Norwegian University
of Life Sciences

Master's Thesis 2021 30 ECTS
Faculty of Science and Technology

Metamodelling of a Computational Model of Cardiac Physiology Using Multivariate Regression and Deep Learning

Ashesh Raj Gnawali
Data Science (M.Sc)

Acknowledgement

First of all, I would like to thank Prof. Kristin Tøndel for her continuous feedback and constructive recommendations throughout the thesis period. Also, I am grateful to her for awakening my interest and providing a thorough grounding in the principles of cardiac electrophysiology.

Further, I would also like to thank my friend Puspa Subedi for encouraging and supporting me on every little setbacks faced during the project.

Finally, I am indebted to my parents for their love, support, and guidance, without which I would never have come this far in life.

Ashesh Raj Gnawali

Ås, 9th March 2021

Abstract

The primary goal of this thesis is to model the heart function. This thesis investigates how data-driven modelling might help with this. Mechanistic models, which are theory-driven and guided by a system of differential equations that describe a well known mechanical, biological and chemical phenomenon or processes, are used to model a majority of complex biological processes. These models exhibit a complex high-dimensional system and a high computational cost. However, metamodels are data-driven, and are calibrated using the input-output data obtained by running a large number of simulations using the mechanistic model. Metamodels are known to reduce computational demand, aid in sensitivity analysis, model comparison, and assist in model parameterization with reference to measured data. This thesis explores two metamodelling approaches, HC-PLSR, and Deep Learning to emulate the Pandit-Hinch-Niederer model that couples cellular functions for rat cardiac excitation-contraction. The input parameters for simulating the mechanistic model were varied using Latin hypercube sampling and the generated action potentials were recorded for 250ms. Additionally, both the classical and inverse metamodelling techniques were used to map input-output relationships. The results reveal that metamodelling using deep learning is a powerful emulator, while the HC-PLSR metamodelling enables a more comprehensive inference of the model behavior. The results also highlight the importance of subspace analysis in explaining the broad spectrum of behavior that complex models display.

Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Full Form
AP	Action Potential
ANN	Artificial Neural Network
FFN	Feed Forward Network
FCM	Fuzzy C-means
HCPLSR	Hierarchical Cluster-based Partial Least Squares Regression
LHS	Latin Hypercube Sampling
PC	Principal Component
PLS	Partial Least Squares
PLSR	Partial Least Squares Regression
MSE	Mean Squared Error
SVM	Support Vector Machine
RF	Random Forests
KNN	K Nearest Neighbors
LR	Logistic Regression
DT	Decision Trees
SERCA	Sarcoendoplasmic reticulum (SR) calcium transport ATPase

1 Introduction

The mechanistic models aimed at realistically describing biological processes such as the generation of the cardiac action potential are complex. They incorporate a large number of parameters and state variables [1]. These parameters and state variables are linked together by a set of non-linear differential equations interconnected by complex feedback mechanisms [2]. Hence, it is very challenging to determine the relationship between such model outputs and input parameter variation. The complex nature of these models also make them computationally expensive [3].

Metamodels, i.e. statistical/machine learning-based approximation models, have been successfully used to decipher the input-output relationships of such mechanistic models [4]. A metamodel is calibrated from the outputs generated by simulating these mechanistic models with a large number of varied inputs. The inputs are varied so that they cover the entire biologically relevant input space, thus ideally covering the space of feasible model outcomes/behaviour. Metamodels are practical tools in sensitivity analysis, parameter fitting and reduction of computational cost [2].

Ordinary Least Squares (OLS) regression and response surface methods based on OLS are common choices for creating metamodels. These methods deal with predicting a single variable at a time, often neglecting the high correlation between output variables [2]. These methods also require the regressors to be linearly independent [5] . However, Li et.al.[6] argue that the regressors are not linearly independent in most biological modelling situations, making the OLS-based metamodels less reliable. Artificial Neural Networks (ANN) and Partial Least Squares Regression (PLSR) have been portrayed as effective tools to emulate complex mechanistic models as these do not require the regressors to be linearly independent [2]. ANNs emulate complex dynamic models with high precision, but a drawback in this approach is the interpretability [7]. ANNs do not provide insights into how all the inputs, auxiliaries, and outputs are related.

PLSR is an efficient tool as it maximizes the covariance between regressors and the responses. It also accounts for model stability by considering intercorrelations between response variables [2]. PLSR aids in dimensionality reduction by converting the input, intermediate states, and output variables into PLS components which are the important features that describe the complex system. Hierarchical Cluster Based Partial Least Squared Regression (HC-PLSR) [2] is an extension of PLS regression that divides the data with different behaviour into several subsets and performs local PLSR modelling on each of the subsets. Thus, HC-PLSR handles highly non-linear and non-monotone input-output relationships through local modelling [2].

In this paper, deep learning and HC-PLSR are used as metamodelling techniques to emulate the Pandit-Hinch-Niederer [8] model that couples cellular functions for cardiac excitation-contraction in rats. In order to test the performance of different metamodelling approaches, it was necessary to select a model that possessed a certain level of complexity and was reasonably simple such that it would be suitable for a Master Thesis to implement and run simulations. Hence, the PHN model fulfilling all the requirements is used as a basis for metamodelling in this thesis. This project investigates how efficiently these metamodels emulate the complex model and how these methods provide better insights for understanding the model behaviour.

2 Theory

2.1 Metamodelling

A metamodel, also known as a surrogate model, is a model that replaces an original model which is characterized by high complexity and computational cost. A metamodel is a data-driven approximation model and is calibrated with input-output data achieved from many simulations with the original, complex model.

Metamodelling aids in sensitivity analysis, i.e., determining the degree of dependency of the model output on variations in the input parameters and identifying hidden patterns and co-variance in the data [9][10]. It further helps model reduction by getting rid of the model insensitive parameters, resulting in reduced computational cost. Moreover, metamodels can be used as surrogate models to further reduce computational demand. Thus, metamodelling can improve fitting of the parameters' values to measured data and the interpretations of the modelling results, which again facilitates real-world use of the computational models. Metamodelling is used in several applications such as risk assessment, manufacturing systems, hospitals, fire stations, and by the military [11][12]. For instance, in medical analysis, metamodels are used to develop patient-specific models by analyzing model behaviour under various input conditions and determining the values of the parameters that replicate measured data.

Several supervised and unsupervised machine learning techniques can be used to develop a metamodel. Metamodels are categorized as classical or inverse. In a classical metamodel, the outputs are predicted functions of the input parameters, calibrated using e.g. regression. In an inverse metamodel, the outputs of a complex model are used to predict the input parameters. Since complex dynamic models are 'sloppy' by nature, meaning that many different combinations of the input parameter values can produce very similar model outputs [13], the inverse metamodelling is often much more challenging than the

classical metamodelling.

2.2 PLSR

Multivariate regression correlates the information in one data matrix **(X)** to another matrix **(Y)**. PLSR models the complex multivariate relationships by breaking down the input and the output matrix into independent covariance features [3]. PLSR is used extensively in the fields of chemometrics, bioinformatics, sensometrics, and neuroscience [14][15].

Let us consider a multivariate input data matrix $\mathbf{X}$ with *'I'* rows and *'J'*columns. The multivariate output matrix $\mathbf{Y}$ also has *'I'* rows and*'K'*columns. The decomposition of the $\mathbf{X}$ and $\mathbf{Y}$ matrices into latent variables called scores and loadings yield:

$$Y = U_A Q_A + F_A \tag{2.2.1}$$

$$X = T_A P_A + E_A \tag{2.2.2}$$

PLSR works by developing the scores and loadings so that the first score in $\mathbf{X}$, i.e., $\mathbf{t_1}$ has maximum covariance with the first score in $\mathbf{Y}$, $\mathbf{u_1}$. This enables us to predict the first score in $\mathbf{Y}$ from the first score in $\mathbf{X}$. Thus, if we can predict the score value in $\mathbf{Y}$, we can predict $\mathbf{Y}$. This is the main working principle of PLSR, it finds the components in such a way that their score values have maximum covariance [15]. Referring to the Figure 2.1, $\mathbf{u_1}$ has maximum covariance with $\mathbf{t_1}$, and $\mathbf{u_2}$ has maximum covariance with $\mathbf{t_2}$.

PLSR is related to the more commonly known technique Principal Component Analysis (PCA). PCA is a feature extraction technique that helps in data compression, whilst retrieving the most relevant information from the data. PCA finds the direction with the maximum variance in the data and projects it into a new subspace that has fewer dimensions compared to the original data [16]. The Principal Components (PCs) point towards the direction with maximum variance with a requirement that the new feature axes are perpendicular to each other. Hence, all the resulting PCs are uncorrelated as

they are mutually orthogonal.

PLSR does not consist of just doing PCA on **X** and PCA on **Y** in determining the scores and the loadings of the respective matrices. Instead of finding the significant variations in **X** and **Y** separately (as in PCA), PLSR looks for a direction in both which is suitable for correlating **X** scores with **Y** scores. Thus, it looks for the relevant **'Y'** information. In PLSR, the loadings in **X** and **Y** are rotated from the PCA solution. Hence, as the loadings are rotated, the scores change correspondingly. The correlation between **T** and **U** increases as the loadings change because PLSR is trying to maximize the covariance, leading to a higher correlation.

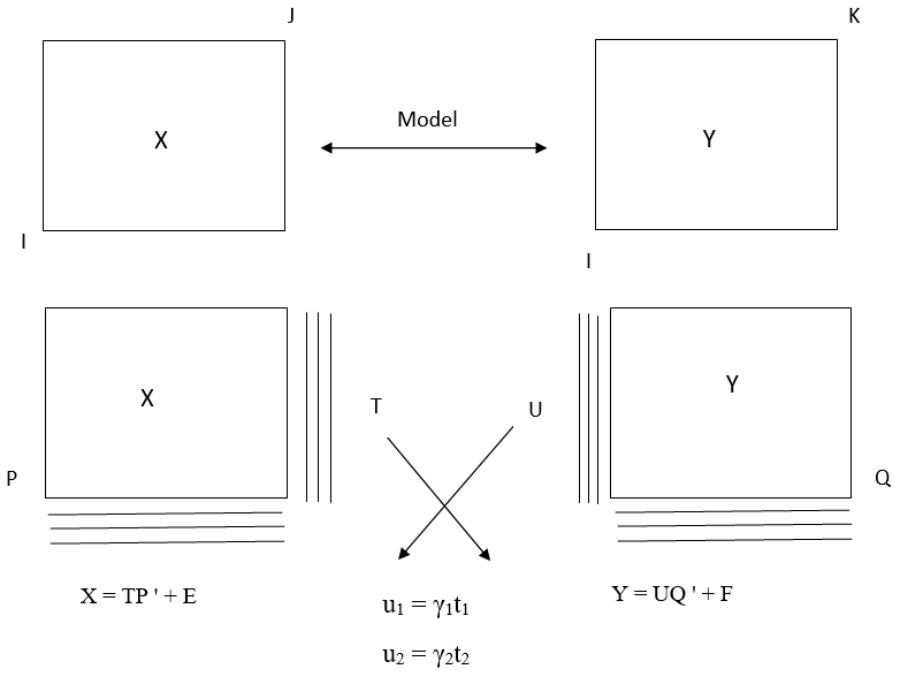


Figure 2.1: Illustration of the working of PLSR. Here, γ is the co-variance between the scores in **X** ($\mathbf{t_n}$) and the scores in **Y** ($\mathbf{u_n}$).

