\mathcal{T}_1 \setminus \mathcal{T}_{1,A}$, 2) $\mathcal{T}_1 \setminus \mathcal{T}_{1,B}$, 3) $\mathcal{T}_1 \setminus \mathcal{T}_{1,C}$, where $\setminus$ represents the set difference operator. These three classifiers are necessary because when testing any image inside each one of these three sequences, we cannot use any image of that same sequence in the training process.

3) Williams index: Let M be a collection of images, for which there are U manual annotations available. Assume that we have a set $\{\mathbf{s}_{i,j}\}$, where $i \in \{1..M\}$ and $j \in \{0..U\}$ indexes the image and the manual annotations, respectively. The index $j = 0$ denotes the computer-generated contour.

The function $D_{j,j'}$ measures the disagreement between users j and j', and it is defined as

$$D_{j,j'} = \frac{1}{M} \sum_{i=1}^{M} d_{-}(\mathbf{s}_{i,j}, \mathbf{s}_{i,j'}), \tag{24}$$

where $d_{-}(.,.)$ is an error measure between two annotations $\mathbf{s}_{i,j}$, $\mathbf{s}_{i,j'}$, which can be any of the measures defined previously in (18)-(23). The modified Williams index is defined as

$$I^{'} = \frac{\frac{1}{U} \sum_{j=1}^{U} \frac{1}{D_{0,j}}}{\frac{2}{U(U-1)} \sum_{j} \sum_{j':j' \neq j} \frac{1}{D_{j,j'}}}. \tag{25}$$

A confidence interval (CI) is estimated using a jackknife (leave one out) non-parametric sampling technique [15] as follows:

$$I^{'}_{(.)} \pm z_{0.95} se, \tag{26}$$

where $z_{0.95} = 1.96$, representing 95^{th} percentile of the standard normal distribution,

$$se = \left\{ \frac{1}{M-1} \sum_{i=1}^{M} [I^{'}_{(i)} - I^{'}_{(.)}] \right\} \tag{27}$$

with $I^{'}_{(.)} = \frac{1}{M} \sum_{i=1}^{M} I^{'}_{(i)}$, and $I^{'}_{(i)}$ is the Williams index (25) calculated by leaving image i out of computation of $D_{j,j'}$. A successful measurement for the Williams index is to have the average and confidence interval (26) close to one. Table II shows the average and confidence intervals for all ultrasound sequences considered in this section.

TABLE II

COMPARISON OF THE COMPUTER GENERATED CURVES TO THE USERS' CURVES WITH RESPECT TO ALL THE ERROR MEASURES FOR THREE SEQUENCES USING THE AVERAGE AND 0.95% CONFIDENCE INTERVAL (IN PARENTHESIS) OF THE WILLIAMS INDEX.

measure	d_{HMD}	d_{AV}	d_{HDF}	d_{MSSD}	d_{MAD}	d_{AVP}
Average	0.80	0.94	0.91	0.70	0.86	0.95
(CI)	$(0.78, 0.81)$	$(0.93, 0.95)$	$(0.90, 0.92)$	$(0.68, 0.72)$	$(0.85, 0.88)$	$(0.94, 0.97)$

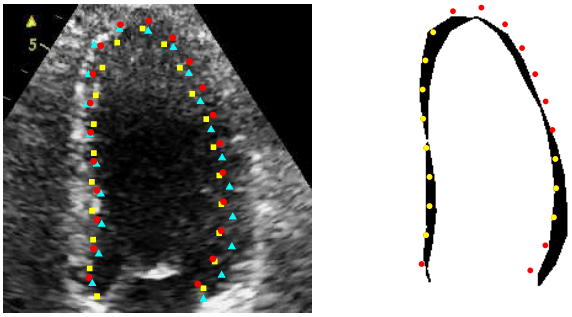


Fig. 13. (Left) Three contours drawn in an ultrasound image, where the yellow (square) and cyan (triangle) are the manual contours, and the red (circle) contour represents the computer-generated segmentation. (Right) The convex hull formed by the manual contours is shown, and the computer generated points are shown in either red (darker markers) or yellow (lighter markers), representing the cases where the points lie outside or inside the convex hull, respectively.

