Proceedings of the International Conference on Statistics: Theory and Applications (ICSTA'19)
Lisbon, Portugal August 13 - 14, 2019
Paper No. ICSTA 24
DOI: 10.11159/icsta19.24

Massive Dimensionality Reduction for the Left Ventricular Mesh

Lukasz Romaszko[1], Alan Lazarus[1], Hao Gao[1], Agnieszka Borowska[1], Xiaoyu Luo[1], Dirk Husmeier[1]
[1]School of Mathematics and Statistics, University of Glasgow,
Glasgow G12 8SQ, UK
lukasz.romaszko@glasgow.ac.uk; a.lazarus.1@research.gla.ac.uk; hao.gao@glasgow.ac.uk;
agnieszka.borowska@glasgow.ac.uk; xiaoyu.luo@glasgow.ac.uk; dirk.husmeier@glasgow.ac.uk

Abstract - Statistical emulation is a promising approach for the translation of cardio-mechanical modelling into the clinical practice. However, a key challenge is to find a low-dimensional representation of the heart, or, for the specific purpose of diagnosing the risk of heart attacks, the left-ventricle of the heart. We consider the problem of dimensionality reduction of the left ventricular mesh, in which we investigate three classes of techniques: principal component analysis (PCA), deep learning (DL) methods based on auto-encoders, and a parametric model from the cardio-mechanical literature. Our finding is that PCA performs as well as the computationally more expensive DL methods, and both outperform the state-of-the-art parametric model.

Keywords: Dimensionality reduction, principal component analysis, deep learning, auto-encoders, left ventricular mesh.

1. Introduction

One central problem in biomechanical studies of personalised human left ventricle (LV) modelling is to estimate material properties from patient specific geometries [1]. A classical approach is to formulate an inverse problem by simulating a forward computational model to minimise the differences between the simulated and measured quantities [2]. Over decades, finite element method (FEM) has been widely used for developing the forward computational models, however, one disadvantage of a LV model based on FEM is that it can require high computational resources, and may take weeks for inferring material properties due to the need of repeatedly solving a FEM model numerically [2]. This has hindered the application of using material properties in real-time clinical decision support systems.

In a proof-of-concept study [3], we have developed a novel Gaussian process emulator to accelerate the inference of material properties using in vivo measurements. This approach allowed us to reduce the computational costs of inference by about three orders of magnitude. However, the emulator was built on a geometry of a representative healthy subject, which limits its application to different subjects. Therefore, there is a need to develop an emulation framework taking into account subject-specific geometries. In a FEM LV model, the 3D LV geometry is approximated by a network consisting of thousands of tetrahedrons or hexhadrons. For example, in this study, the LV geometry is represented by a mesh from a 17k dimensional space. This is prohibitive for an emulator such as in [3]. Therefore, there is a need to identify efficient methods for low-dimensional representation of the LV mesh (LVM). There are various approaches to this, such as principle component analysis (PCA), deep learning methods based on auto-encoders or the simplified parametric representation used in [4], while an unanswered question is which approach works better for the dimensionality reduction of the LVM.

The outline of this paper is as follows. Section 2 describes our mesh dataset constructed from CMR scans. In Section 3 we discuss three types of methods we use for dimensionality reduction of the LVM, PCA, methods based on auto-encoders and the parametric model of [4]. Section 4 presents the results of our analysis comparing PCA to both alternative approaches. We conclude the paper in Section 5 with a discussion and a brief outlook on future work.

2. Data

We collected CMR scans of 70 healthy volunteers (HVs) and 128 patients with acute myocardial infarction (MI), in total 198 subjects. MI patients were identified from the British Heart Foundation MR-MI study population (ClinicalTrials.gov identifier: NCT02072850) from a population of patients with acute STEMI, who had been enrolled into a CMR cohort study between July 14, 2011 and November 22, 2012. CMR was performed at 1.5 Tesla (Siemens Avanto, Siemens Healthcare, Erlangen, Germany). For MI patients the scans were taken within 2 days after hospital admission. MI patients and HVs underwent the same imaging protocol except that HVs below the age of 45 did not receive intravenous gadolinium contrast.