Iterative algorithms such as SIMPLS and NIPALS are used to minimize the residuals and maximize the covariance [15]. The scores and loadings obtained after the final iteration are known as the PLS components which describe the variation in the output matrix.

The PLS component's ordering is such that the first few components describe the largest parts of the variation in the output matrix.

The number of included PC, is typically chosen as the minimum number of components that together explain a sufficiently large portion of the **Y**-variance. In selecting the optimal number of PCs, one seeks the number of PCs where the explained **Y**-variance no longer increases by increasing the number of PCs. The information contained in the remaining PCs (amounting to the residuals) is then considered as noise and discarded from the model. The PCs included in the model thus represent a lower-dimensional subspace of the data, explaining the main patterns of covariance between **X** and **Y**. When all PCs are included, PLSR is identical to OLS regression.

2.3 HC-PLSR

HC-PLSR was presented by Tøndel *et.al.*[17] as a new approach for multivariate analysis in modelling highly non-monotone and non-linear input-output relations. HC-PLSR is a locally linear regression method that aims to split the regressor subspace into several, local, subspaces and identify each of the subspaces' distinct behaviour through local PLSR modelling. As a result, improved prediction accuracy and more in-depth insights into the input-output relations can be achieved. In an experiment by Tøndel *et.al.* HC-PLSR outperformed OLS and PLSR in metamodelling of dynamic models of a mouse ventricular myocyte [17] and the mammalian circadian clock [4].

In this method, a global PLSR model is calibrated using all available data in the calibration set. Clustering is then used to separate the data based on the global PLSR **X** and **Y** scores, and local PLSR models are calibrated based on the observations belonging to each cluster. For prediction of new/test data, the test set **X** -matrix is projected into the global PLSR model to predict the **X** and **Y** scores. Based on these scores, the test samples are classified into one of the clusters and the final prediction is made based on the local PLSR model of the cluster. HC-PLSR can be used with any clustering

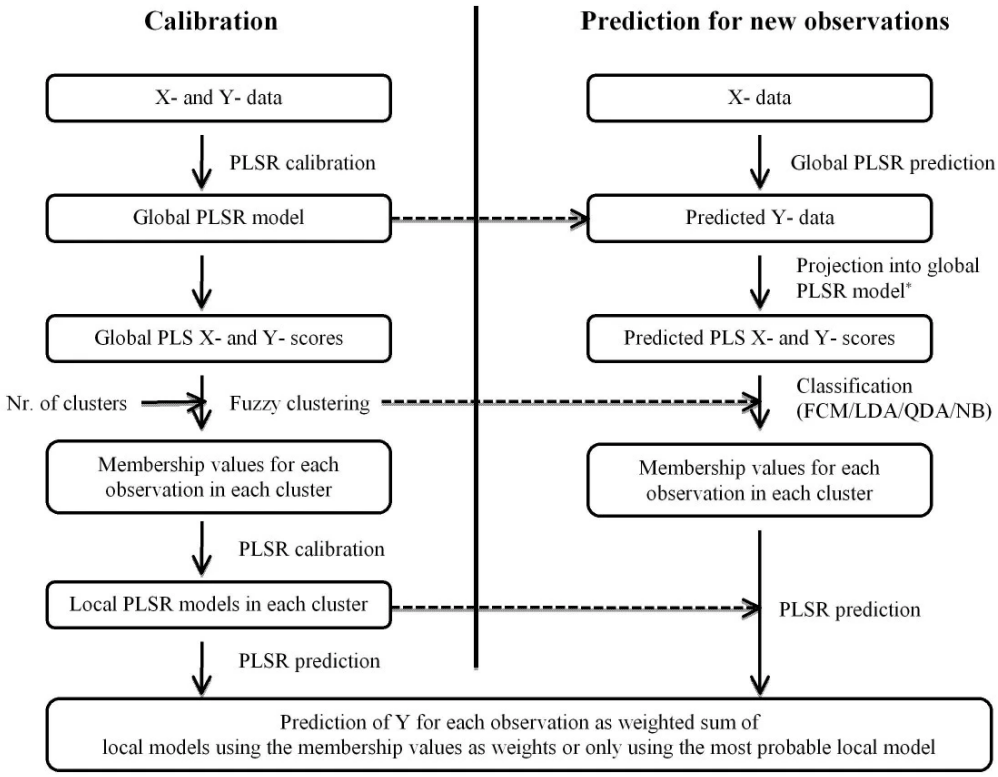


Figure 2.2: Illustration of HC-PLSR [2]

and classification method. In the original implementation, fuzzy C-means clustering was used, giving membership probabilities for each cluster. Hence, alternatively, the predicted response can be calculated as a weighted sum of the predictions achieved with each of the local models, where the cluster membership values serve as weights (an option only achieved when using soft clustering methods) [2].

2.4 Latin Hypercube Sampling

Latin hypercube sampling is a popular statistical sampling method used to generate random samples of input variables used for computer simulation experiments [18]. These samples are taken from a multi-dimensional distribution. This approach generates sam-

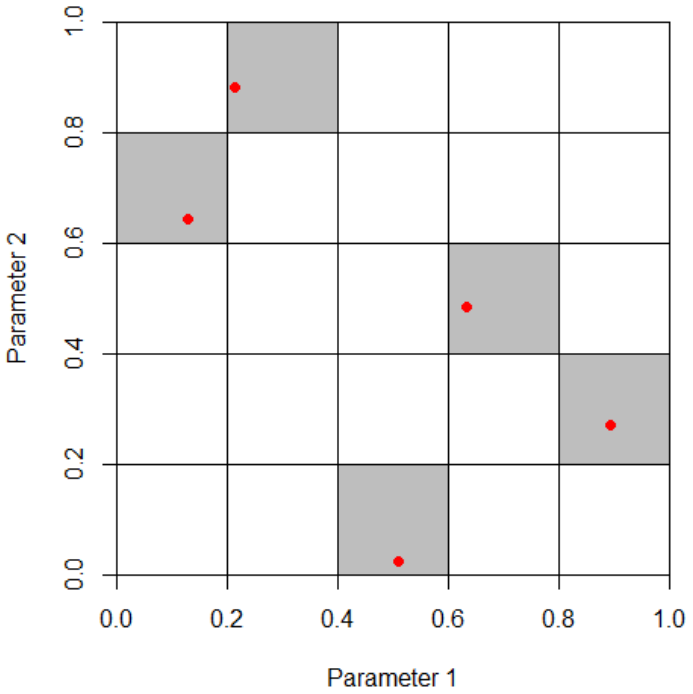


Figure 2.3: Illustration of a Latin Square

ples by stratifying the input probability distribution, which breaks down the cumulative curve into equal intervals [19]. A sample is drawn at random from each of these intervals. A Latin square (Figure 2.3), is a square grid if it contains a single sample in each row and each column. The Latin hypercube generalizes this concept to any number of dimensions. LHS is efficient in multivariate statistical analysis because it is a memory-based approach. It ensures that a sample was taken in each row and column generating near-random samples. This type of stratified sampling is especially useful in high-dimensional systems, since it ensures that all regions of the sampling space are covered.

2.5 Feature Importance

Feature importance is a technique that ranks the input features based on their ability to predict the target variable. The feature importance aids in dimensionality reduction

by getting rid of input features that are irrelevant in predicting the target variable and, thus, insensitive. Selection of the critical features during model training can improve the model's test accuracy as they have a significant influence in predicting the target and are sensitive [20]. Thus, feature importance helps determine the degree of dependency of the model output in relation to the varied input features termed as a sensitivity analysis.

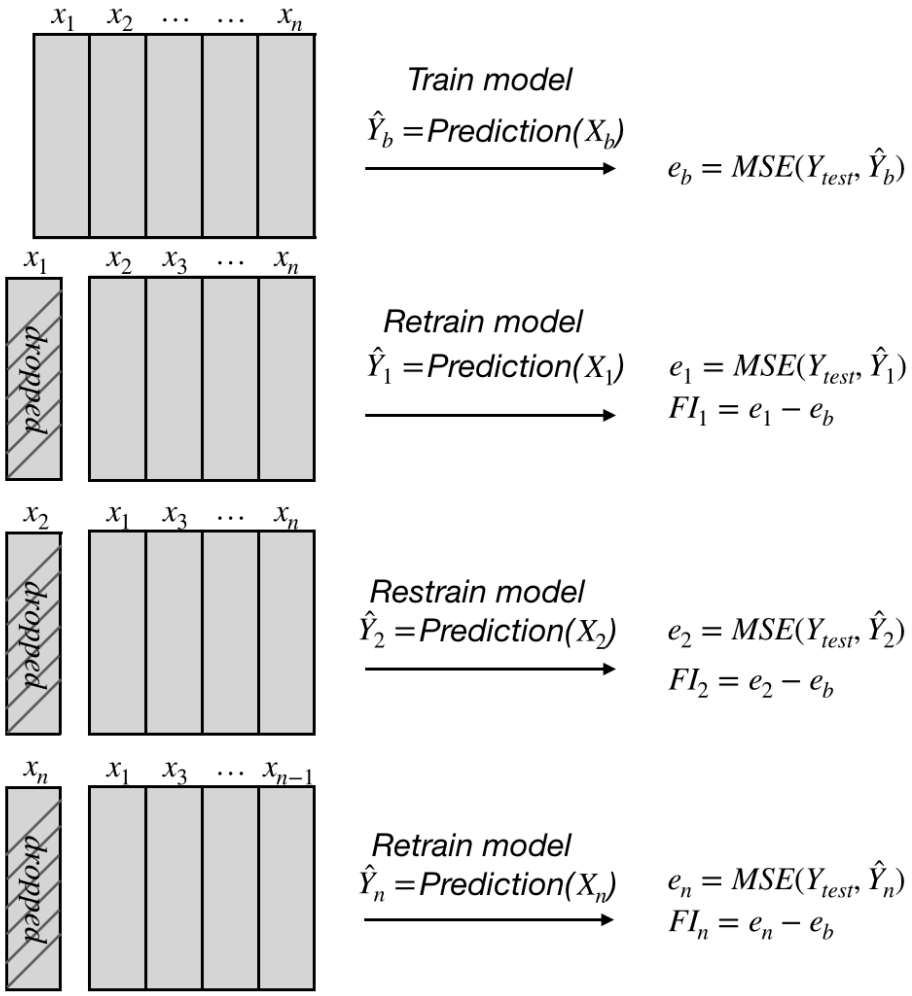


Figure 2.4: Illustration of drop column feature importance

One of the approaches for determining the feature importance is the drop column method. It works by dropping feature columns one at a time, exactly as it sounds. An illustration of the dropout feature importance algorithm is shown in Figure 2.4. Firstly,

a baseline model is fitted by including all the features. The base model error e_b, which represents the model performance, is evaluated from the baseline model. In the next step, the first feature f1 is removed from the training and the test set. A new model is fitted and the model performance e_1 is evaluated. The performance of the baseline model (original model with all features) is compared to that of the dropout model to determine the feature importance. The importance of the first feature is thus evaluated as:

$$\text{Importance of feature f1} = \text{ baseline performance } - \text{ dropout performance } = e_b - e_1 \tag{2.5.1}$$

The larger the decrease in performance, the more important the feature is for predicting the target variable. This process is repeated for each of the features in the model. The drawback with this approach is that an increasing number of features causes an increase in the number of models that are to be fitted, making this approach computationally expensive.

