2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Generating Cardiovascular Data to Improve Training of Assistive Heart Devices

1st Johannes Kummert
Machine Learning Group
Bielefeld University
Bielefeld, Germany
jkummert@techfak.uni-bielefeld.de

2nd Alexander Schulz
Machine Learning Group
Bielefeld University
Bielefeld, Germany
aschulz@techfak.uni-bielefeld.de

3rd Robert Feldhans
Machine Learning Group
Bielefeld University
Bielefeld, Germany
rfeldhans@techfak.uni-bielefeld.de

4th Moriz Habigt
Anaesthesiology Clinic
RWTH Aachen University
Faculty of Medicine
Aachen, Germany
mhabigt@ukaachen.de

5th Maike Stemmler
Institute of Automatic Control
RWTH Aachen University
Aachen, Germany
M.Stemmler@irt.rwth-aachen.de

6th Christina Kohler
Institute of Automatic Control
RWTH Aachen University
Aachen, Germany
C.Kohler@irt.rwth-aachen.de

7th Dirk Abel
Institute of Automatic Control
RWTH Aachen University
Aachen, Germany
d.abel@irt.rwth-aachen.de

8th Rolf Rossaint
Anaesthesiology Clinic
RWTH Aachen University
Faculty of Medicine
Aachen, Germany
rrossaint@ukaachen.de

9th Barbara Hammer
Machine Learning Group
Bielefeld University
Bielefeld, Germany
bhammer@techfak.uni-bielefeld.de

***Abstract*—In many medical applications data is a scarce resource and can often only be obtained with invasive surgery. This is for instance the case for physiological cardiovascular data that is necessary to improve the functionality of assistive heart devices. In this work we explore the viability of a GAN architecture to generate cardiovascular data towards enriching a data set obtained in animal testing on which training of future applications can be improved which potentially reduces the need for further animal testing. We evaluate the usefulness of our synthesized data using a downstream task.**

***Index Terms*—GAN, Data Generation, Data Enrichment, Medical Application**

I. Introduction

Due to the ongoing organ donor shortage for cardiac transplantation, ventricular assist devices (VADs) have evolved as therapeutical and/or supportive devices for patients with terminal heart failure [1]. While VADs are generally applied with a constant rotary speed in clinical practice [2], research for developing and evaluating effective control algorithms often requires invasive animal or human studies [3] and their application in clinical practice is still hindered by e.g. the necessity for invasive signal measurement. Hence, our goal is training a generative model that is capable of generating useful synthetic data based on a database of recorded cardiovascular signals from 25 animals. Therefore, reducing the need for further animal testing.

In the last years, large generative deep learning models have become ubiquitous in many domains, such as image and audio applications [4]–[6]. With their drastically increasing performance they aim to generate realistic new data samples, do reconstruction or segmentation, which is applied successfully e.g. in the medical domain [7]. Here, Generative Adversarial Networks (GANs) [8] are the particularly prominent model class and in the medical domain it is predominantly applied to images [7]. In the present work, in contrast, we focus on the cardiovascular system which is usually represented by signal data over time.

Time series prediction with deep generative models deals with data over time and utilizes typically one of three common architectures: recurrent networks, (variants of) convolutions over time or attention based models. We will focus on the first two classes of models in the following. Examples for the first architecture type include Autoregressive Denoising Diffusion models [9], where a recurrent network aggregates the output of a diffusion model in each time step and Recurrent Conditional GANs [10], where LSTM nodes are used for the generator and the discriminator models. An examples for convolution based architectures is the WaveNet model [6] that utilizes dilated causal convolutions and was successfully applied for raw audio generation. There also do exist combinations of both architectures: The ConvLSTM [11] and the Recurrent Convolutional

Funding by the German Federal Ministry for Education and Research (BMBF) under reference number 01|S21056C is gratefully acknowledged.

TABLE I
DATA SET OF HEART DATA

Animals	25
Total Interventions	320
Intervention 1	167
Intervention 2	5
Intervention 3	113
Intervention 4	3
Intervention 5	32
Average Length	3m9s
Total Length	16h47m51s
Frequency	1000Hz

GAN [12] are geared for spatiotemporal data, but are not applicable here because we don't consider spatiotemporal data. Recurrent Convolutional networks [13], [14] use recurrent connections per layer for static inputs, while we have time-dependent inputs. [14] is designed for text processing.