4) Percent statistics: The second statistical technique computes the percentage of cases for which the computer-generated measurement lies within the inter-user range (see Fig. 13). The expected value for the percent statistics depends on the number of manual curves. Following Lopez *et al.* [54], who revised this value from Chalana and Kim [15], the successful expected value for the percent statistic should at least $\frac{U-1}{U+1}$, where U is the number of manual curves. In our case, $U = 2$, so the expected value for the percent statistic should be at least 33%, and the confidence interval must contain 33%. For the images considered in this section, we obtained a percent statistics of average 35.2% and confidence interval $(2.6\%, 67.8\%)$.

5) Bland-Altman and Scatter plots: We also present quantitative results using the Bland-Altman [55] and scatter plots (from which it is possible to compute a linear regression, the correlation coefficient and the p-value). To accomplish this we have: (i) the gold standard LV volume (computed from the curve average of Sec. V-C.1); (ii) the Cardiologists' LV volumes, and (iii) the computer generated LV volume.

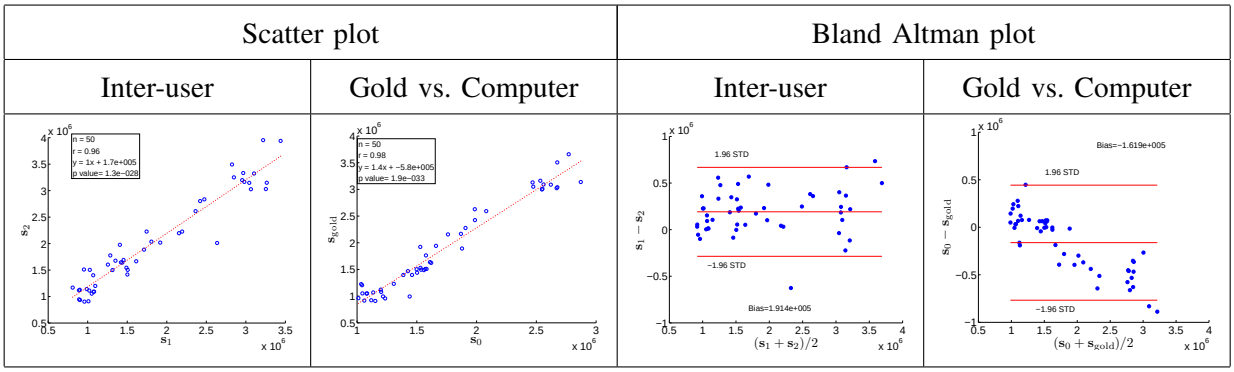


Fig. 14. Scatter plots with linear regression and Bland-Altman bias plots

To estimate the LV volume from 2-D contour annotation we the Area Length method [56].

Volumes are calculated using the area-length equation introduced by Sandler and Dodge [57] (see also [58]), i.e., $V = \frac{8A^2}{3\pi L}$, where A denotes the projected surface area, L is the distance from upper aortic valve point to apex, and V is expressed in pixels^3.

Figure 14 shows the scatter and Bland-Altman plots for the three sequences considered in this section. For the scatter plot, notice that the correlation coefficient between the users is 0.96 (p-value = 10^{-28}) and for the gold standard versus computer the correlation is 0.98 (p-value = 10^{-33}). In the Bland-Altman plots, our method shows similar bias (in absolute value) and standard deviation when compared with the users' inter-variation.

D. Run-time Complexity

Given the figures presented in Sec. IV-D.1, the full search average complexity is around 2 to 10 times bigger than the gradient descent search and 1.5 to 5 times bigger than the Newton step search. In practice, using a non-optimized Matlab implementation, the full search takes around 20 seconds to run, and gradient descent and Newton's method search run in between 5 to 10 seconds on a typical laptop computer with the following configuration: Intel Centrino Core Duo (32 bits) at 2.5GHz with 4GB of memory. We expect that an efficient C implementation exploiting code parallelization can reduce the running time of the all search methods to well under 1 second on a typical Quad core desktop computer (64 bits) with 8 GB of memory.