Another approach for evaluating feature importance is the feature importance permutation. The schematic illustration of the feature importance permutation method is shown in the Figure 2.5. Like the dropout feature importance method, a baseline model is trained and its baseline performance e_b on the test data is recorded. Instead of dropping the first feature, the rows of the first feature (f1) are permuted. The predictions are made from this modified test set, and its performance e_1 is determined. The feature importance calculation method is the same as shown in the equation 2.5.1. This procedure also needs to be repeated for each of the features in the model, however, this approach does not require retraining the model as for the dropout method since the importance is evaluated just from the permuted test set. Hence, the feature importance permutation method is utilized in this paper due to its low computation cost. A drawback of this method is that if the features are correlated, this approach tends to provide higher importance to one of the features and suppresses the other's importance.

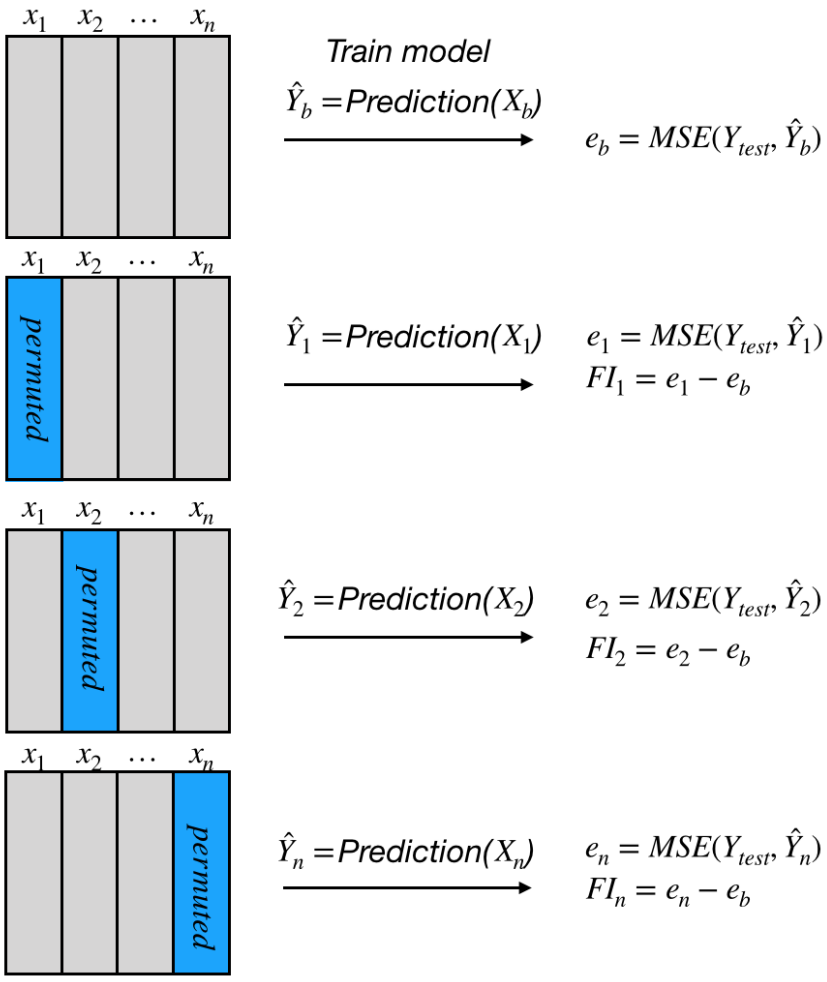


Figure 2.5: Illustration of feature importance permutation

2.5.1 Regression coefficients as measures of feature importance

In regression analysis, the regression coefficient quantifies the relationship between the input and the output (given that the regressors are scaled to equal variance prior to the analysis). A lower absolute value of the regression coefficient indicates a low correlation between the feature and the quantity of interest [21]. Thus, discarding the features with lower correlation, we improve the metrics associated with our prediction.

2.6 Classification

Classification is a supervised machine learning approach that predicts the categorical class of the observations. The classification model is trained with samples whose categorical label is known. The class label represents the group membership of each of the samples. The model learns a set of rules such that it can predict the class labels of the test samples. A problem can also have multiple class labels, giving multiclass classification.

2.6.1 Logistic Regression

Despite its name, logistic regression is a classification model. The performance of logistic regression is best when the classes are linearly separable [22]. It is often used in binary classification problems but can also be implemented in multiclass classification by applying the one-versus-rest technique (OvR). The OvR breaks down multi-class classification into one binary classification problem per class.

The logistic regression predicts the class probability of a particular sample with a sigmoid function. The sigmoid function takes the real values as inputs and transforms them into the range [0,1] with an intercept at $\phi(z) = 0.5$. The prediction probability is converted to the binary outcome through a threshold function.

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{2.6.1}$$

$$\hat{y} = \begin{cases} 1 & \text{if } \phi(z) \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{2.6.2}$$

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} \left[-y^{(i)} \log\left(\phi\left(z^{(i)}\right)\right) - \left(1 - y^{(i)}\right) \log\left(1 - \phi\left(z^{(i)}\right)\right) \right] \tag{2.6.3}$$

The equations 2.6.1, 2.6.2, 2.6.3 describe the sigmoid function, threshold function and the cost function of logistic regression, respectively.

2.6.2 K Nearest Neighbors (KNN)

The KNN classifier is a so-called lazy learning algorithm. Unlike the parametric models such as logistic regression, linear discriminant analysis, and perceptron, it does not estimate a set of parameters to learn the classification problem. Instead, KNN memorizes the training data and is therefore referred to as lazy, implying that the model training cost is zero [16].

The workflow of the KNN algorithm can be summarized in the following three points:

1. The algorithm starts with the user choosing the number of neighbours *'k'*.

2. It finds the nearest neighbours from the training samples, i.e., neighbouring samples that have similar properties or are closest to the test sample. The closeness is evaluated by using distance metrics. Commonly, Euclidean and Minkowski's distance metrics are used.

3. The sample is classified based on a majority vote from its *'k'*- nearest neighbours.

An advantage of this method is that it adapts quickly to new data, as it memorizes the training data and excludes training. A drawback of this approach is that the computational complexity increases linearly with the number of samples in the training dataset.

2.6.3 Support Vector Machines (SVMs)

SVMs (Figure 2.6) are another widely used classification algorithm, where the margin, i.e., the distance between the training samples and the decision boundary, is maximized. The samples that are closest to this boundary are referred to as support vectors. The basic intuition is that a wider margin leads to a lower generalization error [23]. In contrast, smaller margins often lead to overfitting.

The positive and negative hyperplanes which are parallel to the decision boundary can be described as:

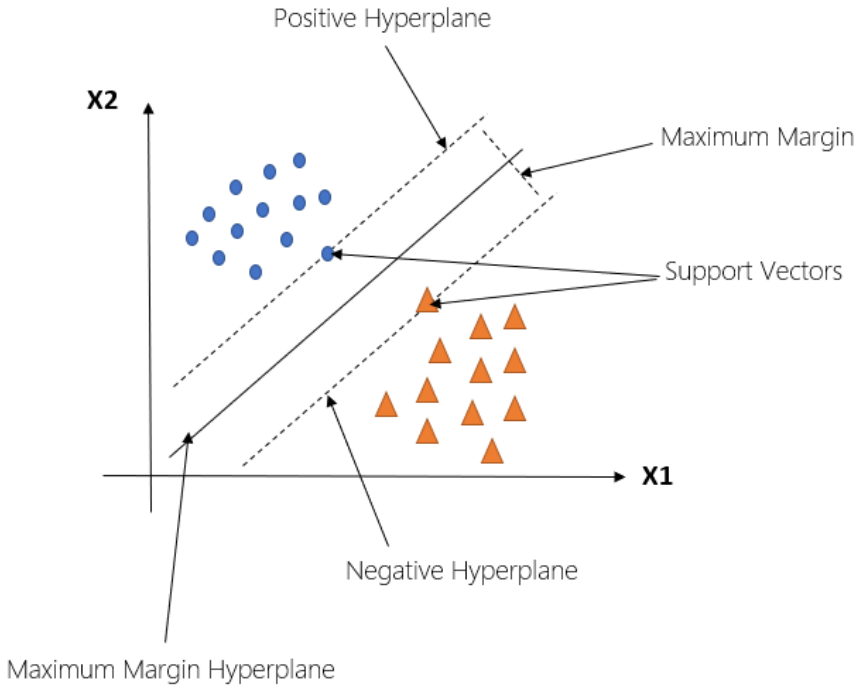


Figure 2.6: Illustration of maximum margin and support vectors

$$w_0 + \boldsymbol{w}^T \boldsymbol{x}_{pos} = 1 \tag{2.6.4}$$

$$w_0 + \boldsymbol{w}^T \boldsymbol{x}_{neg} = -1 \tag{2.6.5}$$

The distance between the positive and negative hyperplane or the "margin" is achieved by subtracting equation 2.6.4 and 2.6.5.

$$\boldsymbol{w}^T \left(\boldsymbol{x}_{\text{pos}} - \boldsymbol{x}_{\text{neg}} \right) = 2 \tag{2.6.6}$$

We normalize the equation 2.6.6 by the length of weight vector **w** as follows:

$$\|\boldsymbol{w}\| = \sqrt{\sum_{j=1}^{m} w_j^2} \tag{2.6.7}$$

$$\frac{\boldsymbol{w}^T\left(\boldsymbol{x}_{pos}-\boldsymbol{x}_{neg}\right)}{\|\boldsymbol{w}\|}=\frac{2}{\|\boldsymbol{w}\|} \tag{2.6.8}$$

Thus, the objective of SVM is to maximize the margin by maximization of $\frac{2}{\|w\|}$ provided that the samples are classified correctly. However, in practice, the inverse term $\frac{1}{2}\|\boldsymbol{w}\|^2$ is minimized as it can be solved efficiently using quadratic programming [24].

Dealing with non-linearly separable cases Vladimir Vapnik introduced in 1995 the idea of slack variables which gave rise to the soft-margin classification. The idea behind the slack variable was relaxing the linear constraints when dealing with non-linearly separable data [25]. This allows the optimization process to converge by penalizing misclassification by adding suitable cost penalty.