The CMR protocol involved steady-state free precession cine scans with a short-axis left ventricular stack from the base to the apex. The slice thickness was 7 mm with a 3mm gap, typical cine imaging parameters were matrix = 180×256, flip angle = 80^o, TR: 3.3 ms, TE: 1.2 ms, bandwidth: 930 Hz/pixel. The voxel size was $1.32 \times 1.32 \times 7\,\text{mm}^3$. Cine images were acquired in the three-chamber, horizontal long-axis, and vertical long-axis planes.

We used short-axis (SA) and long-axis (LA) cine CMR images at early-diastole to generate our final dataset of 3-dimensional LVMs. LV wall boundaries were manually segmented at each time instance by using an in-house `MATLAB` code, and the SA LV wall boundaries were further aligned to the boundaries in LA images. Segmented LV boundaries were then fitted to a prolate spheroidal B-spline described template LV mesh with one layer of hexahedron elements across the wall. The final mesh consists of 2×2896 = 5792 vertices. As there are three x-y-z coordinates per vertex, each mesh has 3×5792 = 17368 values, hence the original input is denoted by “17k”.

3. Methods

We consider three types of methods for dimensionality reduction: PCA, AE, and the parametric model. A natural baseline for the reconstruction assessment is provided by the mean mesh (MM), which always predicts the average mesh in our sample. To measure the methods' performances we use the mean reconstruction error, computed as the average mean square error (MSE) between a test LVM in the dataset and its reconstruction.

3.1. Principal Component Analysis

We first consider the standard principal component analysis (PCA), which has several advantages. First, it is a popular method, known to non-statisticians, e.g. medical practitioners. Second, it is fast as it does not require iterative training. Third, it can be performed on any standard, personal computer, so no expensive hardware such as GPU (graphical processing unit) is necessary. It has, though, important limitations, as it is a linear mapping, constraining the components to be orthogonal.

Since our dataset consists of two classes of subjects, HVs and MI patients, we can expect that there might be nonlinearities in the relationships between unobserved components characterising both groups. In such a case, using mixtures of probabilistic PCA (MPPCA), see [5], might be advantageous over using the basic version of PCA in terms of potentially providing a better fit to the data. An obvious shortcoming of MPPCA for our purposes would be the need of including the mixture component index as an additional input variable in the emulator and the resulting discontinuity in the emulator's parameter space. Thus, an important question to answer is whether there is a potential gain from using a more flexible but more complex method. To this end, we need to investigate whether there is a clear separation between both subject classes in the PCA mapping space.

3.2. Auto-encoders and Deep Learning

Given the complex 3D shape of the LVM, mapping into a linear subspace provided by PCA might be too restrictive. Thus, we next consider nonlinear methods from deep learning (DL) based on auto-encoders (AE).

3.2.1. Neural Network Inputs and Architectures

Since we want the neural networks to reduce the dimensionality of LVMs to 4 dimensions, as per PCA, we set the bottleneck of each network to 4 neurons. We investigate two types of input to networks, the original, high-dimensional one and the reduced one, obtained via initial dimensionality reduction with PCA. The rationale behind introducing the latter input is that learning the network weights with the original input is cumbersome due to the large size of the observations in that dataset. With 17k-dimensional objects, setting 100 neurons in the intermediate hidden layer (see Table 1) leads to 1.7mln weights to learn for the encoder, which, together with the decoder of the same size, results in 3.4 mln weights in total. Using PCA to simplify the input allows us to enhance training of the network, as we feed it with an input of much lower dimension than that of the original dataset, yet at the same time it does not compromise the encoding accuracy as the reduced input can still closely represent the original LVMs. As our experiments with PCA indicate, it is enough to keep the first 60 principal components to explain over 99.9% of the original data variance. Hence, the reduced input consists of 60-dimensional objects.