VI. Conclusion and Future Work

We presented a new pattern recognition approach for the problem of fully automatic LV detection and segmentation using ultrasound data. Our main contribution is the use of deep learning architectures that

produces quite competitive segmentation results using up to two orders of magnitude less training data than current state-of-the-art pattern recognition approaches. Another contribution is the use of efficient search methods to speed up the segmentation process without causing significant negative impact on the accuracy of the method. We present extensive quantitative evaluations that show that our system is competitive and produces acceptable results in terms of inter-user variations. In the future we plan to include a dynamical model to improve even more the performance of the algorithm.

Acknowledgments: We would like to thank G. Hinton and R. Salakhutdinov for making the deep belief network code available on-line. We also would like to thank Dr. José Morais for providing the manual LV annotations.

REFERENCES

[1] J. K. Oh, J. B. Seward, and A. J. Tajik, *The Echo Manual, 2nd ed.* Philadelphia, PA: Lippincott-Raven, 1999.

[2] R. M. Lang and *et al.*, "Recommendations for chamber quantification," *Eur. J. Echocardiography, Elsevier*, vol. 24, no. 7, pp. 79–108, 2006.

[3] J. G. Bosch, S. C. Mitchell, B. P. F. Lelieveldt, F. Nijland, O. Kamp, M. Sonka, and J. H. C. Reiber, "Automatic segmentation of echocardiographic sequences by active appearance motion models," *IEEE Trans. Med. Imag.*, vol. 21, no. 11, pp. 1374–1383, 2002.

[4] M. Sonka, X. Zhang, M. Siebes, M. Bissing, S. Dejong, S. Collins, and C. Mckay, "Segmentation of intravascular ultrasound images: A knowledge-based approach," *IEEE Trans. Med. Imag.*, vol. 14, pp. 719–732, 1995.

[5] L. Zhang and E. Geiser, "An effective algorithm for extracting serial endocardial borders from 2-d echocardiograms," *IEEE Trans. Biomed. Eng.*, vol. BME-31, pp. 441–447, 1984.

[6] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 4, no. 1, pp. 321–331, 1987.

[7] N. Paragios, "A level set approach for shape-driven segmentation and tracking of the left ventricle," *IEEE Trans. Med. Imag.*, vol. 22, no. 6, pp. 773–776, 2003.

[8] D. Comaniciu, X. Zhou, and S. Krishnan, "Robust real-time myocardial border tracking for echocardiography: An information fusion approach," *IEEE Trans. Med. Imag.*, vol. 23, no. 7, pp. 849–860, 2004.

[9] T. Cootes and C. Taylor, "Statistical models of appearance for computer vision," Univ. Manchester, Div. Imag. Sci. Biomed. Eng., Manchester, Tech. Rep., 2004.

[10] B. Georgescu, X. S. Zhou, D. Comaniciu, and A. Gupta, "Databased-guided segmentation of anatomical structures with complex appearance," in *Conf. Computer Vision and Pattern Rec. (CVPR)*, 2005.

[11] Y. Zheng, A. Barbu, B. Georgescu, M. Scheuering, and D. Comaniciu, "Four-chamber heart modeling and automatic segmentation for 3-D cardiac CT volumes using marginal space learning and steerable features," *IEEE Trans. Med. Imaging*, vol. 27, no. 11, pp. 1668–1681, 2008.

[12] S. Zhou and D. Comaniciu, "Shape regression machine," in *IPMI*, 2007, pp. 13–25.

[13] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, March 2004.

[15] V. Chalana and Y. Kim, "A methodology for evaluation of boundary detection algorithms on medical images," *IEEE Trans. Med. Imag.*, vol. 16, no. 10, 1997.

[16] G. Carneiro, J. C. Nascimento, and A. Freitas, "Robust left ventricule segmentation from ultrasound data using deep neural networks and efficient search methods," in *Int. Symp. Biomedical Imaging: from nano to macro (ISBI)*, 2010.