The slack variables are added to the linear constraints as follows:

$$\begin{aligned} w_0+\boldsymbol{w}^T\boldsymbol{x}^{(i)} &\geq 1-\xi^{(i)} \text{ if } y^{(i)}=1 \\ w_0+\boldsymbol{w}^T\boldsymbol{x}^{(i)} &\leq -1+\xi^{(i)} \text{ if } y^{(i)}=-1 \end{aligned} \tag{2.6.9}$$

The new objective to be minimized is then:

$$\frac{1}{2}\|\boldsymbol{w}\|^2+C\left(\sum_i \xi^{(i)}\right) \tag{2.6.10}$$

The variable 'C' controls the penalty for misclassification. A high value of 'C' means a large error penalty and a lower value corresponds to a lower penalty. Thus, the parameter 'C' controls the width of the margin and adjusts the bias-variance tradeoff [16].

2.6.4 Decision Trees

Decision trees are popular classification models when the interpretability of the model result is of high importance. Decision trees separate data into classes by asking a number of questions learned from the features [26]. The decision tree starts at the root and divides the data based on the largest information gained. This is an iterative process which is

repeated until a pure child node is achieved. This often leads to overfitting as the depth of the tree is increased, and it is thus often necessary to set a maximum limit for the tree depth, something known as pruning the tree [22]. The objective is to maximize the information gained by splitting the nodes at the most informative features.

$$IG\left(D_p, f\right) = I\left(D_p\right) - \sum_{j=1}^{m} \frac{N_j}{N_p} I\left(D_j\right) \tag{2.6.11}$$

where: f = feature on which the split is performed

D_{p} = parent dataset

D_{j} = dataset of the j^{th} child node

I = impurity measure

N_{p} = number of samples in the parent node

N_{j} = number of samples in the j^{th} child node

Thus, information gain is the difference between the parent node's impurity and the sum of the impurities of the child nodes. Hence, the information gain is higher when the sum of the impurities of the child nodes is lower.

The impurity measures used in binary decision trees are Gini impurity ($\mathrm{I_G}$) and entropy ($\mathrm{I_H}$) [27].

$$I_H(t) = -\sum_{i=1}^{c} p(i \mid t) \log_2 p(i \mid t) \tag{2.6.12}$$

$$I_G(t) = \sum_{i=1}^{c} p(i \mid t)(1 - p(i \mid t)) = 1 - \sum_{i=1}^{c} p(i \mid t)^2 \tag{2.6.13}$$

where, $p(i \mid t)$ is the proportion of the samples that belong to class c and node t.

If all the samples at a particular node belong to the same class, the entropy is zero. In contrast, it is at maximum if the class distribution at a node is uniform. Thus, it is

evident that entropy tries to maximize joint information in a tree. Likewise, Gini impurity tries to minimize the probability of misclassification.

2.6.5 Random Forests

Random forests are extensively used in classification problems because of their good classification performance, simple interface and scalability [22]. In this method, multiple decision tress are overfitted by making them deep. These trees with high variance help build a robust final model [28]. The algorithms for random forests can be summarized as:

1. Selecting random samples from training data with replacement. These samples of data are also known as bootstrap samples.

2. Building a decision tree from the randomly selected samples by selecting a subset *'d'* without replacement and splitting the tree node where the information gain is at maximum.

3. The steps 1 and 2 are repeated *'k'* times, where k is the total number of trees in random forest.

The model's final prediction depends on a majority vote of the classes predicted by all the individual decision trees in the random forest.

A user does not have to worry much about hyperparameter tuning in random forest. It is not necessary to prune the trees, as it performs best when the trees are overfitted and the final prediction depends on a majority vote [22]. Also, a greater number of trees in the forest leads to an increased performance as compared to using less trees. However, keeping a small bootstrap size decreases the model performance, as it increases the randomness of the forest and decreases overfitting. This means that the trees in the forest will be very different from each other, as a single sample might not be placed multiple times in the bootstrap [22]. Thus, it does not fit the training data more closely, leading to a lower

generalization performance. Meanwhile, increasing the bootstrap sample size increases overfitting, making the trees in the forest more similar to each other. Hence, fitting the training data more closely and making robost predictions.

2.7 Regularization

Regularization helps reduce the complexity of a model, handles collinearity, filters the noise, and prevents overfitting [16]. It adds additional bias to the weights by penalizing higher weight values.

$$L2: \quad \lambda \|w\|_2^2 = \lambda \sum_{j=1}^{m} w_j^2 \tag{2.7.1}$$

The λ is a regularization parameter and controls the strength of regularization. A high value of lambda keeps the values of the weights lower. In scikit-learn [29], the parameter 'C' is used to control the strength of regularization. 'C' is the inverse of λ. Consequently, decreasing the value of C increases the regularization strength.

$$L1: \|\boldsymbol{w}\|_1 = \sum_{j=1}^{m} |w_j| \tag{2.7.2}$$

Compared to L2 regularization, L1 regularization replaces the square of the weights by the sum of the absolute values of the weights. In contrast to L2 regularization, it produces sparse feature vectors, i.e., the weights of some of the features are zero. Sparsity is useful when handling high-dimensional data as it removes irrelevant dimensions. Intuitively, L1 regularization can also be used for feature selection.

Referring to the Figure 2.7, the contours are the cost function for weight coefficients $\mathbf{w_1}$ and $\mathbf{w_2}$. The cost function is the sum of squared errors (SSE) [30]. The aim is to find the optimal combination of weight coefficients which yields a low cost for the training data. The shaded ball represents the L2 regularization term which is quadratic in nature. The combination of weights cannot fall outside the shaded area. Thus, increasing the

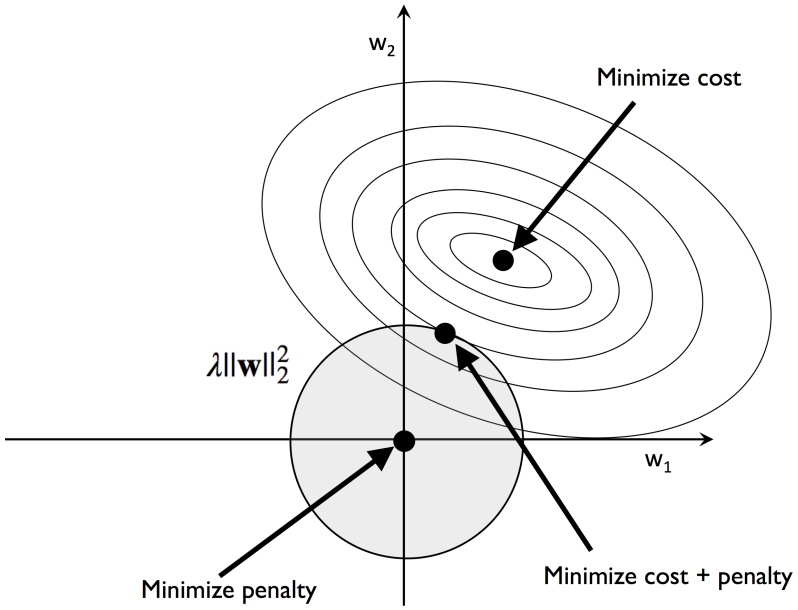


Figure 2.7: Illustration of L2 regularization [22]

lambda value will increase regularization and reduce the shaded area, giving extremely low weights. However, this might not lead to the shaded area's intersection with any of the cost contours. Hence, it is necessary to find a suitable value of lambda that increases the regularization budget while decreasing the cost [22].

The concept behind L1 regularization, is the same as that of L2, but since it is a sum of absolute weight coefficients, its regularization term is represented by a diamond shape. In Figure 2.8 , we can see that the contour of the cost function and the diamond area intersects where the $\mathbf{w_1} = 0$. Hence, the intersection of the cost function (ellipses) and the boundary of the L1 diamond is bound to be located at the axes, which favors sparsity [22].

$$\lambda_{\text{enet}}(\beta) = \gamma\left[\alpha\lambda_1(\beta) + (1-\alpha)\lambda_2(\beta)\right] \tag{2.7.3}$$

Elastic Net regularization is a combination of both L1 and L2 regularization. Equation

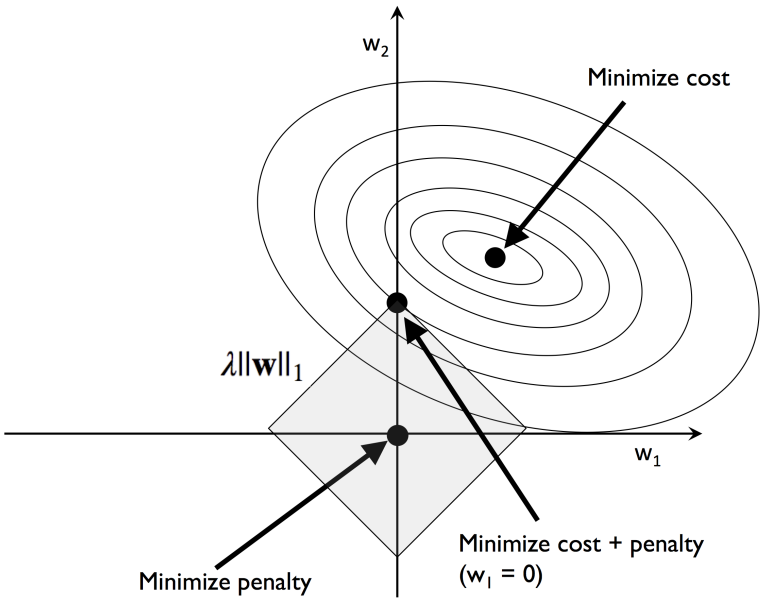


Figure 2.8: Illustration of L1 regularization [22]

2.7.3 shows that the L1 part ($\lambda_1(\beta)$) of the penalty generates a sparse model, whereas the quadratic part of the penalty ($\lambda_2(\beta)$) removes the limitation on the number of selected features, encourages a grouping effect and stabilizes the L1 regularization path. The parameter γ controls the regularization strength.

2.8 Feature Selection

Feature selection refers to the selection of only a subset of meaningful features from the original features of a dataset. Feature selection supports dimensionality reduction by getting rid of irrelevant features and,in turn, avoids overfitting. It further reduces the generalization errors, increases computational efficiency, and sometimes increases the predictive capability of a model [31].

2.8.1 Sequential Backward Selection (SBS)

The SBS algorithm reduces the number of features in the original subspace to the user defined feature counts. A criterion function **'J'**, which is to be minimized, decides whether to select a feature or not. The criterion function can be as simple as the difference in the model's performance before and after the removal of a feature [32]. Thus, getting rid of the features that cause the lowest performance loss.

SBS is a greedy algorithm. A greedy algorithm produces sub-optimal results as it utilizes locally optimal solutions whilst an exhaustive search algorithm evaluates all possible outcomes. Hence, the latter is computationally expensive but guarantees an optimal solution to a problem.

2.8.2 Repeated Elastic Net Technique (RENT)

RENT is a feature selection technique that selects features from an ensemble of models. These models are trained from different subsets of data and are approximately unique. The elastic net regularization might select different features in each of these unique models. RENT analyzes the weight distribution of the weight sizes of the features in all the models and uses a user defined specific threshold value for feature selection criteria [33].

