Supplementary notes

Here we assess the performance of conventional linear dimensionality reduction in place of an autoencoder, and determine how varying the dimension of the input data and internal validation approach affects predictive performance. We also provide details of how the Kaplan-Meier plots were created and prediction models were compared.

Dimensionality reduction method

We designed a discriminative autoencoder network to learn non-linear representations of complex motion data that is robust to noise. Here we assess a conventional linear dimensionality reduction using a previously-published semi-supervised principal components method,[1, 2] – taking the deep learning-derived segmentations as input data. Internal validation was carried out using bootstrap-based optimism adjustment, with 100 bootstrap samples. Within each bootstrap sample, each of the input features was fitted to the survival outcome data using a univariate Cox proportional-hazards model, yielding 1 regression coefficient per feature. Salient features were selected based on whether their regression coefficient magnitude exceeded a certain threshold (defined via cross-validation). PCA was then applied to selected features and the first principal component was used as a continuous variable for survival prediction. The optimism-corrected C-index was 0.68 (95% CI: 0.62-0.74). Supervised dimensionality reduction through autoencoders has been shown to learn low-dimensional latent representations more effectively than PCA and also provides mappings in both directions between the data and code spaces.[3] An autoenoder with linear activations and a single hidden layer would approximate to the same subspace as PCA.

Down-sampling of the 3D mesh

Our segmentation and mesh generation pipeline produced high-resolution mesh models of the heart at 20 frames of the cardiac cycle, with each mesh composed of 18,028 vertices. Before using these meshes for our deep learning analysis, we down-sampled them (by a factor of ~99%) to 202 vertices using a mesh decimation algorithm. This was done to restrict the size of our DL network's input vector to a manageable range. For deep learning models, the size of the input data directly influences the number of network parameters to be estimated. In determining an appropriate input size, our goal was to strike a balance between a sufficiently rich input and an adequately parametrized model. Through visual inspection, the decimated meshes were compared to the original meshes, to ensure that the former still sufficiently captured the salient fine-scale geometric features of the latter.

We performed sensitivity analyses to assess the model performance for different numbers of vertices. We refitted our DL network on data consisting of meshes with 101 vertices, and meshes with 501 vertices. Internal validation was carried out via nested cross-validation. With 100 vertices, the Harrell's concordance index was 0.65 (95% CI: 0.57 - 0.7). For the 500-vertex case, the Harrell's concordance index was 0.68 (95% CI: 0.63 - 0.74). This suggests that our model with 202 vertices provides an effective degree of down-sampling.

Using coordinate data for prediction

In our study, heart motion over the cardiac cycle was represented using coordinate-wise displacement distances of vertex meshes from their positions at end diastole. A possible alternative would be to use a concatenated list of vertex meshes across all frames, as the displacement distances can be inferred from this information. However, the advantage of casting the input data as displacement vectors is that our DL network is presented with an explicit representation of the 3D dynamics of heart contraction, rather than having to infer it from lower level features (vertex positions). As deep learning methods tend to be data hungry, presenting our network with adequately refined input features helps to reduce the amount of extra information it needs to compute. And given our limited sample size, using cruder input features might reduce network performance.

To test this idea, we fitted our DL network using a concatenated list of vertex coordinates across all 20 frames, producing an input vector of length 12,120 (=3 coordinates/vertex * 202 vertices/mesh * 20 meshes [over 20 time frames]). The network setup for this input was identical to the one used for the original displacement-based input. Internal validation was carried out using nested cross-validation, with hyperparameter tuning carried out in inner folds, and validation performed in outer folds. The cross-validated Harrell's concordance index was 0.54 (95% CI: 0.47-0.61) indicating inferior performance to the explicit motion model.