[17] T. Chen, J. Babb, P. Kellman, L. Axel, and D. Kim, "Semiautomated segmentation of myocardial contours for fast strain analysis in cine displacement-encoded MRI," *IEEE Trans. Med. Imag.*, vol. 27, no. 8, pp. 1084–1094, 2008.

[18] C. Corsi, G. Saracino, A. Sarti, and C. Lamberti, "Left ventricular volume estimation for real-time three-dimensional echocardiography," *IEEE Trans. Med. Imag.*, vol. 21, no. 9, pp. 1202–1208, 2002.

[19] E. Debreuve, M. Barlaud, G. Aubert, I. Laurette, and J. Darcourt, "Space-time segmentation using level set active contours applied to myocardial gated SPECT," *IEEE Trans. Med. Imag.*, vol. 20, no. 7, pp. 643–659, 2001.

[20] M. Lynch, O. Ghita, and P. F. Whelan, "Segmentation of the left ventricle of the heart in 3-D+t MRI data using an optimized nonrigid temporal model," *IEEE Trans. Med. Imag.*, vol. 27, no. 2, pp. 195–203, 2008.

[21] O. Bernard, B. Touil, A. Gelas, R. Prost, and D. Friboulet, "A rbf-based multiphase level set method for segmentation in echocardiography using the statistics of the radiofrequency signal," in *ICIP*, 2007.

[22] N. Lin, W. Yu, and J. Duncan, "Combinativemulti-scale level set framework for echocardiographic image segmentation," *Medical Image Analysis*, vol. 7, no. 4, pp. 529–537, 2003.

[23] A. Sarti, C. Corsi, E. Mazzini, and C. Lamberti, "Maximum likelihood segmentation of ultrasound images with rayleigh distribution," *IEEE T. on Ult., Fer. and F.C.,*, vol. 52, no. 6, pp. 947–960, 2005.

[24] S. Mitchell, B. Lelieveldt, R. van der Geest, H. Bosch, J. Reiber, and M. Sonka, "Multistage hybrid active appearance model matching: Segmentation of left and right ventricles in cardiac MR images," *IEEE Trans. Med. Imag.*, vol. 20, no. 5, pp. 415–423, 2001.

[25] V. Zagrodsky, V. Walimbe, C. Castro-Pareja, J. X. Qin, J.-M. Song, and R. Shekhar, "Registration-assisted segmentation of real-time 3-D echocardiographic data using deformable models," *IEEE Trans. Med. Imag.*, vol. 24, no. 9, pp. 1089–1099, 2005.

[26] G. Jacob, J. A. Noble, C. Behrenbruch, A. D. Kelion, and A. P. Banning, "A shape-space-based approach to tracking myocardial borders and quantifying regional left-ventricular function applied in echocardiography," *IEEE Trans. Med. Imag.*, vol. 21, no. 3, pp. 226–238, 2002.

[27] A. Ng and M.Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *NIPS*, 2002.

[28] T. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active shape models - their training and application," *Comput. Vis. Image Understand.*, vol. 61, no. 1, pp. 38–59, 1995.

[29] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 681–685, 2001.

[30] ——, "Active appearance models," in *Eur. Conf. Comp. Vis.*, 1998, pp. 484–498.

[31] T. Cootes, C. Beeston, G. Edwards, and C. Taylor, "A unified framework for atlas matching using active appearance models," in *Information Processing in Medical Imaging*, 1999, pp. 322–333.

[32] R. Beichel, H. Bischof, F. Leberl, and M. Sonka, "Robust active appearance models and their application to medical image analysis," *IEEE Trans. Med. Imag.*, vol. 24, no. 9, pp. 1151–1169, 2005.

[33] H. Zhang, A. Wahle, R. K. Johnson, T. D. Scholz, and M. Sonka, "4-D cardiac mr image analysis: Left and right ventricular morphology and function," *IEEE Trans. Med. Imag.*, vol. 29, no. 2, pp. 350–364, 2010.