We consider two main types of network architectures: shallow linear auto-encoders (AEs) and deeper nonlinear AEs, both with the two kinds of input discussed above. Figure 1 presents the architecture scheme of the deep AEs. The shallow AEs directly encode the input into 4 values, so the only hidden layer is the bottleneck. The deep AEs use an intermediate layer

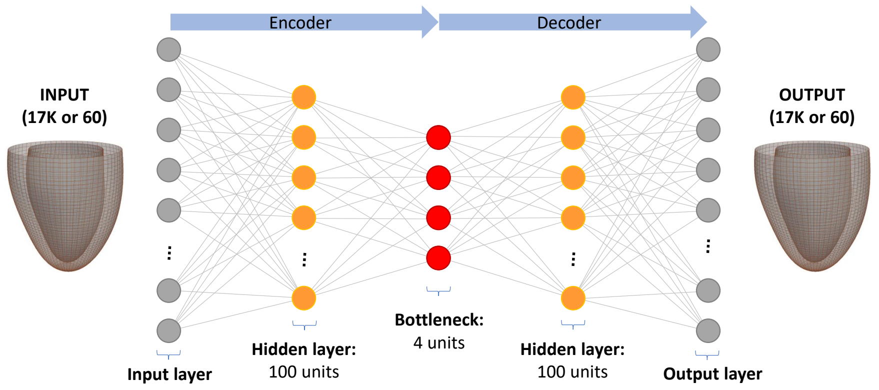


Fig. 1: Architecture scheme of the auto-encoder (AE) networks, with 5 layers (input, hidden, bottleneck, hidden, output. Shallow AEs skip the two intermediate hidden layers (denoted yellow).

of hidden neurons, which results in a 5-layer NN, i.e: input layer, hidden layer, bottleneck layer, hidden layer, output. As the activation function mapping from the input layer to hidden layer, we use the hyperbolic tangent function (tanh) and the leaky rectified linear unit function (ReLU), given by $x = \max(0.2x, x)$.

We denote the resulting architectures by providing the input size, the name of the activation function (if present) and the bottleneck size (common for all the AE). Hence, the name describes the encoder only, since the decoder is symmetric.

To facilitate network training and accelerate convergence, we rescaled both input types prior to training. For the original input, we divided its standardised version by 10. For the reduced input, we divided the PCA coefficients by 100 as high values were not allowing the tanh networks to converge. Consequently, the direct outputs form our AEs need to be scaled back to represent the source input.

3.2.2. Training Details and Hyperparameter Search

Due to a relatively small number of observations (<200) we use one batch consisting of the whole training dataset. We use the Adam optimiser [6] to learn the weights and biases of each AE. The GPU computations were performed on a NVIDIA Quadro P4000. Dropout, i.e. turning off a subset of neurons at random at each evaluation, is a standard practice to prevent overfitting in neural networks, see [7]. We experimented with dropout, but as it turned out not to be helpful overall, we did not use it eventually.

We trained the networks for different configurations of two key hyperparameters: L2-regularisation value (weight decay factor) and learning rate (LR). These values were optimised with leave-one-out cross-validation (1-out CV). Since these values are shared across all 198 networks, we did not face the problem of overfitting to a particular example.

The L2-regularisation parameter is multiplied by the total sum of squares of all the weights. We have 198 networks and we searched for the best performing L2-regularisation parameter in the set $[1, 3] \times 10^Z, Z \in [-8, \dots, -3]$.

The speed of convergence in a given network depends on its architectures, the dimension of the input layer and the chosen L2-regularisation parameter. Since we consider 6 different architecture-input network types, for each different network we chose LR based on the additional validation split (with train:validation sizes being $2 : 1$ and the leave-one-out example being excluded). We chose LR so that the error for the validation split is the lowest on average for all the 198 networks in approx. 600 epochs. This allows us to compare different AE architectures while keeping the number of learning updates fixed. We explored values of LR $\in [1, 3, 5] \times 10^Z$, $Z \in [-5, -4, -3, -2]$. We trained each network for the number of epochs at which the result on the validation set was the best, using both the training and validation splits, and testing on the remaining example.

Table 1 shows the optimal hyperparameters used for different architectures. Interestingly, AEs with 60 inputs worked well for the same L2 parameter as AEs with 17k input. This can be explained by different magnitudes of both input types: 17k inputs are much larger than 60 principal components. If the scales of both inputs were similar, the appropriate L2-parameter would be approx. 300 times lower for the 17k network, as there are 300 times more weights. However, 17k input *values* are larger, which coincidentally counterbalances their greater quantity.

Table 1: Hyperparameters of AEs: LR and L2 parameters. AEs are sorted by the input size.

