Simulating hiPSC Cardiomyocyte Electromechanics with Machine Learning Surrogate Model

Olli Ylinen[1], Antti Ahola[1], Ossi Noita[1], Jussi T. Koivumäki[1], Jari Hyttinen[1]

[1] Tampere University, Tampere, Finland

Abstract

Differential equation-based in silico models are a mainstay in cardiac modelling. However, due to their complexity, they are solved numerically, which is computationally expensive especially when large numbers of model runs are required. This can happen with human induced pluripotent stem cell derived cardiomyocyte (hiPSC-CM) models in tasks like population of models approach or optimization of model parameters. Machine learning-based surrogate models can estimate numerical solutions at a lower computational cost.

The aim was to make a machine learning based surrogate model to quickly and accurately replicate the electrophysiological steady-state outcomes of an established hiPSC-CM ODE-based model. We generated a dataset by defining 22 in-silico model parameter multipliers as inputs to produce 5 output traces: membrane potential, calcium transient, active tension, L-type calcium current and delayed rectifier potassium current. A convolutional neural network based surrogate model was trained on the data to predict the steady state response and demonstrated the use with simulated drug tests.

Compared to the ODE-based model, the surrogate model achieved 1 000x faster runtime with minimal output error and similar output data breadth. The model is publicly available.

1. Introduction

In-silico models of human cardiac cells are complex systems, based on measurements of electrical currents and conductances of different ion channels in the cell. Traditionally, ordinary differential equations (ODEs) are fitted to these measurements, which together can form complex models of the cell with often no exact solutions. Therefore the ODE-based models have to be solved numerically, can require tens of seconds of computing time for a single model run. Simulating disease condition, drug impacts or patient-specific cases can be done via adjustment of model parameters. This task of optimization requires the models to be run multiple times, which can become computationally expensive despite parallelization, especially with a population-of-models-approach.

One method of reducing computational burden is surrogate modelling that estimates the solution based on input data [1,2]. The goal of surrogate modelling is to implement an alternative way to estimate the original problem, accepting small errors in results for drastically lower computational costs. Here, we present a surrogate (SUR) model and software tool that replicates the hiPSC-CM in-silico model hiMCES [3], estimate its accuracy and demonstrate its use in drug effect modelling.

2. Methods

The overview of the surrogate model training pipeline and its components is illustrated in Figure 1.

2.1. Data pipeline

Using the Matlab implementation of the hiMCES model (Figure 1B), we generated 65 000 input and output pairs. Inputs consist of 22 electrophysiological model parameter multipliers consisting of conductances, ratios and time constants [3, 5], and outputs of action potential (AP), calcium transient (CaT), active tension (AT), L-type calcium current (I_{CaL}) and delayed rectifier potassium current (I_{kr}) traces. The model was paced at 2/3 Hz to elicit an AP, and run for 802.4s to reach steady state [2] and capturing a 1.5 s output trace. The specified model parameters varied using Latin hypercube sampling between 0.5 to 2 and generated with population of models approach using POMtool [4].

The generated data was further filtered to remove undesired behavior based on three criteria. First, the stimulation was modified to activate only below -52 mV membrane voltage to prevent mid-AP stimulation. Second, to ensure the output is starting near resting state, the first sample should start below -52 mV. Finally, the stimulated peak of membrane voltage occurring in the first 300 ms should be at least 0.9 as large as the largest peak in the trace.

The Matlab implementation of the hiMCES model uses an adaptive ODE solver, resulting in time differences be-