[34] E. Oost, G. Koning, M. Sonka, P. V. Oemrawsingh, J. H. C. Reiber, and B. P. F. Lelieveldt, "Automated contour detection in X-ray left ventricular angiograms using multiview active appearance models and dynamic programming," *IEEE Trans. Med. Imag.*, vol. 25, no. 9, pp. 1158–1171, 2006.

[35] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[36] J. C. Nascimento and J. S. Marques, "Robust shape tracking with multiple models in ultrasound images," *IEEE Trans. Imag. Proc.*, vol. 17, no. 3, pp. 392–406, 2008.

[37] G. Carneiro, B. Georgescu, S. Good, and D. Comaniciu, "Detection and measurement of fetal anatomies from ultrasound images using a constrained probabilistic boosting tree," *IEEE Trans. Med. Imaging*, vol. 27, no. 9, pp. 1342–1355, 2008.

[38] J. Koikkalainen, T. Tölli, K. Lauerma, K. Antila, E. Mattila, M. Lilja, and J. Lötjönen, "Methods of artificial enlargement of the training set for statistical shape models," *IEEE Trans. Med. Imag.*, vol. 27, no. 11, pp. 1643–1654, 2008.

[39] R. Bartels, J. Beatty, and B. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.

[40] G. Hinton. http://videolectures.net/nips09_hinton_dlmi/.

[41] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, no. 323, pp. 533–536, 1986.

[42] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[43] R. Salakhutdinov and G. Hinton, "Learning a non-linear embedding by preserving class neighbourhood structure," in *AI and Statistics*, 2007.

[44] M. Carreira-Perpinan and G. Hinton, "On contrastive divergence learning," in *Workshop on Artificial Intelligence and Statistics*, 2005.

[45] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[46] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Conf. Computer Vision and Pattern Rec. (CVPR)*, 2001, pp. 511–518.

[47] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. John Wiley and Sons, 2001.

[48] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Elsevier, 1990.

[49] A. Hammoude, "Computer-assited endocardial border identification from a sequence of two-dimensional echocardiographic images," Ph.D. dissertation, University Washington, 1988.

[50] D. Huttenlocher, G. Klanderman, and W. Rucklidge, "Comparing images using hausdorff distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 9, pp. 850–863, 1993.

[51] X. S. Zhou, D. Comaniciu, and A. Gupta, "An information fusion framework for robust shape tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 1, pp. 115–129, 2005.

[52] Y. Akgul and C. Kambhamettu., "A coarse-to-fine deformable contour optimization framework," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 2, pp. 174–186, 2003.

[53] I. Mikić, S. Krucinki, and J. D. Thomas, "Segmentation and tracking in echocardiographic sequences: Active contours guided by optical flow estimates," *IEEE Trans. Med. Imag.*, vol. 17, no. 2, pp. 274–284, 1998.

[54] C. Alberola-Lopez, M. Martin-Fernandez, and J. Ruiz-Alzola, "Comments on: A methodology for evaluation of boundary detection algorithms on medical images," *IEEE Trans. Med. Imag.*, vol. 23, no. 5, pp. 658–660, 2004.

[55] J. Bland and A. Altman, "Statistical methods for assessing agreement between two methods of clinical measurement," *Lancet*, vol. 1, no. 8476, pp. 307–310, 1986.

[56] A. Parisi, P. Moynithan, C. Feldman, and E. Folland, "Approaches to determination of left ventricular volume and ejection fraction by real-time two-dimensional echocardiography," *Clin. Cardiology*, vol. 12, no. 2, pp. 855–867, 1979, g. Witzstrock Publishing House Inc.

[57] H. Sandler and H. T. Dodge, "The use of single plane angiocardiograms for the calculation of left ventricular volume in man," *Amer. Heart J.*, vol. 75, no. 3, pp. 325–334, 1968.

[58] J. C. Reiber, A. R. Viddeleer, G. Koning, M. J. Schalij, and P. E. Lange, "Left ventricular regression equations from single plane cine and digital X-ray ventriculograms revisited," *Clin. Cardiology*, vol. 12, no. 2, pp. 69–78, 1996, kluwer Academic Publishers.