Method	60-4	60-[100:tanh]-4	60-[100:ReLU]-4	17k-4	17k-[100:tanh]-4	17k-[20:ReLU]-4
LR	0.005	0.001	0.0003	0.0005	0.0001	0.0001
L2		0.00001			0.00001	

3.3. Parametric Model

Next to dimensionality reduction techniques from statistics and DL, we consider a benchmark parametric method from the biomechanical literature, proposed in [4]. This symmetric model describes the LV geometry using 6 parameters , which we collect in the vector $\boldsymbol{\xi}$. For each of 198 LVMs in our dataset, we fitted the model twice, by minimising two loss functions with respect to the parameter vector $\boldsymbol{\xi}$. These loss functions belong to the intersection-over-union (IOU) class and are based on the volume of the intersections of the predicted (given the parameter $\boldsymbol{\xi}$) wall $W_{\boldsymbol{\xi}}$ and cavity $C_{\boldsymbol{\xi}}$ with the corresponding volumes in the true LVM (see Section 2), denoted W_T and C_T, respectively. Our two objective functions are then

$$E_{WC} = 1 - \frac{1}{2}\left\{\frac{W_{\boldsymbol{\xi}} \cap W_T}{W_{\boldsymbol{\xi}} \cup W_T} + \frac{C_{\boldsymbol{\xi}} \cap C_T}{C_{\boldsymbol{\xi}} \cup C_T}\right\}, \quad E_W = 1 - \frac{W_{\boldsymbol{\xi}} \cap W_T}{W_{\boldsymbol{\xi}} \cup W_T}, \tag{1}$$

where E=0 is the optimal LV alignment. The objective function on the left hand side in (1) is the original objective used by [4] and takes into account both cavity and wall volumes. The right-hand side function in (1) is our new loss function based on the wall volume only, which we found to outperform the original objective in some of our simulations.

For optimisation, parameters were initialised at their measured values except the conicity parameter (one of the parameters in $\boldsymbol{\xi}$, see [4]) which was drawn from $\mathcal{U}[0.5, 1]$. Optimisations were done in `MATLAB`[1], where the volumes were evaluated by set allocation of a grid of points. We use a grid of equally spaced points in three dimensions and assign an indication of wall allocation and cavity allocation for the idealised (fitted) and actual (true) geometries. Suitable constraints were placed on the values of the parameters since most parameters can be accurately estimated from the Cartesian representation of the left ventricle. These estimates also provide suitable initialisations for the parameter values.

3.4. Evaluation

Comparison of PCA with AEs is straightforward since it can be based on the average mean square error (MSE) between each test example in the dataset and the corresponding reconstructions. This is due to both PCA and AE predicting the original 17k mesh. We always perform a leave-1-out cross validation (1-out-CV), i.e. train on 197 examples and test on the remaining one example. Note that this means that for each CV fold there are slight differences in the benchmark mean mesh as well as in the standardisation parameters used to standardise the relevant training data (of $197 \times 17k$ values).

On the other hand, assessing the difference in performance between PCA and the parametric model is far from trivial due to the mesh predicted by the latter being of different dimension than the original one. Instead of 2896 nodes on the epicardium and endocardium, we now have 851 nodes on both surfaces. We use the criteria in eq. (1) , i.e. the same loss functions that we use to fit the parametric model, to compare the PCA results with these delivered by the parametric model.

4. Results

The parametric model differs from the PCA and DL approaches in two main aspects. First, as pointed out in Section 3.4, DL methods are directly comparable with PCA, which is not the case for the parametric model. For this reason we carry out comparisons with PCA separately for the DL methods and for the parametric model. Second, for PCA and the DL methods we can choose the dimension to which we want to reduce the original inputs. For PCA this is done by selecting the number of principal components for the final analysis, for the DL methods by setting the size of the bottleneck layer in AE. On the other hand, the parametric model requires a fixed number of 6 parameters to represent each mesh. Due to these differences, we first analyse the PCA results aiming to find its appropriate specification (number of components; single PCA vs MPPCA). Having selected the specification, we can directly compare it with AEs based on the same number of latent variables (parameters).

[1]Using `fminsearch` for local optimisation and `particleswarm` for global optimizations.

Finally, we can move to the parametric model, which we can compare with PCA using methods described in Section 3.4.