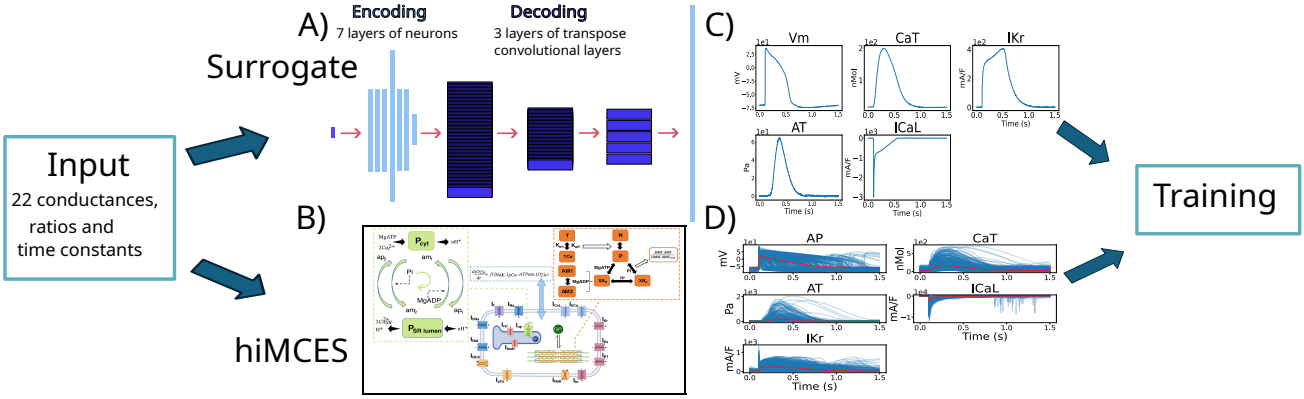

Figure 1. Overview of the training pipeline, inputs, A) SUR model architecture, B) hiMCES model visualization [3], C) SUR model output and D) hiMCES model output and a part of the generated training dataset.

tween neighboring points and the number of points for each sample being different [3]. To ensure consistent input and output sizes for the neural network, each output trace was linearly interpolated to 2000 samples the 1.5 s. Figure 1D shows 500 output traces from the filtered training data with red line as mean.

The data was normalized to counteract the different scales and amplitudes of the hiMCES model output signals, making the training of the model easier with more scale-invariant, normalized outputs. The normalization was done using the mean and variance of each output signal in the entire training set with z-score normalization. While this results in the output of the surrogate model being normalized, it is reversed using the mean and variance of the training set (Figure 1C).

2.2. Surrogate model architecture

The SUR model network consists of encoder and decoder parts, where the encoder converts the parameter multipliers to a learned feature space and the decoder converts those features to continuous signal representations (Figure 1A). The inputs are parameter multipliers and outputs are time-series data parallel to each other. As the inputs are independent of each other, the encoder uses a regular neural network of neurons. However, the outputs are dependent of each other in time and between dimensions, so convolution can utilize the relations in the decoder architecture. Due to the abundance of training data, the main idea is to learn an ideal feature representation for the task.

Before the encoding block, the inputs are normalized to a [0,1] range. The encoding block consists of 7 regular neural network layers, with tanh activation functions between them. The first 3 layers have 500 neurons each, followed by a layer with 1000 neurons, two layers with 500 neurons each, and the final layer with 250 neurons, producing the custom feature vector. The feature vector is passed on to the decoding block, consisting of 3 transpose convolutional layers, containing 256, 128 and 5 kernels (size 4, stride 2), respectively, with the final 5 kernels corresponding to the output channels. Between each transpose convolutional layer, a tanh activation function is applied. Finally, a linear output layer of 2000 neurons is applied to help in scaling the values beyond amplitude of 1.

The model has 6.03 million parameters, using 22.99 MB of memory with computational complexity of 0.32 Gflops.

2.3. Training

The generated data was split in 10:1:2 ratio for training, validation and testing. The training was done on Tampere Center for Scientific Computing (TCSC) cluster with a Tesla V100 16 GB GPU. The training of the final model was done with 1 000 epochs that took 7.7 hours to run, with a learning rate of 0.0001 and batch size of 100. The training was done with absolute loss function (L1) that compared every point for every output with the target output and averaged them into one number of absolute error per sample.

2.4. Model evaluation

The SUR model data generation, training and runtime time were compared against the Matlab-based hiMCES model runtime to evaluate computational efficiency for eventually determining use cases. The calculation was done on the TCSC with 32 GB of memory and 1 core of AMD EPYC 7343 16-Core processor, without GPU.

The accuracy of the SUR model was quantified by calculating the mean absolute error (MAE) between the SUR model and hiMCES model output traces, using the test dataset as input. To evaluate the output breadth and to have a comparison point for error values, we also calculate for both models the MAE between the output of the test set

and the model baseline (BL) state, where the input is the model default parameter set (Figure 2).