As described in equation 2.7.3, a pre-defined combination of several values of γ and α is passed before RENT. The best combination of γ and α is determined through five-fold

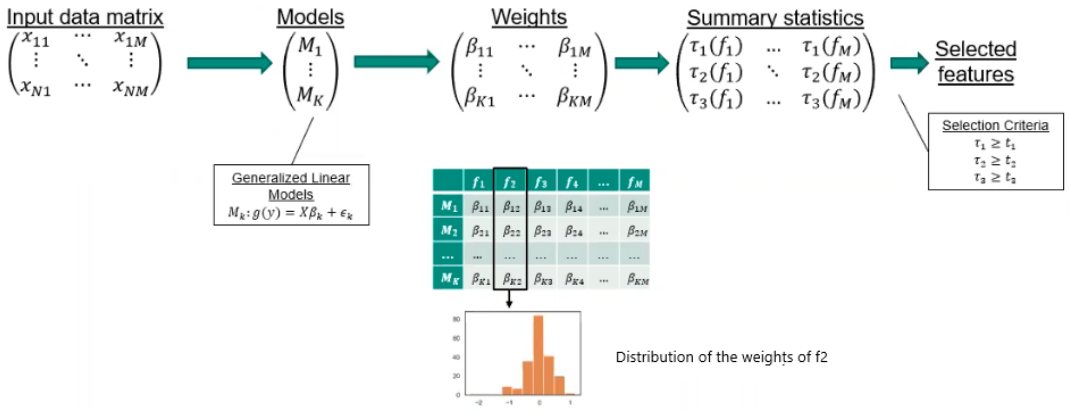


Figure 2.9: Illustration of working of RENT [34]

cross-validation. This best combination of values is utilized by RENT to train the unique ensemble models for feature selection.

The RENT starts by building *'k'* different models from the input data set. It performs *'k'* train-test-splits from the original dataset giving different test and training datasets. This results in a weight matrix having weights for each of the *'k'* models. For example, in Figure 2.9, β_{11} is the weight of the first feature in Model 1. Utilizing the weight matrix, we obtain a statistical summary of each of the feature weights. An example of a distribution of weights of the second feature is shown in the histogram.

$$\begin{aligned} \tau_1(\beta_n) &= c(\beta_n) \\ \tau_2(\beta_n) &= \frac{1}{K}\left|\sum_{k=1}^{K} \operatorname{sign}(\beta_{n,k})\right| \\ \tau_3(\beta_n) &= t_{K-1}\left(\frac{\mu(\beta_n)}{\sqrt{\frac{\sigma^2(\beta_n)}{K}}}\right) \end{aligned} \tag{2.8.1}$$

A feature should pass all the three criteria (2.8.1) embedded in RENT to get selected. The first criterion deals with the number of times a feature is selected on these *'k'* different models. The idea behind this is that a feature that is selected on most of the models is most likely important. Hence, the first criterion gets rid of the features that don't contribute much to the prediction. The second criterion is all about the stability of the weights of the features. A feature is deemed unstable if its weights are sometimes nearly zero and

at other times it shifts between a positive and a negative value. The second criterion removes the unstable features. Likewise, the third criterion eliminates the features that have weights close to zero in all the *'k'* models. The third criterion is important because even if a feature passes the first and the second criteria, it ensures that the features with weights close to zero are not selected. All these criteria are directed by thresholds τ_1, τ_2, and τ_3 respectively [33].

Cut-off Thresholds

- $\tau_1 > 0.9$
- $\tau_2 > 0.9$
- $\tau_3 > 0.975$

	f1	f2	f3	f4	f5
τ_1	0.5	0.17	1.0	1.0	0.95
τ_2	0.5	0.13	1.0	1.0	0.95
τ_3	0.999	0.999	1.0	1.0	1.0

Figure 2.10: Illustration of the feature selection criteria [34]

The threshold values in this example (2.10) are $\tau_1 = 0.9$, $\tau_2 = 0.9$ and $\tau_3 = 0.975$. This means that a feature should be selected ninety percent of the times in *'k'* models, the stability of the sign of the weight should be above ninety percent and the weights should be far from zero. We can see that all three criteria are fulfilled by the third, fourth and the fifth feature and thus are selected. On the other hand, we do not select the first and the second feature as they do not pass the first and the second criteria despite fulfilling the third criteria.

2.9 Cross-Validation

Cross-validation approaches are used to evaluate the generalization performance of a model, i.e., how well it performs on unseen data. It helps find a sweet spot in bias-variance tradeoff. The two most popular cross-validation approaches are leave-one-out cross-validation and k-fold cross-validation.

2.9.1 Holdout Cross-Validation

This is a classical cross-validation technique that splits the input data into a training set, a validation set and a test set. The training set is used to set different models while the validation set is utilized for hyperparameter tuning and performance evaluation for model selection. The test set which the model has never seen before is then used to evaluate the generalization performance of the model. This gives a less biased estimate of the model as compared to using only a training set and a test set [35]. On the contrary, this method is sensitive to how the input data is split into the three different subsets, as the estimates might vary between different splits [22].

2.9.2 K Fold Cross-Validation

'k' repetitions of the holdout method on *'k'* subsets of the data leads to k-fold cross-validation [22]. This method splits the training data into *'k'* subsets and 'k-1' of the subsets are used for model training. The remaining single fold is then used for model performance evaluation. Each of the *'k'* folds are held out once. This results in *'k'* models and *'k'* estimates. The overall performance of the model is evaluated as the average between the estimates of the *'k'* models. These estimates are bound to be less sensitive than that of the individual predictions. Often K-fold cross-validation is used for hyperparameter tuning and finally all of the data is used to train the model with the selected hyperparameter values. As k-fold samples the observations without replacement, each of the samples get selected into the training set and the validation set exactly once. Consequently, the estimates have lower variance [22].

Figure 2.11 illustrates a five fold cross-validation. In the initial iteration, the last four training data subsets are used for training while the first subset is used for validation. The final performance of the model is evaluated as the average of all errors of the iterations. Studies show that a ten-fold cross-validation is a good choice for most problems [37]. However, an increasing number of folds leads to an increase in computational demand.

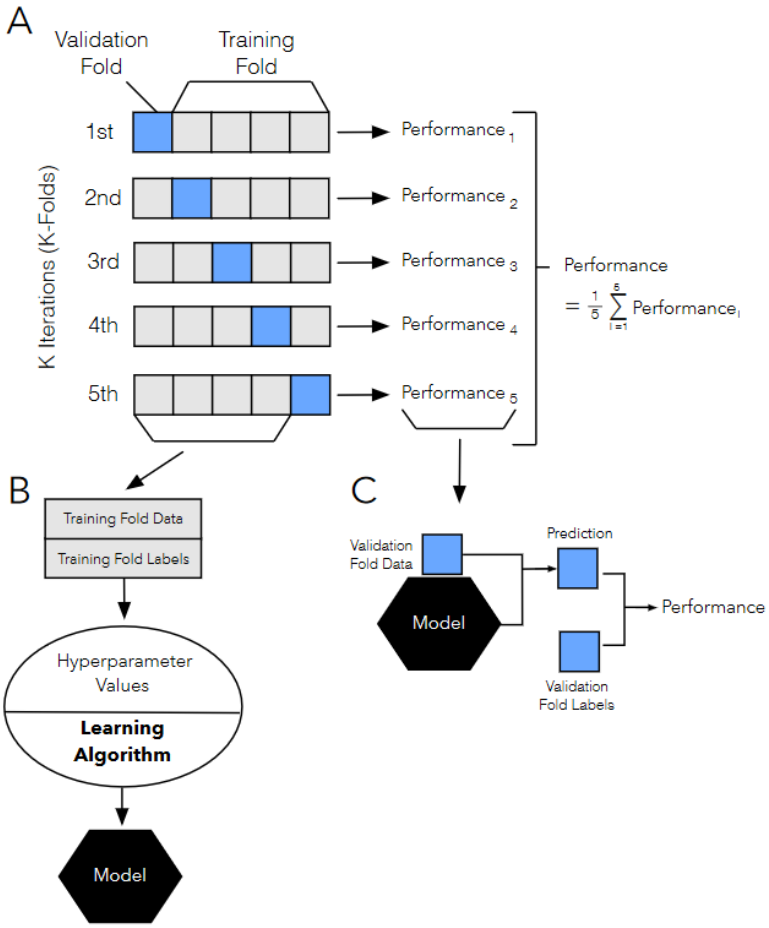


Figure 2.11: Illustration of five fold cross-validation [36]

Another approach known as stratified K-fold yields better bias-variance tradeoff when the data's class labels are disproportionate [37]. This method ensures that the class proportions are equal in each of the folds of the training dataset.

2.10 Evaluating the performance of a model

2.10.1 Confusion Matrix

A confusion matrix is a square matrix that reports the performance of a model. It provides the count of true positives (TP), false positives (FP), true negatives (TN), and

false negatives (FN).

2.10.2 Evaluation Metrics

Prediction accuracy is the most widely used metric when evaluating a model's performance. Nevertheless, several other metrics are used to measure the relevance of a model such as precision, recall, and f1 score. In classification problems, accuracy and prediction error provide information on the number of samples that are misclassified. The prediction error and accuracy are evaluated as:

$$PredictionError = \frac{FP + FN}{FP + FN + TP + TN} \tag{2.10.1}$$

$$PredictionAccuracy = \frac{TP + TN}{FP + FN + TP + TN} = 1 - PredictionError \tag{2.10.2}$$

In most of the imbalanced class problems, True positive rate (TPR) and False Positive Rates (FPR) are used as the evaluation metrics.

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{2.10.3}$$

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP} \tag{2.10.4}$$

The precision and recall are related to TPR and TNR. The recall is the same as the true positive rate. The precision describes how surely the model predicts true positives whilst the recall explains the degree by which the model does not miss any positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{2.10.5}$$

$$Recall = TPR = \frac{TP}{P} = \frac{TP}{FN + TP} \tag{2.10.6}$$

In general, another score metric called f1 score is used, which is a combination of precision and recall.

$$F1 = 2 * \frac{PRE \times REC}{PRE + REC} \tag{2.10.7}$$

In regression analysis, Mean Squared Error (MSE) is commonly used to evaluate model performance. MSE is the average value of the sum of squared cost that is minimized while fitting the regression model. To get a metric that is on the same scale as the predicted values, one often reports the square-root of this value, i.e. the Root Mean Squared Error (RMSE).

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)} \right)^2 \tag{2.10.8}$$

Alternatively, a standardized version of MSE known as the coefficient of determination (R^2) can be used. R^2 is evaluated as:

$$SST = \sum_{i=1}^{n} \left(y^{(i)} - \mu_y \right)^2 \tag{2.10.9}$$

$$R^2 = 1 - \frac{SSE}{SST} \tag{2.10.10}$$

where, SSE= sum of squared error and SST= total sum of squares

The value of R^2 lies between 0 and 1 for the training dataset but it can be negative for the test data. The coefficient of determination is 1 if the model fits the data perfectly with zero MSE.

2.11 Clustering

Clustering is an unsupervised classification technique that organizes unlabelled data into meaningful clusters. This technique does not have any prior information about the group memberships of the observations. Clustering helps identify the hidden structures in the

data. Groups of observations that are similar or share a certain degree of similarity (and are dissimilar to other observations) are gathered together. Clustering is e.g. used to identify customer behaviours, developing distinct marketing programs, and in recommender systems [38].