4.1. Principal Component Analysis

We start the analysis of PCA by investigating the number of components required for a reasonable performance of the method, subject to the constraint imposed by the emulator requiring a very low dimensional representation of the LVM, with 3–6 variables. Table 2 shows the variance explained using different numbers of PCA components. Given the constraint of using 3–6 components, 4 is a reasonable choice, since the 4th component explains more than the 5th, 6th, 7th components together. This conclusion is supported by the analysis of MSEs of the predicted meshes (in a 1-out-CV) for different numbers of PCA components, see Table 3. The marginal reduction in MSE is the highest for 2 components and is equal to 0.0069; however, 2 components do not explain enough variance (only 70%). The second largest marginal reduction in MSE of 0.0065 is for 4 components and for our purposes almost 86% of total variation explained by 4 components is satisfactory. Interesting, adding the 4 components decreases the MSE more than adding the 3rd component (by 0.0065 and 0.0056, respectively).Thus, we use the latent representation of 4 values in the further experiments, with the corresponding PCA error of $E_{\text{PCA}} = 0.0117$.

Table 2: Number of PCA components vs % of variance explained. Bold values: the chosen 4-component representation.

Number	1	2	3	**4**	5	6	7	8
Var. expl. (%)	60.5	9.9	7.9	**7.3**	3.0	2.0	1.5	1.3
Total var. expl. (%)	60.5	70.4	78.3	**85.6**	88.6	90.6	92.1	93.4

Table 3: Number of PCA components vs the predictive MSE error. Zero-component representation: the mean baseline. Bold values: the chosen 4-component representation.

Number	0	1	2	3	**4**	5	6	7	8
Error ($\times 10^4$)	769	307	238	182	**117**	94	80	68	58
Marginal error reduction ($\times 10^4$)	–	462	69	56	**65**	23	14	12	10

Having decided on the number of principal components, we are interested in visualising the effects of each of the four components on the shape of the LVM. Figure 2 illustrates that the first principal component controls the size of the mesh, while the remaining components are responsible for other shape properties such as endocardium width or asymmetry.

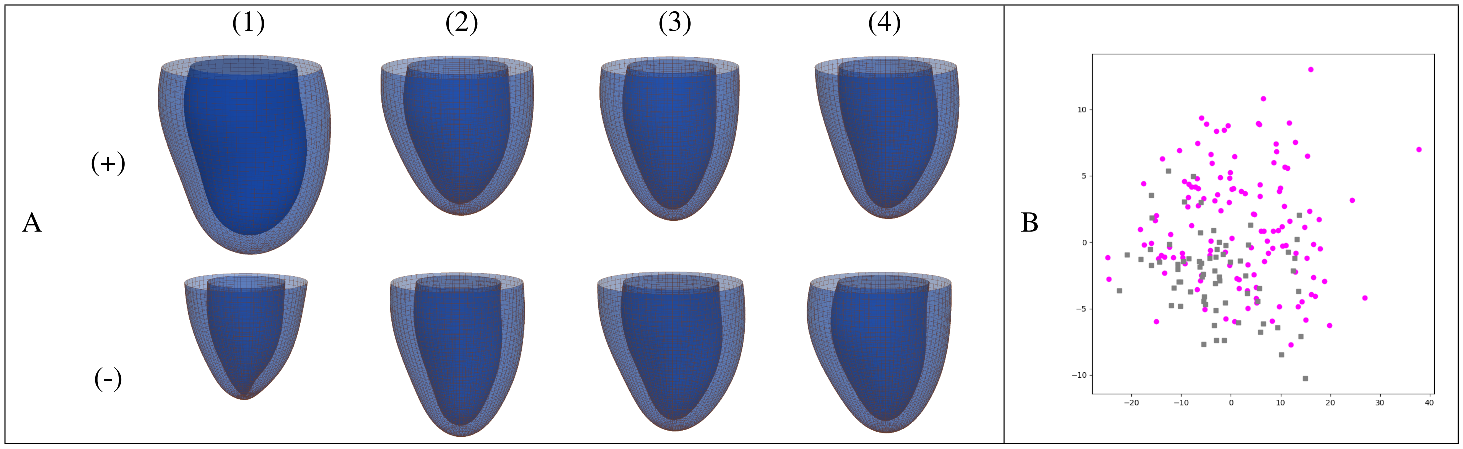


Fig. 2: A: template mesh deformed by varying the first four principal components. Columns – subsequent principal components (1) to (4); rows – setting each principal component to be 2 and −2 times the eigenvalue, for the top and bottom row, respectively. B: PCA projections into the space spanned by the 1st and 2nd components. Grey – healthy volunteers, magenta – MI patients.