In our contribution, we evaluate a novel combination of training convolution based GAN networks in a recurrent fashion, that is particularly suited for our setting. We compare our approach with two architectures from the literature and evaluate the quality of our generated samples by investigating in how far the performance on a downstream task can be improved by adding synthetic data generated with our model.

II. FOUNDATIONS

In this section, we introduce the data set and core learning approaches that we build upon in the present work.

A. Heart Data

The data set used for this work was obtained by animal tests carried out at the University Hospital RWTH Aachen in Germany where healthy pigs received VAD implantations [15], [16]. A variety of physiological indicators of the heart were measured while inducing different levels and kinds of variations on the cardiovascular system including preload, afterload and contractility changes. We refer to the latter as different interventions. The different stages of an intervention (setup, ramp up, execution, ramp down, closing) are marked as phases. We use only data from the setup and the execution phase for training. The setup phase for giving a baseline of a resting heart state and the execution phase for giving clear examples of possible problems with the heart. In the following, we utilize the data subset consisting of the pressure and volume in the left ventricle, since these constitute promising candidates for VAD control algorithms [15], [16]. Both of these signals are also only available through invasive sensors, therefore, synthetic generation is particularly interesting to potentially avoid a more invasive procedure. Relevant characteristics on the data set can be found in Table (I).

B. Generative Adversarial Networks

Introduced by Goodfellow et al in [8] GANs consist of two competing neural networks, the generator G and discriminator D. The goal of the generator is to create data that resembles a given set. The role of the discriminator is to distinguish between created and original data.

C. Pix2Pix

Isola and colleagues describe in [17] a conditional GAN used for image to image translation where the generator creates an image given an input image as the condition. The training process was adapted so that the discriminator has to decide whether an image was created based on a given input image. During one training step, this is done for the created image and the image from the original data set.

III. METHODS

In this section, we describe how we adapt and apply the previously described material and approaches.

A. Data Format

We sub-sample the data from 1000Hz down to 50Hz since this is the format one possible future application, the training of a VAD controller, uses. We focus on generating two cardiovascular signals which we also use as input for the model: the volume and pressure in the left heart chamber, for their physiological significance. But our method can be easily expanded to generate more signals.

Since we utilize convolutional networks, we need to specify the length of input time windows for intermediate predictions. To specify a reasonable length, we make use of domain knowledge: different breathing states affect the signals at hand, hence the model should get a few heart beats as input in order to be able to (implicitly) detect the current one. Accordingly, we decide to employ input windows of length $n = 100$ (two seconds) such that they include at least a couple of heartbeats. Our goal is to predict longer sequences, where we consider exemplarily a prediction length of $o = 1000$.

B. Training Goal and Augmentation

We define $\hat{s}_{i,\dots,j}$ as a training sequence from our data set from time step i to time step j. The goal of our model is to use a starting window $\hat{s}_{1,\dots,n}$ of length n to generate a sequence $s_{1,\dots,o}$ of length o that closely resembles the training sequence. This means the first n steps of each sequence are the same:

$$\hat{s}_{1,\dots,n} = s_{1,\dots,n} \tag{1}$$

For evaluation of a model we use the L1 error ($|s_{n,\dots,o} - \hat{s}_{n,\dots,o}|$) between the sequences from time step n to o.

Data augmentation is a common and often necessary approach for deep networks in order to make the training more stable and to increase the performance. Typical data augmentation strategies from the image domain, such as e.g. flipping and mirroring cannot be applied sensibly in our domain. However, we make use of the following augmentation strategy: Due to absolute time invariance of the data we can shift the input signals. More precisely, for each input sequence of length $n = 100$ (where the target is long enough) we generate n input sequences by shifting the input sequence for one of $\{1, \dots, n\}$ time steps. The according target has to be shifted correspondingly.