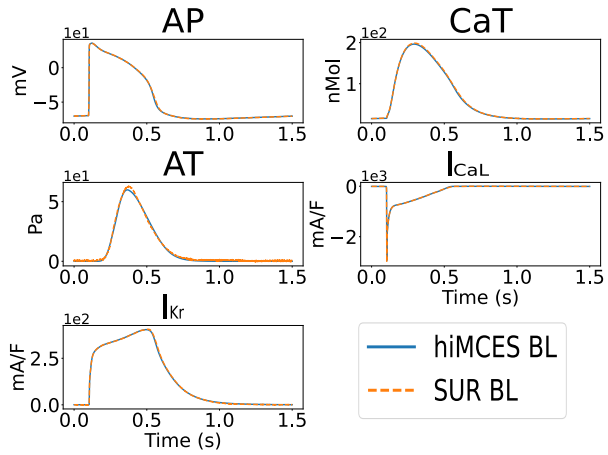

Figure 2. The outputs defined in Section 2.1 for hiMCES BL and Surrogate BL, indicating their default states.

We validated the model performance by replicating known drug effects [6] with both models. We changed model input parameters to match Astemizole 0.001 µMol, Bepridil 0.1 µMol, Diltiazem 0.1 µMol and Dofetilide 0.001 µMol, and calculated biomarkers and MAE from the output traces of both models. The biomarkers were AP duration at 90% repolarization (APD_{90}), CaT duration at 90% decay ($CaTD_{90}$), time from peak contraction to 80% of relaxation (RT_{80}), I_{CaL} minimum ($I_{CaL(min)}$) and I_{Kr} maximum ($I_{Kr(max)}$).

3. Results

3.1. Runtime comparison

We compared the runtime of the SUR and hiMCES models for producing the steady state output. Taking the training time and data generation time into consideration for the SUR model, the break-even point for runtime was 65 561 runs (Table 1).

Table 1. Runtime comparison

Model	Runtime per output (s)	Training (s)	65 000 data generation (s)	65 561 runs (s)
SUR model	0.005	27 720	3 250 000	3 278 047
hiMCES (Matlab)	50	0	0	3 278 050

3.2. Accuracy and output variability

The SUR model output accuracy and breadth was compared with the hiMCES model outputs. Mean absolute error (MAE) between the outputs of the models was calculated, along with both model outputs against their respective BL. The results are shown in Table 2.

Table 2. Mean absolute error (MAE) between models, and between model outputs defined in Section 2.1 and its respective baseline.

MAE	SUR vs hiMCES	hiMCES vs hiMCES BL	SUR vs SUR BL
AP (mV)	1.16	18.8	18.6
CaT (nMol)	3.85	50.4	50.0
AT (Pa)	5.83	44.0	42.6
I_{CaL} (mA/F)	1.49	115	113
I_{Kr} (mA/F)	5.43	92.7	92.4

3.3. Drug validation

In addition to testing the SUR model with the generated datasets, we predicted the effect of selected drugs on the model outputs defined in Section 2.1 based on literature data. The comparison of the SUR model and hiMCES model outputs is shown in Table 3.

Table 3. Biomarkers defined in Section 2.4 and mean absolute error (MAE) are calculated from the outputs of the SUR and hiMCES models that are defined in Section 2.1.

Drug	Astemizole	Bepridil	Diltiazem	Dofetilide
Dose (μmol)	0.001	0.1	0.1	0.001
APD_{90} (ms)	597/600	653/659	470/477	771/783
$CaTD_{90}$ (ms)	693/695	742/745	590/598	837/842
RT_{80} (ms)	283/301	291/310	280/300	385/416
$I_{CaL(min)}$ (mA/F)	-3040/ -2880	-3100/ -2620	-2770/ -2870	-3020/ -2920
$I_{Kr(max)}$ (mA/F)	302/306	238/240	403/405	202/201
MAE				
AP (mV)	0.281	0.317	0.358	0.533
CaT (nMol)	1.16	0.606	0.931	0.691
AT (Pa)	0.647	0.605	0.635	1.17
I_{CaL}(mA/F)	3.65	2.93	2.19	3.69
I_{Kr} (mA/F)	0.912	1.15	1.77	1.52

The impact of Bepridil application with 0.1 µMol concentration on the selected outputs as predicted by the SUR model and hiMCES model, compared to the hiMCES BL is shown in Figure 3.