Cross-validation vs Bootstrap internal validation

For internal validation of the models presented in this study, we chose the bootstrap-based approach *a priori* as it gives stable estimates of the performance drop (optimism correction) one would expect moving from internal to external validation.[4] Another common internal validation technique is cross-validation. To compare this method to our bootstrap-based approach, we performed internal validation of our DL model using nested cross-validation. Briefly, we randomly split the full data into 8 (roughly) equal parts (which we will refer to as outer folds). For each outer fold v, we carried out the following procedure: v was held out as a test set while its complement (the remaining sample not in v, which we will refer to as ~v) was used for hyperparameter tuning, via 8-fold cross-validation. The optimal hyperparameters derived from this 'inner' cross-validation were then used to train a final model on ~v. The trained model was then applied to fold v and model performance was assessed using Harrell's Concordance Index. This process was repeated for the rest of the outer folds, and concordance index across all outer folds was averaged to yield a final value. We repeated our original DL analysis using this nested cross-validation approach and the cross-validated Harrell's concordance index was 0.72 (95% CI: 0.667 - 0.779). Confidence intervals for the concordance index were derived using standard errors computed via Noether's method.[5, 6]

These results indicate that the cross-validation approach yields a slightly lower concordance index than bootstrap validation but with overlapping confidence intervals.

Generation of Kaplan-Meier plots and model comparison

Kaplan-Meier plots presented in the paper were generated using 'out-of-bag' predictions from the bootstrap-based internal validation procedure. During the internal validation procedure, each bootstrap sample was created by taking n random draws (with replacement) from the full

sample (where *n*=302, the number of unique subjects in our study data). According to bootstrap resampling theory, for large n, each bootstrap sample will contain (on average) only ~63.2% (=1 – e^{-1}) of the subjects from the full sample. This means that each bootstrap sample will almost always exclude a fraction of subjects from the full sample. In machine learning literature, this excluded subsample is sometimes referred to as the 'out-of-bag' subsample (and conversely, subjects included in the bootstrap sample are termed 'in-bag'). For a model trained on a particular bootstrap sample *b*, we can compute its predicted values for subjects in the out-of-bag subsample of *b*. After training a series of models over *b* = {1,...,*B*} bootstrap samples, we can, for each subject in the full sample, identify the bootstrap samples for which that subject was out-of-bag, and average the subject's predictions across these bootstrap samples. Thus for each subject, this will yield a predicted risk computed by aggregating predictions from models trained with data excluding that subject. These are referred to as out-of-bag predictions. This technique is reminiscent of 'bagging' (bootstrap aggregating), an ensemble method used in machine learning algorithms such as random forests. We generated Kaplan-Meier plots for the full sample using the out-of-bag predictions. Kaplan-Meier plot generation was implemented using the R packages *survival* and *survminer.*[7] Confidence intervals for the Kaplan-Meier plots were computed using the method of Link.[8]

To compare pairs of models fitted on our data, we computed for all subjects the out-of-bag predictions from each model. These out-of-bag predictions were derived from the bootstrap-based internal validation procedure, as described in the previous paragraph. Then we applied the *concordance.index* function in the R package *survcomp* using the survival time, censoring status and out-of-bag predictions (from the 2 models) across all subjects. This function utilizes a paired t-test to compare competing prognostic survival models applied to the same sample. P-values derived from this test were reported.

References

1. Dawes TJW, de Marvao A, Shi W, Fletcher T, Watson GMJ, Wharton J, . . . O'Regan DP. Machine learning of three-dimensional right ventricular motion enables outcome prediction in pulmonary hypertension: a cardiac MR imaging study. *Radiology*. 2017;283:381-390.
2. Bair E and Tibshirani R. Semi-supervised methods to predict patient survival from gene expression data. *PLoS Biol*. 2004;2:E108.
3. Hinton GE and Salakhutdinov RR. Reducing the dimensionality of data with neural networks. *Science*. 2006;313:504-7.
4. Steyerberg EW. *Clinical prediction models : a practical approach to development, validation, and updating*. New York, NY: Springer; 2009.
5. Noether GE. Elements of nonparametric statistics. 1967.
6. Pencina MJ and D'Agostino RB. Overall C as a measure of discrimination in survival analysis: model specific population value and confidence interval estimation. *Statistics in medicine*. 2004;23:2109-2123.
7. Kassambara A, Kosinski M, Biecek P and Fabian S. survminer: Survival Analysis and Visualization. 2017; URL: https://cran.r-project.org/web/packages/survminer/
8. Link CL. Confidence intervals for the survival function using Cox's proportional-hazard model with covariates. *Biometrics*. 1984:601-609.