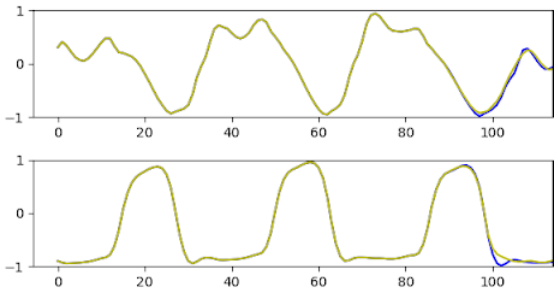

Fig. 1. Example of difficulty to differentiate between original (blue) and generated (yellow) sequences on short prediction windows. Volume signal on top, pressure below.

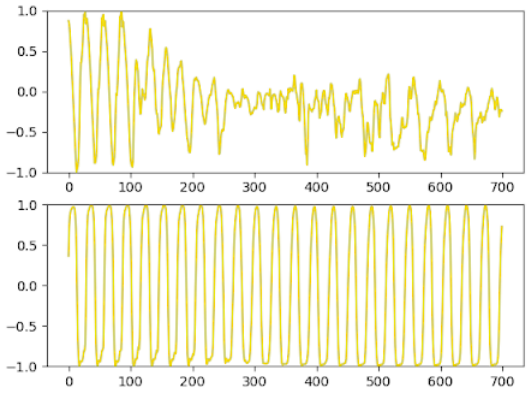

Fig. 2. Example of a sequence deteriorating after repeated applications. Volume signal on top, pressure below.

C. GAN Architecture

We employ the pix2pix learning setup as the model architecture (see (II-C)) where instead of image-to-image translations we aim to translate the start of a sequence to a prediction of the next m steps. Therefore, the generator G of our model calculates

$$G(\hat{s}_{1,\dots,n}) = s_{n+1,\dots,m} \tag{2}$$

The discriminator D is evaluated for the following two inputs: the input sequence of the generator $\hat{s}_{1,\dots,n}$ is concatenated with the prediction to judge whether its a realistic sample:

$$D(\hat{s}_{1,\dots,n}, s_{n+1,\dots,m}) = E_g. \tag{3}$$

Here, D is supposed to learn to output *fake input*. Also, the actual continuance of the sequence from the training data is used as input, as is usually done for GAN training

$$D(\hat{s}_{1,\dots,n}, \hat{s}_{n+1,\dots,m}) = E_r \tag{4}$$

where E_r is supposed to be mapped to *real input*. E_g is used in conjunction with

$$E_{L1} = |s_{n+1,\dots,m} - \hat{s}_{n+1,\dots,m}|, \tag{5}$$

the L1 error between the generated and real sequence, as the error signal for the generator G. Both E_g and E_r are used as the error signal for the discriminator D. The advantage of this method is that the output sequence can be used as input (or as a part of it) to create, in theory, infinitely long new sequences by repeatedly applying the generator using

$$G_i(s_{im+1,\dots,n+im}) = s_{n+im+1,\dots,n+(i+1)m} \tag{6}$$

The generator uses a U-Net architecture as proposed by [17] utilizing skip connections between four encoding and decoding layers. Of the four encoding layers, the first is a convolutional layer over time and space and the next three only over time. The latent space created by the encoding is of size 64 and the intervention and phase (see (II-A)) of the sample is given as additional input and attached to the middle layer before decoding. This is done to potentially create samples that go through different phases of one intervention or even through multiple different interventions. The discriminator uses a patch GAN architecture with four convolutional layers. Rectified Linear Units (ReLU) [18] are used as the activation function and Adam [19] as optimizer. For the decoding layers of the generator a dropout of 0.3 is applied.