2.11.1 Hard and Soft Clustering

Hard clustering algorithms such as the K-means algorithm assigns each sample strictly to a single cluster. In contrast, soft clustering (fuzzy clustering) algorithms assign a sample to more than one cluster. The most widely used soft clustering algorithm, FCM (Fuzzy C-means) [39], was developed by James C. Bedzek in the 1980s, which improved Joseph C. Dunn's original idea.

2.11.2 K-Means Clustering

K-Means is a prototype-based clustering technique. Each of the clusters is represented by a prototype, i.e., centroid (average) of similar points with continuous features. K-means performs better with spherical clusters, but a downside with this approach is that we have to specify the *'k'* (number of clusters) value [16]. An inappropriate value of *'k'* leads to poor clustering.

The workflow of the K-Means algorithm is summarized in the following points:

1. Randomly assigning *'k'* cluster centroids from the observations as initial clusters.
2. Assigning each of the observations to its nearest centroid.
3. Moving centroids to the centre of the observations that were assigned to it. The steps 2 and 3 are repeated until the clusters do not change or after the tolerance or the maximum number of iterations is reached.

Euclidean distance measure is used to determine the similarity between the samples.

The lower distance between the two points x and y indicates that the samples are similar. The distance between two points x and y on m-dimensional space is calculated as:

$$d(\boldsymbol{x}, \boldsymbol{y})^2 = \sum_{j=1}^{m} (x_j - y_j)^2 = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{2.11.1}$$

Where *'j'* refers to the *jth* feature column of the sample points x and y. This Euclidean distance metric is used to modify the K-means algorithm as an optimization problem by minimizing the within-cluster sum of squared error (SSE). SSE is also sometimes referred to as cluster inertia.

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{k} w^{(i,j)} \left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)}\right\|_2^2 \tag{2.11.2}$$

2.11.3 Fuzzy Clustering

The working principle of FCM is similar to that of the K-means algorithm except that, unlike K-means, the FCM assigns each observation a cluster probability. This value describes the belongingness of the sample to different clusters. The sum of the membership values for a given sample is one. The algorithm starts by assigning *'k'* number of centroids and randomly setting membership values for each observation. The cluster centroids are computed as the mean of the cluster members. The cluster membership for each point and the centroids are updated iteratively until convergence or the maximum number of iterations has been reached [22]. The loss function to minimize in FCM is the within-cluster sum of squared error, based on a chosen distance measure:

$$J_m = \sum_{i=1}^{n} \sum_{j=1}^{k} w^{m(i,j)} \left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)}\right\|_2^2 \tag{2.11.3}$$

where, *w(i,j)* is a real value that represents cluster membership. The exponent m is the fuzziness coefficient which typically takes the value two and controls the fuzziness.

Increasing the fuzziness coefficient leads to fuzzier clusters as the cluster memberships become smaller.

$$w^{(i,j)} = \left[\sum_{p=1}^{k} \left(\frac{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)} \right\|_2}{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(p)} \right\|_2} \right)^{\frac{2}{m-1}} \right]^{-1} \tag{2.11.4}$$

$$\boldsymbol{\mu}^{(j)} = \frac{\sum_{i=1}^{n} w^{m(i,j)} \boldsymbol{x}^{(i)}}{\sum_{i=1}^{n} w^{m(i,j)}} \tag{2.11.5}$$

Equation 2.11.4 and 2.11.5 describe how the cluster membership and the cluster centres are calculated respectively.

2.11.4 The Elbow Method

The main issue in unsupervised learning is not knowing the conclusive answer. We do not have the ground truth values to evaluate the model performance. Thus, intrinsic methods such as a within-cluster sum of squared errors, also known as 'distortion', are used to compare clustering models' performance with different numbers of clusters [22]. The Elbow method is a graphical tool used to evaluate the optimal number of clusters for a task at hand. This method follows the idea that choosing a large number of clusters leads to a decrease in distortion. This is true, since the samples will be closer to their cluster centroids. However, this should be balanced against the complexity of the cluster model. Thus, this method plots the distortion for different values of *'k'* (number of clusters) and its optimal value is the one where the distortion no longer decreases significantly by increasing the number of clusters.

2.11.5 Silhouette Plots

Silhouette analysis [40] is a graphical tool that describes how tightly the samples in a cluster are grouped. The silhouette coefficient $s^{(i)}$ is evaluated as a difference between

the cluster cohesion ($a^{(i)}$) and cluster separation ($b^{(i)}$) divided by the greater of the two values, shown as:

$$s^{(i)} = \frac{b^{(i)} - a^{(i)}}{\max\{b^{(i)}, a^{(i)}\}} \tag{2.11.6}$$

Cluster cohesion is the mean distance between an observation $x^{(i)}$ and all other samples in that cluster. In contrast, the cluster separation is the average distance between the observation $x^{(i)}$ and all the samples in the nearest cluster. The silhouette coefficient values range from -1 to 1. From equation 2.11.6 we can observe that if the cluster cohesion and the cluster separation values are equal, the silhouette coefficient is zero. To get the perfect clustering, i.e. the silhouette coefficient of 1, $b^{(i)} >> a^{(i)}$, since the high value of $b^{(i)}$ indicates that the cluster samples are very dissimilar from other clusters' samples. Also, smaller $a^{(i)}$ means that a cluster sample is very similar to the other samples in its own cluster.

2.12 Artifical Neural Networks

Biological neurons are interconnected nerve cells in the brain which transmit and process electrical and chemical signals. McCullock and Pitts [41] describe these neurons as simple logic gates producing two binary outputs. As the signals arrive in the dendrites (Figure 2.12), they are integrated into the cell body. If the accumulated signals exceed a certain threshold, an output signal is generated, carried by the axon. A general model of ANN inspired by biological neurons is shown in Figure 2.13.

2.13 Deep Learning

Deep learning is a subfield of machine learning that incorporates many successive layers that learn meaningful representations from the data. Thus, the 'deep' in deep learning indicates the depth of these layers (i.e., the number of layers) in a model. Most machine

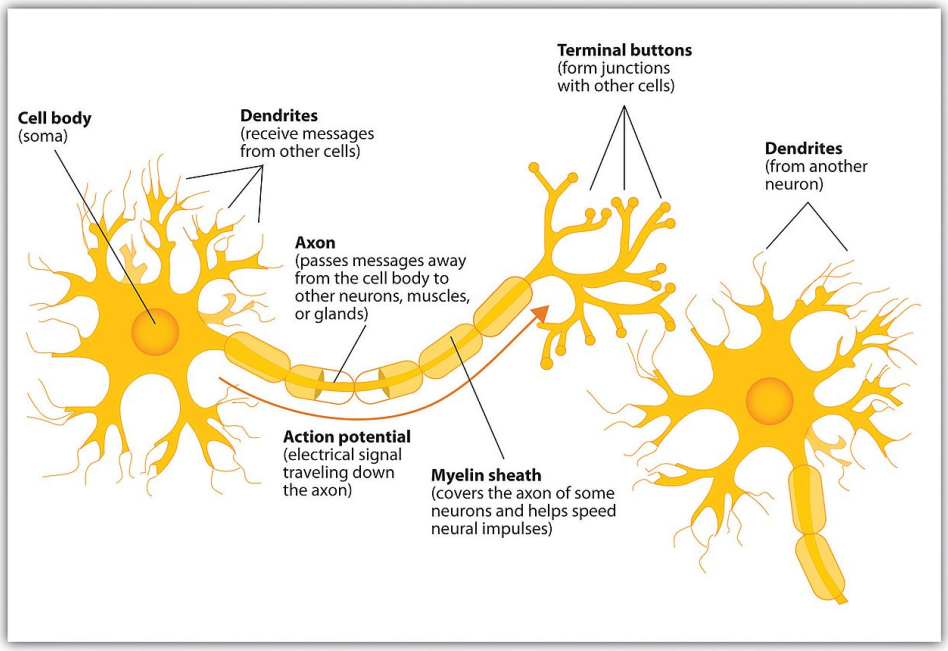


Figure 2.12: Illustration of a biological neuron [42]

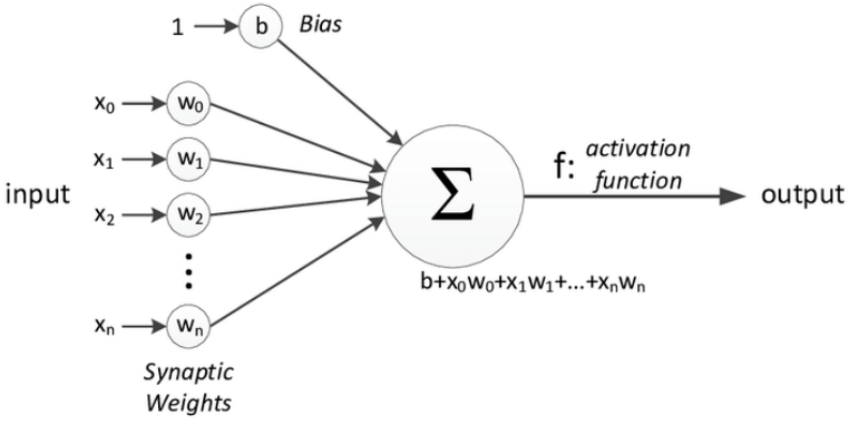


Figure 2.13: Illustration of an artificial neural network. A single layer neural network is also referred to as Perceptron.

learning methods focus on learning only one or two layers of representations of the data and are thus referred to as "shallow learning". Meanwhile, modern deep learning models use tens or even hundreds of successive layers stacked on top of each other to learn data representations [43].

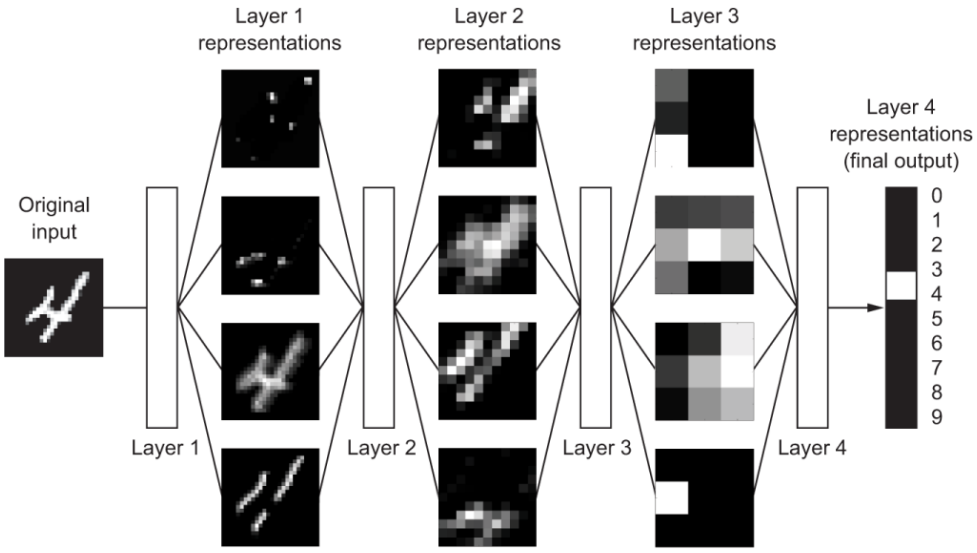


Figure 2.14: Digit classification model using deep learning [43]

Figure 2.14 shows the illustration of a digit classification model using deep learning. The network transforms the image such that the learning representation of the layers are very different from the original image. These representations are also very informative in comparison to the original image in predicting the final result. The layers can be interpreted as a series of filters in a distillation process, through which there is a flow of purified information [43]. Basically, learning means finding the optimal weight values for all the network layers such that the network maps the input to their targets as correctly as possible, where the weights are numeric values that guide the transformation of a layer.