Given that we have two types of subjects in the dataset, HV and MI patients, we next investigate whether there might be gains in the fit from applying an extension of the basic PCA, i.e. the MPPCA approach in [5]. If each subject class were much better explained by its "own" PCA mapping, then mixing both linear subspaces would result in an enhanced fit compared to applying single PCA to the joint dataset. Thus, we investigate the projections of the full dataset into the PCA space. As Figure 2, Panel B shows, in general both classes are well-mixed together and there is no clear boundary between both classes. The LVMs of MI patients and HVs differ mainly due to their coordinate values in the PCA space, with the MI patients being characterised by more extreme values. This suggests that using MPPCA might be suboptimal in our case, given an increased computational burden for the statistical emulator.

To quantitatively verify the findings from Figure 2, Panel B, we tested two separate PCAs, with 4 components each, fitted on each class separately. The resulting mean MSEs are 0.0108 and 0.0116 for MI patients and HVs, respectively. Given $E_{\text{PCA}} = 0.0117$ obtained for PCA fitted on the joint dataset, we confirm that there is only a slight improvement in the fit (and only for MI patients) from defining a separate mapping for each class. Since the cost of this potential subtle gain in accuracy is using two times more models, we decide to use single PCA.

Intuitively, these results are not surprising, as the examples in the dataset are fairly homogeneous due to being obtained by similar deformations from the same template mesh. Even though some deformations might be more frequent for MI patients, more extreme values of the shared PCA mapping are sufficient to capture these differences so that a different mapping might not be necessary.

4.2. Auto-encoders

We start the discussion of the results for AEs by reporting on the shallow neural networks. Next, we move to the deep architectures. We summarise our findings for AEs by a brief comparison with PCA.

4.2.1. Shallow AE

Figure 4 compares the MSEs from AE with two types of inputs with those for PCA. Interestingly, the shallow AE for the original 17k input converged exactly to the results from PCA, for all the examples, obtaining an average MSE of 0.0117, which is the same as for PCA. The results for the reduced input are almost identical (recall that the employed 60 first principal components capture almost 100% of the variability of the original input).

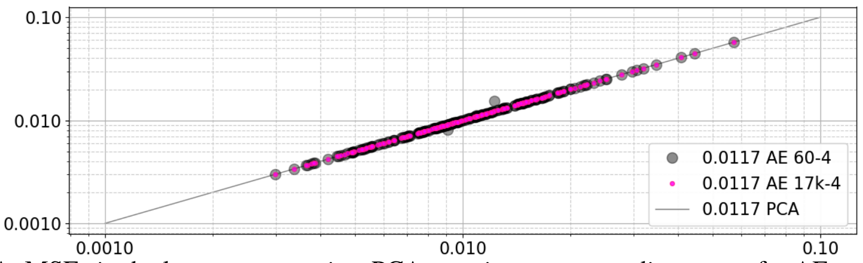
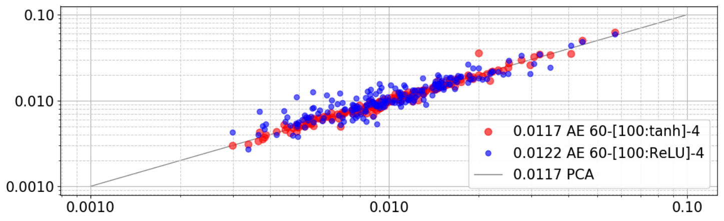


Fig. 3: Shallow AE vs PCA: MSEs in the log space. x-axis – PCA, y-axis – corresponding errors for AE, markers – each of the 198 test cases, black line – PCA identity. The legend additionally shows the average values of MSEs.