D. Proposed Discriminator Adaptation

Since this architecture with a standard discriminator is not capable of generating long realistic sequences by iterative application of the generator, we suggest to adapt the discriminator as follows. The simple but effective core idea is to extend the input of the discriminator from the single prediction of the generator to its iterative application as denoted in equation (6). More specifically, we apply the generator i times such that later parts of the prediction can be evaluated by the discriminator, therefore changing the error signals to

$$D(\hat{s}_{1,\dots,n}, s_{n+1,\dots,n+im}) = E_g \tag{7}$$

$$D(\hat{s}_{1,\dots,n}, \hat{s}_{n+1,\dots,n+im}) = E_r \tag{8}$$

IV. Experiments

In our experiments, we first analyze the performance of a classic GAN approach with iterative application of the generator and demonstrate that the performance of the discriminator is a problem. Consequently, we evaluate our proposed training scheme with various hyperparameter settings and show a strong improvement. Subsequently, we investigate in how far the generated samples can help improve a downstream task by augmenting underrepresented classes and compare this approach to a baseline for data augmentation. Finally, we conduct a qualitative sanity check of our generative model with the use of dimensionality reduction. Training of the GAN model was done using Tensorflow 2.0 [20].

A. Initial Experiments

In our initial experiments we observe that the model becomes capable predicting the sequence very accurately for a single application of the discriminator: In Fig. (1) it can be seen that for short prediction windows it is barely possible

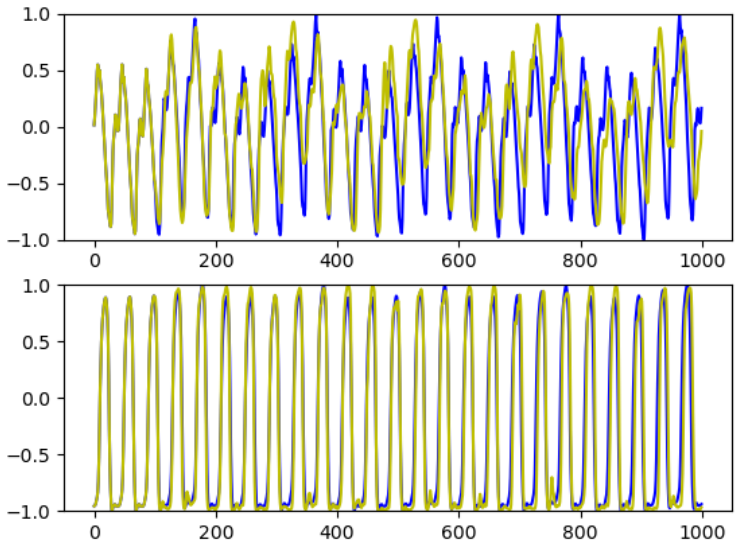

Fig. 3. Example of fully generated sequence (yellow) compared to the original (blue).

to distinguish between the real (in blue) and the generated (in yellow) sample. This is also evident by a very low classification accuracy of 59.31% for the discriminator even after further stand alone training of the discriminator. Additionally, with repeated application of the generator, the signal often deteriorates which is shown exemplarily in Fig. (2).

Therefore, we adapt the training of the GAN, more precisely the discriminator, as previously described (see III-D).

In our experiments, we first analyze the performance of a classic GAN approach with iterative application of the generator and demonstrate that the performance of the discriminator is a problem. Consequently, we evaluate our proposed training scheme with various hyperparameter settings and show a strong improvement. Subsequently, we investigate in how far the generated samples can help improve a downstream task by augmenting underrepresented classes and compare this approach to a baseline for data augmentation. Finally, we conduct a qualitative sanity check of our generative model with the use of dimensionality reduction.

B. Initial Experiments

In our initial experiments we observe that the model becomes capable predicting the sequence very accurately for a single application of the discriminator: In Fig. (1) it can be seen that for short prediction windows it is barely possible

TABLE II
BASELINE TEST RESULTS ($o = 200$)

Model	**L1 Error**
GAN adapted	51.95
LSTM	138.3

to distinguish between the real (in blue) and the generated (in yellow) sample. This is also evident by a very low classification accuracy of 59.31% for the discriminator even after further stand alone training of the discriminator. Additionally, with repeated application of the generator, the signal often deteriorates which is shown exemplarily in Fig. (2).

Therefore, we adapt the training of the GAN, more precisely the discriminator, as previously described (see (III-D)).