The loss or the objective function of the network measures how far the model predictions are from the actual target value. The loss functions calculate a score value by computing the distance score between the true value and the predicted value. This score is the performance evaluation metric of the model. At the start, the weights of the net-

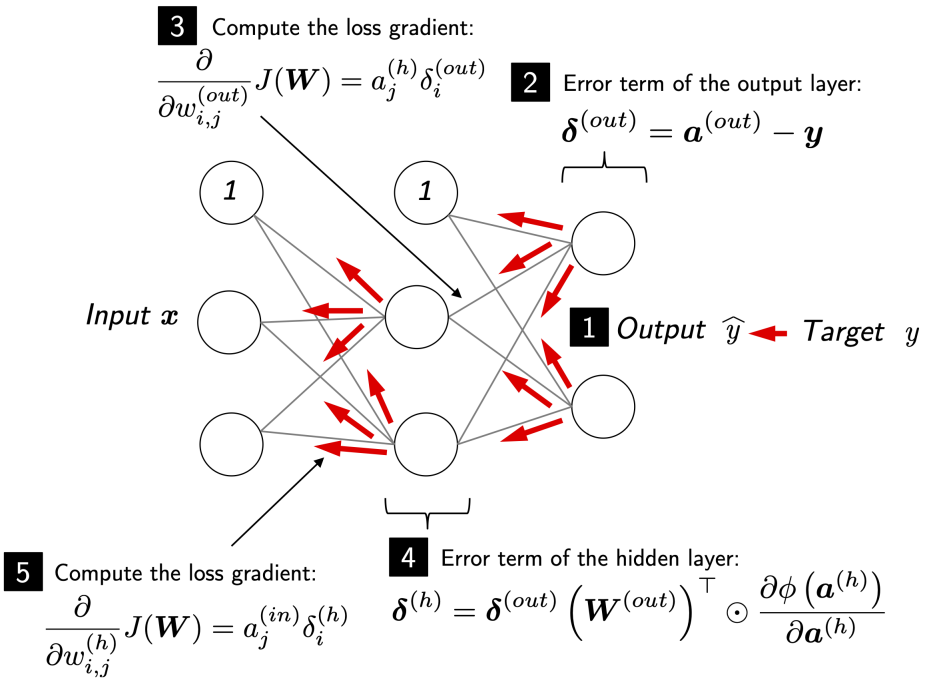


Figure 2.15: Illustration of the backpropagation algorithm [22]

works are set to random values; thus, the output of such models are naturally far off from the target values and likewise, the model loss is high. In a process referred to as backpropagation, Figure 2.15, an optimizer uses this score as a feedback signal to adjust the weight values with the goal of reducing the loss score [16]. Backpropagation is thus the backbone of neural networks. The gradient of the prediction error is computed as a function of the weights of the neurons, and the weights are adjusted to minimize this error. These output errors are propagated back to deduce the error in the hidden layers. The gradient of the error in the hidden layers are computed, and the layer weights are adjusted in the same manner [43].

For example, the Adam optimizer monitors the earlier partial derivatives and if the consecutive gradient points are in the same direction, it adds momentum leading to a faster training time and increasing the chances to avoid saddle points [44]. This process is

repeated for each of the samples many times, and is called a training loop. At the end of the training loop, a trained network with a minimal loss produces outputs that are close to the target values.

2.14 Feed-Forward Neural Network (FFN)

In an FFN, several artificial neurons are stacked on top of each other. Figure 2.16 illustrates a data flow in a neural network with the input layer, output layer, and n hidden layers between them. The circles represent the nodes, while the arrows indicate the direction of data flow. The input layer $x = [x_1, x_2, \ldots\ldots, x_m]$ consists of m neurons which are the unprocessed input data. As the m inputs are passed onto the next layer, it is multiplied by weights $w = [w_1, w_2, \ldots\ldots w_d]$, and a bias is added. An activation function is applied to the linear combination of the previous layer's node output. Hence, in a neural network, the output of the previous layers is an input to the following layer, and the output of the final layer is the predictions. In short, a neural network's learning goal is to find all the weight matrices such that the final predictions are as close as possible to the actual results. In an FFN, all the neurons in a layer are connected to all other neurons in the following layer and are referred to as dense layers. FFNs have demonstrated their ability to detect complex non-linear relationships between the dependent and independent variables along with the detection of possible interactions between the predictor variables.

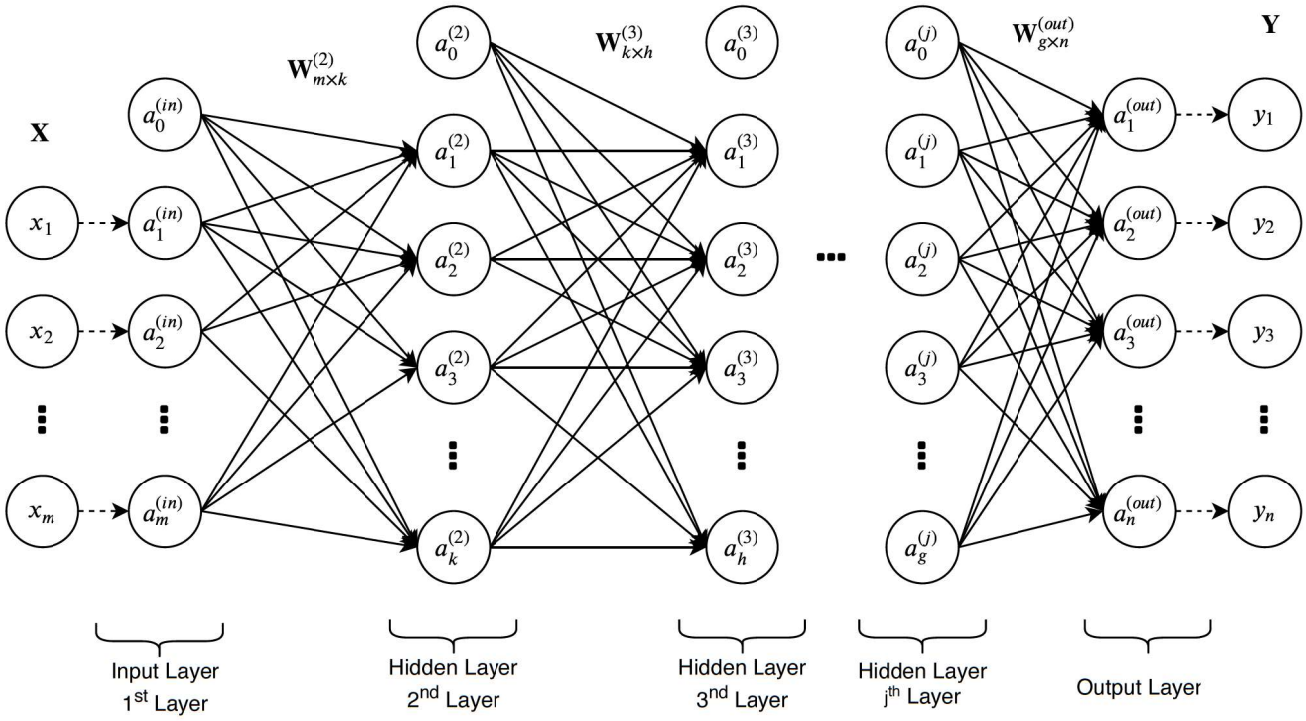


Figure 2.16: Illustration of a feed-forward neural network. *$a_i{}^{(j)}$ denotes the i^{th} activation unit of the j^{th} layer. $\boldsymbol{W}^{(j)}{}_{s \times t}$ denotes the weight matrix between the (j-1)th layer with s units and j^{th} layer with t units.*

2.15 Convolutional Neural Network (CNN)

CNN is another type of deep learning model that is primarily used in computer vision applications. Unlike FFN, it is characterized by a smaller number of parameters and consequently less training time. Another difference between FFN and CNN is that a convolutional network learns the local patterns while the former learns the data's global patterns [43]. Similar to working with images, time can be considered a spatial dimension as an image height or width. These kinds of 1D convolutional neural networks have considerably cheap computational costs.

Let **A** and **B** be tensors with shape (m_1, m_2,.....,m_n) and (n_1, n_2,.....,n_n) respectively. The convolution of **A** and **B** is defined as:

$$[A * B]_{i_1,\ldots,i_N} = \sum_{j_1=1}^{n_1} \cdots \sum_{j_N=1}^{n_N} A_{i_1-j_1,\ldots,i_N-j_N} B_{j_1,\ldots,j_N} \tag{2.15.1}$$

Here, **B** is also referred to as the convolution kernel.

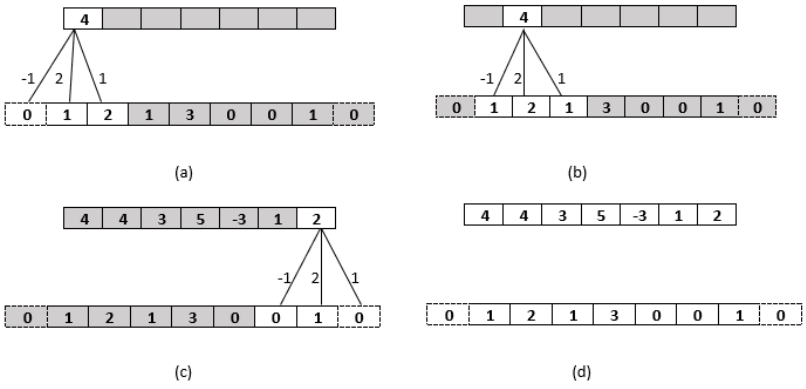


Figure 2.17: Illustration of 1D convolution operation with padding. The input vector is represented by the row of numbers in the bottom. The convolution vector B is (-1,2,1), and the top row represents the output of the convolution. (a) - (c) illustrates the computation of first, second and last elements of A*B while (d) presents the input-output vectors of the convolution.

However, from the definition above, we can see that it is not easy to deal with the

boundary pixels. The filter begins at the left of the image with the filter's left-hand side positioned on the image's far left pixels. The filter is then moved across the image, a single column at a time until the filter covers the far-right pixels of the image. Thus, to ensure that each pixel in an image takes the center position of the filter and maintains the output feature map's size, padding is used. Padding expands an input tensor in all directions by creating new tensor elements whose values are set to zero.