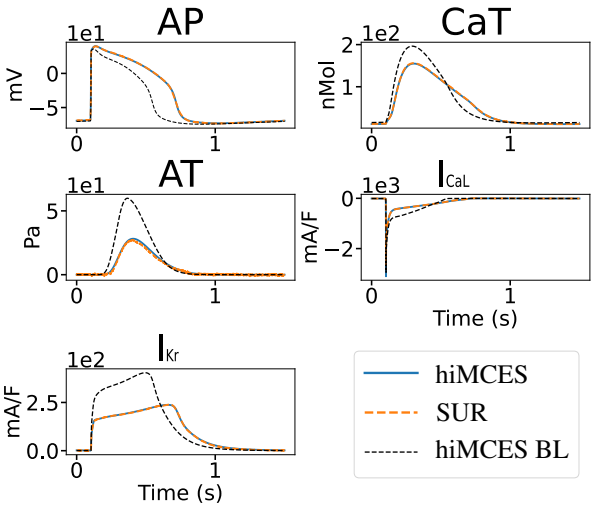

Figure 3. Impact of Bepridil application on the outputs defined in Section 2.1 as predicted by the hiMCES, SUR and hiMCES BL model.

3.4. Model availability

The software, model and trained weights are publicly available for use in Github [5] (CC BY-NC 4.0).

4. Discussion & Conclusions

This work presents a surrogate (SUR) model for estimating the five steady state key time series of the hiPSC-CM model hiMCES. The new model reduces the computational runtime burden 1 000-fold on a CPU. The SUR model runtime is easily parallelized and can be further improved using a GPU. Based on visual inspection, the qualitative error of SUR model predictions is small. Quantitatively, the errors are minor especially regarding membrane voltage and calcium transient.

Computationally, the SUR model excels at tasks where large numbers of model runs are required due to its low runtime. On the other hand, the prerequisite of dataset generation and training induces substantial computational cost and requiring 65 561 model runs to break-even in the runtime comparison. The model tends to have most difficulties in predicting AT and fast changes in magnitude seen I_{CaL} peak. This usually results in noise-like behavior and under- or over-predictions. The presented model is also limited compared to the original, because it only tries to estimate a part of the original model capabilities. Furthermore, the obvious distinction is that the SUR model tries to estimate the differential equations primarily on the ranges of the learning data inputs and it disregards the knowledge of biophysical mechanisms and physiological function captured in the system of ODE equations.

An interesting future use case for the SUR model could be the opposite of what the model does: the inverse problem that uses the output traces to estimate the electrophysiological changes that caused them. The SUR model could be used to try and solve this problem with iteratively changing input parameters to achieve a target output. This could provide very quickly and efficiently some in-depth information what is causing the observed changes in cellular phenotype, supporting treatment planning. Despite this, several problems other than runtime remain in the inverse problem solving, such as multiple different parameter combinations for same states.

Future work could have many possible directions. The SUR model could likely be trained with significantly less data, or the architecture could be more optimized for accuracy or speed. The SUR model publicly available weights could be fine-tuned on a smaller dataset for a different cell model, inputs or outputs, which could results in greatly reduced break-even point in performance. A surrogate model could be made to represent multiple cells and their interactions at the same time instead of a single cell.

Acknowledgments

The study is part of SMASH-HCM project, and receives funding from the European Commissions Horizon Europe Programme under grant agreement number 101137115.

References

[1] Alizadeh et al., "Managing computational complexity using surrogate models: a critical review," *Research in Engineering Design*, 2020.

[2] Grandits et al., "Neural network emulation of the human ventricular cardiomyocyte action potential for more efficient computations in pharmacological studies," *eLife*, 2024.

[3] Forouzandehmehr et al., "In silico study of the mechanisms of hypoxia and contractile dysfunction during ischemia and reperfusion of hiPSC cardiomyocytes," *Disease Models & Mechanisms*, 2024.

[4] Noita et al., "POMtool: Software tool for efficient population of models creation," *Computing in Cardiology*, 2026. Available: `https://github.com/TAU-CBIG/POMtool`

[5] Ylinen et al., "hiPSC-CM Surrogate model," *GitHub repository*, 2026. Available: `https://github.com/oYlinen/hiPSC-CM-Surrogate-model`

[6] Paci et al., "All Optical Electrophysiology Refines Populations of In Silico Human iPSC-CMs for Drug Evaluation," *Biophysical Journal*, 2020.

Address for correspondence:

Antti Ahola
Korkeakoulunkatu 3, 33014, Tampere, Finland
antti.ahola@tuni.fi