We note that the shallow AEs have the same computational complexity power as PCA since, similarly as PCA, they are just linear mappings from the input space to the 4 dimensional bottleneck space. What is interesting is that even without the orthogonality restriction inherent to PCA, both AEs converge to essentially the same mappings as PCA. Unlike PCA, however, training of AEs is relatively time-consuming. For the reduced input (AE 60-4), the training takes 12 minutes (approx. 4 seconds to train one network), while for the original input (AE 17k-4) it takes 6 hours (approx. 100 seconds to train one network).

4.2.2. Deep AE

The outcomes for deeper architectures are very similar to those from PCA and shallow AEs, yet more noisy. Figure 4 illustrates these points. The noise in predictions is magnified when the deep neural networks need to extrapolate, as then nonlinear units often lead to a poor fit. Moreover, the deep architectures are more prone to overfitting, especially for the original (17k) input, which requires around 100 times more weights. Another disadvantage of deep AEs in this context is their extended training time: for the reduced input the training took 30 minutes (20s per network), while for the original input it took 7 hours (130s per network).



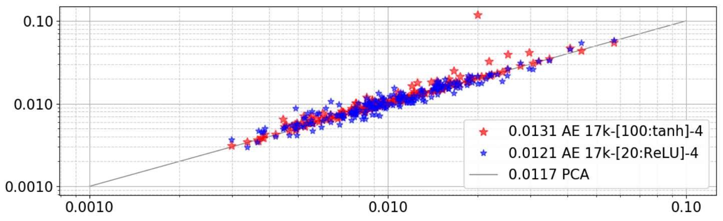


Fig. 4: Deep AE vs PCA: MSEs in the log space, left: reduced input; right: original input. x-axis – PCA, y-axis – corresponding errors for AE, markers – the 198 test cases, black line – PCA identity. Average MSEs in the legend.

4.2.3. Comparison with PCA

Table 4 compares the results for AEs with these for PCA. In general, the shallow AEs converged to the PCA transformation, delivering the same average MSEs. The deep AEs behave similarly and converged to mappings close to the PCA one, however their much more complex structure introduces more noise.

Table 4: MSEs for the statistical and machine learning methods: mean mesh (MM), PCA and auto-encoders (AEs) with the reduced (60) and original input (17k). AEs with different architectures: linear and with tanh and ReLU activation functions.

Method	MM	PCA	AEs					
			60-4	17k-4	60-100:tanh-4	17k-100:tanh-4	60-100:ReLU-4	17k-20:ReLU-4
Error	0.0769	0.0117	0.0117	0.0117	0.0117	0.0131	0.0122	0.0121
St. dev.	0.0709	0.0073	0.0073	0.0073	0.0073	0.0108	0.0073	0.0075

4.3. Parametric Model

Our main finding for the parametric method is that in general it performs poorly on our dataset. We attribute this to the symmetry constraint present in this method, which for numerous LVs, such as the one in Figure 5, Panel B, is a highly invalid assumption. This constraint is more suited to the setting of the original paper [4], where interest was in the end of diastole LV geometry (where the LV is fully inflated), which presents a more symmetric representation of the LV than early diastole (early inflation), which is of interest to us.

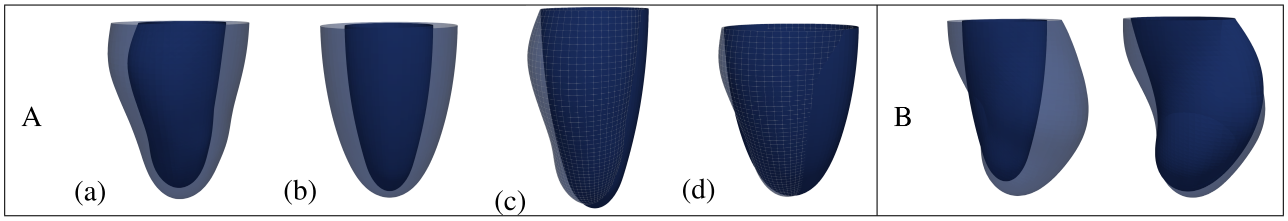


Fig. 5: A: parametric LV fit at the optimised parameters for a healthy left ventricle: (a) the true left ventricle, (b) geometry fitted using the parametric model, (c) true endocardium and fitted endocardium, (d) true epicardium and fitted epicardium. B: fitting results for the parametric model (left) and PCA (right): light blue – true mesh, dark blue – fitted mesh. The parametric method is unable to account for asymmetry in LV shape, PCA has no problem with this.