C. Discriminator Adaptation Results

Firstly, we evaluate the L1 error on a smaller prediction window (of length 200) of our model against a LSTM model as a baseline comparison to estimate viability of our method. Table (II) shows that our model achieves a significantly lower reconstruction error on the signal compared to the baseline. Therefore, we decided to optimize and evaluate our model for longer prediction lengths.

This evaluation of our model is done for different step lengths m and different reccurence levels i. The results of our experiment with this adaptation can be seen in Table (III).

TABLE III
DISCRIMINATOR ADAPTATION RESULTS: $o = 200$ TOP, AND $o = 1000$ BOTTOM.

Step Length m	**Recurrence Level** i	**L1 Error** GAN	**L1 Error** No E_g
10	9	672.18	841.91
25	9	287.57	464.39
50	9	300.29	470.31
100	9	304.11	479.66
10	20	**270.22**	443.71
25	20	280.58	467.11

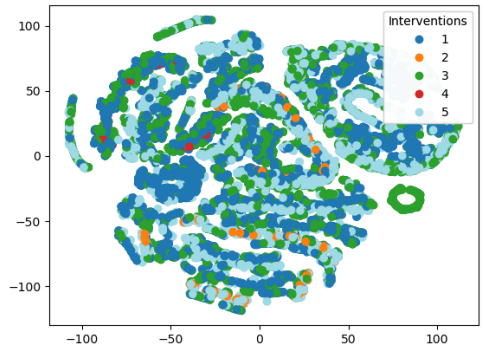

Fig. 4. Visualization of the latent space of one of the initial models ($m = 25$) colored by Intervention.

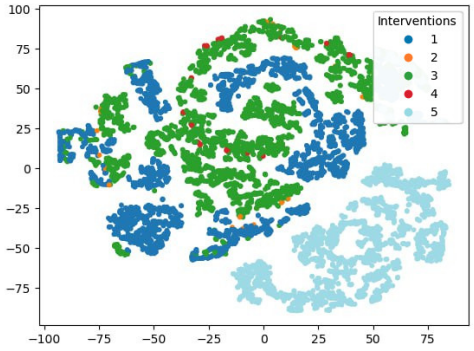

Fig. 5. Visualization of the latent space of one of a model with discriminator adaptation ($m = 25,\ i = 20$) colored by Intervention.

we can see that results generally improve for shorter prediction windows unless the recurrence level is not deep enough. The best result was achieved with short prediction length and deep recurrence. The last column shows the error when the model was trained without the generator error where we see a significant drop in performance. Therefore, we conclude that the GAN part helps to train the model. One example of a fully generated sequence can be seen in Fig. (3).

D. Validation via Class Balancing

As mentioned before our goal is not just to create a one-to-one reconstruction of our original data set but to meaningfully enrich it with generated data to improve possible further learning tasks. As an example of such a task we train a random forest classifier (RFC) on the original data set to classify which intervention the sequence belongs to. We use the implementation provided in scikit-learn [21]. The class wise accuracies of the RFC trained on only original data can be seen in Tab. (IV).

We can see that accuracies for classes two and four are very poor. This is due to them being underrepresented in the data compared to the bigger classes one, three, and five (see Table (I) for the details). We apply SMOTE [22] as a common method for balancing data by up sampling the two minority classes and down sampling the bigger classes. We compare those results to a classifier trained on data that was augmented to the same ratios as SMOTE by our GAN approach by predicting windows of the smaller classes into the future and using those as part of the class for training. Tab. (IV) shows the accuracy of the classifiers trained on the different data sets and evaluated on a test set of only original data. We see a significant improvement with data enhanced by our method for the imbalanced classes two and four while maintaining similar accuracies for the other classes and overall classification accuracy. Our method also outperforms data balanced by SMOTE across the board, with the Wilcoxon rank-sum test showing a significance $p < 0.01$.

TABLE IV
RFC RESULTS TRAINED ON THE ORIGINAL DATA SET, DATA AUGMENTED WITH OUR GAN METHOD, AND DATA AUGMENTED BY SMOTE.