5. Conclusion

The main takeaway message from our study is that PCA provides the best performance in terms of the accuracy – computing time trade-off among three classes of methods for dimensionality reduction considered in this paper. DL methods are able to attain the same accuracy level as PCA in terms of mean MSEs, however, this comes at the price of long training

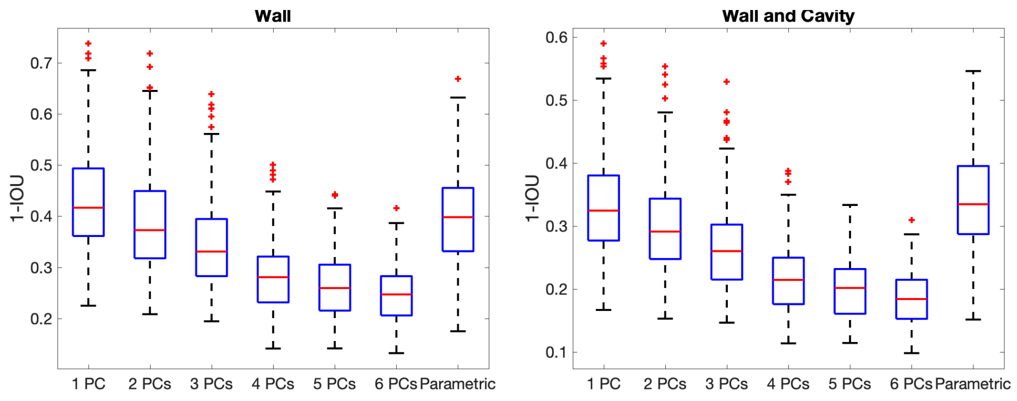


Fig. 6: Results for PCA with 1–6 first principal components (PCs) and the parametric model: the intersection-over-union (IOU) criteria from eq. 1. Left: E_W criterion (IOU for the wall only); right: E_{WC} criterion (IOU for the wall and cavity).

and specialist hardware. The fact that nonlinear DL methods cannot provide a more accurate reconstruction than PCA can be explained by the way each subject's LVM is constructed from the common template mesh. The parametric model comes with interpretable parameters but due to the symmetry constraint on the LVM provides a relatively poor fit to the real meshes, especially for MI patients. The dimensionality reduction work carried out in this paper paves the way for the construction of a statistical emulator to predict patient specific material properties of the left ventricle for which a proof of concept study [3] has already been carried out. The challenging aspect in scaling up this emulation was inclusion of the geometry space, for which PCA has been shown in this paper to provide an optimal low dimensional representation.

Acknowledgements

This work was funded by the UK Engineering and Physical Sciences Research Council (EPSRC), grant number EP/N014642/1. Alan Lazarus is partially funded by a grant from GlaxoSmithKline plc. Dirk Husmeier is supported by a grant from the Royal Society of Edinburgh, award number 62335.

References

[1] H. Gao, A. Aderhold, K. Mangion, X. Luo, D. Husmeier, and C. Berry, "Changes and classification in myocardial contractile function in the left ventricle following acute myocardial infarction," *Journal of The Royal Society Interface*, vol. 14, p. 20170203, July 2017.

[2] H. Gao, W. G. Li, L. Cai, C. Berry, and X. Y. Luo, "Parameter estimation in a Holzapfel–Ogden law for healthy myocardium," *Journal of engineering mathematics*, vol. 95, no. 1, pp. 231–248, 2015.

[3] U. Noè, A. Lazarus, H. Gao, V. Davies, B. Macdonald, K. Mangion, C. Berry, X. Luo, and D. Husmeier, "Gaussian process emulation to accelerate parameter estimation in a mechanical model of the left ventricle: a critical step towards clinical end-user relevance," *under revision for to Journal of the Royal Society Interface*, 2019.

[4] P. Di Achille, A. Harouni, S. Khamzin, O. Solovyova, J. J. Rice, and V. Gurev, "Gaussian process regressions for inverse problems and parameter searches in models of ventricular mechanics," *Frontiers in Physiology*, vol. 9, p. 1002, 2018.

[5] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural computation*, vol. 11, no. 2, pp. 443–482, 1999.

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

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.