Class	Original Data	GAN	SMOTE
1	89.84 ± 1.14	89.53 ± 1.14	85.84 ± 1.38
2	0.67 ± 2.11	86.95 ± 7.96	79.76 ± 1.35
3	93.34 ± 0.98	91.92 ± 1.01	89.65 ± 1.78
4	32.84 ± 8.85	100.0 ± 0	95.66 ± 1.83
5	90.14 ± 0.76	90.19 ± 1.21	84.17 ± 1.32
Overall	89.72 ± 0.84	**90.67 ± 0.47**	85.69 ± 0.73
Average	61.37 ± 1.98	**91.72 ± 1.5**	87.02 ± 2.41

E. Latent Space Visualization

If we consider the activations in the bottleneck layer of our generator as a latent representation and visualize it using dimensionality reduction via tSNE [23] we can further see the improvements of the proposed discriminator adaption in regards to the intervention classification. Visualizing the latent space of the model with the discriminator adaptation we see a clearer structure and separation between the different interventions.

V. CONCLUSION

In this work we introduce a method to utilize a neural network using a GAN architecture to predict sequence data. We show that the generated data can be used to enrich the original data to improve the accuracy of classification problems. We think this shows viability of this method to improve further learning tasks specifically in the medical applications of training controller for VAD devices. Investigation of the model's functionality, i.e. by plotting classification boundaries [24], can be done to further optimize and improve performance.

REFERENCES

[1] D. Mancini and P. C. Colombo, "Left ventricular assist devices: A rapidly evolving alternative to transplant," *Journal of the American College of Cardiology*, vol. 65, no. 23, pp. 2542–2555, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0735109715020665

[2] J. K. Kirklin, D. C. Naftel, F. D. Pagani, R. L. Kormos, L. W. Stevenson, E. D. Blume, S. L. Myers, M. A. Miller, J. T. Baldwin, and J. B. Young, "Seventh intermacs annual report: 15,000 patients and counting," *The Journal of Heart and Lung Transplantation*, vol. 34, no. 12, pp. 1495–1504, 2015, special Issue: Mechanical Circulatory Support. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053249815014503

[3] M. A. Habigt, J. Gesenhues, M. Ketelhut, M. Hein, P. Duschner, R. Rossaint, and M. Mechelinck, "In vivo evaluation of two adaptive starling-like control algorithms for left ventricular assist devices," *Biomedical Engineering / Biomedizinische Technik*, vol. 66, no. 3, pp. 257–266, 2021. [Online]. Available: https://doi.org/10.1515/bmt-2020-0248

[4] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[5] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Mller, J. Penna, and R. Rombach, "Sdxl: Improving latent diffusion models for high-resolution image synthesis," 2023.

[6] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[7] X. Yi, E. Walia, and P. Babyn, "Generative adversarial network in medical imaging: A review," *Medical image analysis*, vol. 58, p. 101552, 2019.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[9] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf, "Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8857–8868.

[10] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," *arXiv preprint arXiv:1706.02633*, 2017.

[11] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[12] K. Deng, T. Fei, X. Huang, and Y. Peng, "Irc-gan: Introspective recurrent convolutional gan for text-to-video generation." in *IJCAI*, 2019, pp. 2216–2222.

[13] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3367–3375.

[14] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 29, no. 1, 2015.

[15] M. A. Habigt, J. Gesenhues, M. Ketelhut, M. Hein, P. Duschner, R. Rossaint, and M. Mechelinck, "In vivo evaluation of two adaptive starling-like control algorithms for left ventricular assist devices," *Biomedical Engineering/Biomedizinische Technik*, vol. 66, no. 3, pp. 257–266, 2021.

[16] M. A. Habigt, M. Hein, J. Gesenhues, D. Abel, R. Rossaint, and M. Mechelinck, "In vivo evaluation of a novel control algorithm for left ventricular assist devices based upon ventricular stroke work," *ASAIO Journal*, vol. 69, no. 1, pp. 86–95, 2023.

[17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[18] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

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

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[23] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[24] A. Schulz, F. Hinder, and B. Hammer, "Deepview: Visualizing classification boundaries of deep neural networks as scatter plots using discriminative dimensionality reduction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 2305–2311, main track. [Online]. Available: https://doi.org/10.24963/ijcai.2020/319