The 1D convolutional network extracts local 1D patches from the input sequences. This ability to extract features from local input patches increases the data efficiency representation modularity. These 1D layers can recognize the local patterns anywhere in the sequence as the same input transformation is done in every patch, making them translation invariant [43]. Filters help in extracting these local patterns. Filters are a set of vectors that are trainable and are thus adjusted by the gradient descend. The filter features such as the width and strides (steps) are predefined. Each of these filters also has another trainable parameter called a bias. The output produced by the filters is referred to as "feature maps" [45]. The number of spatial features extracted by the filters depends on the number of features, whereas the filter's width guides the range from which the spatial dependencies are extracted.

Pooling (Figure 2.18) is a technique used to downsample the image tensors in CNN spatially. Likewise, in 1D convolution, pooling can extract 1D patches from inputs and yield the maximum value (max pooling) and the average value (average pooling) as the output [44]. It helps in reducing the lengths of 1D inputs. There is a sparse connection between several layers of a CNN, such as convolution layers, pooling layers, and even dense layers. The dense layer is commonly used in the final layer to map spatial features into a set of output nodes.

CNN is also a feed-forward neural network since the input parameters are propagated throughout the layers. Therefore, to avoid confusion, only FFN without convolutions are referred to as FFN in this thesis.

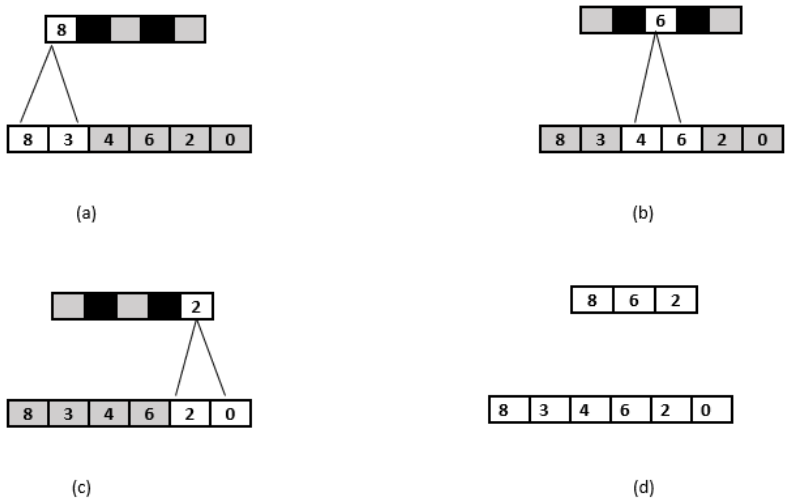


Figure 2.18: Illustration of 1D pooling operation. A max pooling operation is performed, where the output is the maximum value in the input image. (a) - (c) shows the output and the computational process of max pooling operation while (d) shows the final input and output from the max pooling operation.

2.16 Activation Functions

Activation functions like ReLU, sigmoid and tanh enable the layers to learn non-linear dependencies [46]. A drawback of the linear activation functions is that they shrink the hypothesis space of the layers. No matter how many layers are added, if the layer activations are linear, the hypothesis space cannot be broadened, since a stack of linear layers always learns linear representations.

$$\phi_{sigmoid}(x) = \frac{1}{1 + exp(-x)}. \tag{2.16.1}$$

$$\phi_{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.16.2}$$

$$\phi_{softmax}(\mathbf{x})_i = \frac{exp(x_i)}{\sum_j exp(x_j)}. \tag{2.16.3}$$

$$\phi_{ReLU}(x) = max(0, x). \tag{2.16.4}$$

ReLU, sigmoid and tanh aid in non-linear learning but have different activation ranges. The sigmoid and ReLU limit the activation to a positive value, whereas the tanh provides a broader activation range from -1 to 1. ReLU is a piecewise linear function that sets all negative values to zero. ReLU is the most widely used activation function in the hidden layers in recent years [47], because it avoids the vanishing and exploding gradient problem since its derivative is zero for negative inputs and one for positive inputs [48][49]. In addition, it contributes to larger update steps and leads to efficient convergence. The choice of activation in the output layer of a neural network is dependent on the specific problem type. For example, softmax is used in multiclass single-label classification, whereas a linear activation function is used in most of the regression problems. Unlike other activation functions, the softmax takes vectors as inputs, as a results it's output sums up to one. The nature of these activation functions is illustrated in Figure 2.19.

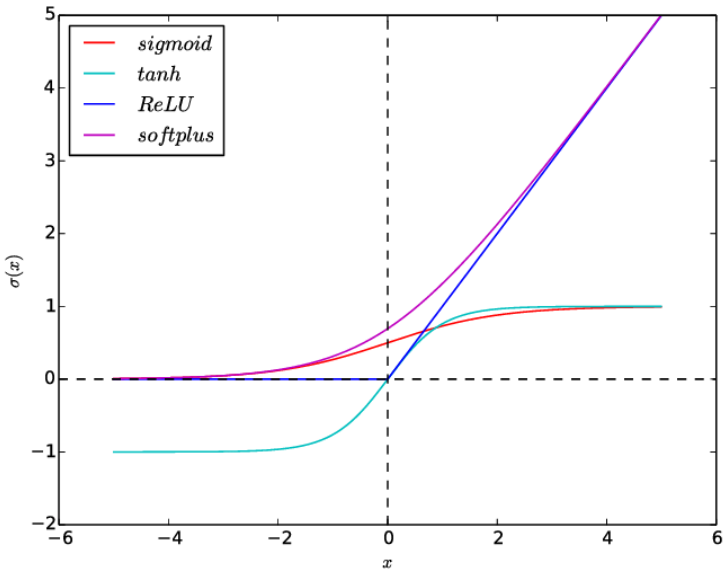


Figure 2.19: Illustration of various activation functions

2.17 Regularization in Neural Networks

The primary issue in any machine learning algorithms is the balance between optimization and generalization. The process of getting the best out of a model during training is called optimization. Generalization refers to how well the model performs on data which it has never seen. The goal of any machine learning model is to gain better generalization, but we cannot control generalization. Instead, it is only possible to adjust the model based on the training data. Trying to generalize to samples outside of the space of the training data is referred to as extrapolation, and often leads to unreliable predictions.

At the start of the model training, optimization and generalization are correlated. This means that lower loss on training data yields a lower loss on the test data. In this context, the model is said to be underfitting, i.e., the model has not yet learnt all the relevant patterns in the training data. As the number of training iterations increase, at a certain point, the generalization stops improving. Following this, the validation metric value flattens and starts falling. In this case, the model is said to be overfitting, i.e., it has started to focus on the patterns that are only specific to the training data but are ambiguous when it comes to unseen data.

The best solution for preventing a model from overfitting is getting more data. Usually, a model that is trained on more data generalizes better provided that the new data is sufficiently different from the original data [16]. Another approach is called regularization, i.e., adding constraints to the model, preventing it from memorizing irrelevant patterns found in the data. Weight regularization is a technique that penalizes higher weights and forces the weights to be smaller. The weight regularization is achieved by adding a cost to the loss function. In L1 regularization, the added cost is directly proportional to the absolute value of the weight coefficient. In contrast, in L2 regularization [50], it is proportional to the square of the value of the weight coefficient.

Another way of reducing overfitting is to reduce the model's capacity (number of

learnable parameters of the model). This will reduce the memorization capacity of the model, and make it generalize better [43]. On the other hand, if the model's capacity is low, the models tend to underfit. Therefore, it is necessary to find the sweet spot between too much capacity and not enough capacity.

2.17.1 Dropout

Dropout is the most popular regularization technique used in neural networks. It was developed by Geoff Hinton and his students at the University of Toronto [16]. The dropout works by setting zero values to random output features during training. The fraction of the features that is converted to zero is called the dropout rate. None of the units are dropped during the test, but the output values from the layer are scaled down by the dropout rate. This ensures that all the units are available at testing as compared to training. Hence, the dropout reduces the model's capacity during training and forces a redundancy between various features learnt in various hidden units, leading to increased robustness [51].

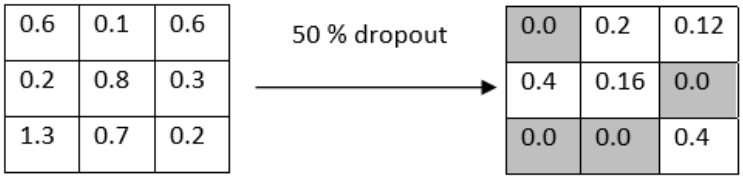


Figure 2.20: Illustration of the dropout method

2.18 Callbacks

A callback is an object (class instance) passed to a model in the fit method. The model calls this object at various steps during training. The Keras/TensorFlow [52] callbacks module has several built-in callbacks.

2.18.1 Earlystopping

This callback interrupts training once the validation or the target metric stops improving after several epochs. It stops training as soon as the model starts overfitting, eliminating the need to retrain it to the optimal number of epochs [53].

2.18.2 Reduce Learning Rate on Plateau

This callback reduces the learning rate when the validation loss stops improving. Increasing and decreasing the learning rate is an effective technique to avoid local minima [43].

2.19 Batch Normalization

Normalization brings all the features to the same scale. Normalization centers the data on zero by subtracting the mean and dividing by its standard deviation. The data after normalization follows a standard normal distribution, with a mean value of zero and a standard deviation of one.

It is common practice to normalize the data before feeding it to the fitting method. There is no guarantee that a neural networks' transformations (its output) are normalized. The batch normalization is a layer that normalizes the output activations from a layer before forwarding it to the next layer. This assists in developing deeper networks by aiding gradient propagation [48][47].

An introduction to the basic mechanisms of the heart physiology are presented in Appendix C.

2.20 Heart Muscle Cells

Cardiac muscles are striated muscles since they have alternating dark and light bands. They are also involuntary as the autonomous nervous system innervates them. A cardiac muscle cell consists of numerous myofibrils that extend throughout the length of the muscle fiber. Myofibrils are contractile elements that constitute up to 80% of the total volume of the muscle fiber. Each of the myofibrils constitutes a thick and a thin filament made up of myosin and actin proteins respectively. A sarcomere is the functional unit of cardiac muscle, i.e., the smallest muscle fiber unit that can contract. Small bridge-like structures can be spotted on the thick filaments stretching toward the thin filaments in the overlap zone at high magnification. They're called cross-bridges, and are responsible for the movement and the force generated during contraction. Each of the cross-bridges has two sites necessary for the contraction process: an actin-binding site and a myosin ATPase (ATP-splitting). Actin is the dominant protein in the thin filaments and consists of a binding site to bind with myosin. This binding between an actin molecule and myosin at a cross-bridge results in contraction of heart muscle cells [54].

2.21 Generation of a Cardiac Action Potential

A brief illustration of the process of generation of cardiac action potential is presented in Figure 2.21. The Phase 0 follows a depolarization process allowing the sodium ions to move rapidly into the cell. At Phase 1, the sodium channels are closed. Phase 2 is also known as a plateau phase where the potassium ions rapidly move out of the cell while the calcium ions slowly enter the cell. During the rapid repolarization (Phase 3) the calcium channels close and the potassium ions still continue to move out of the cell rapidly. In

the Phase 4, the sarcolemma becomes impermeable to sodium ions and leaky potassium channels open out modulating the cell responsiveness and modularity.

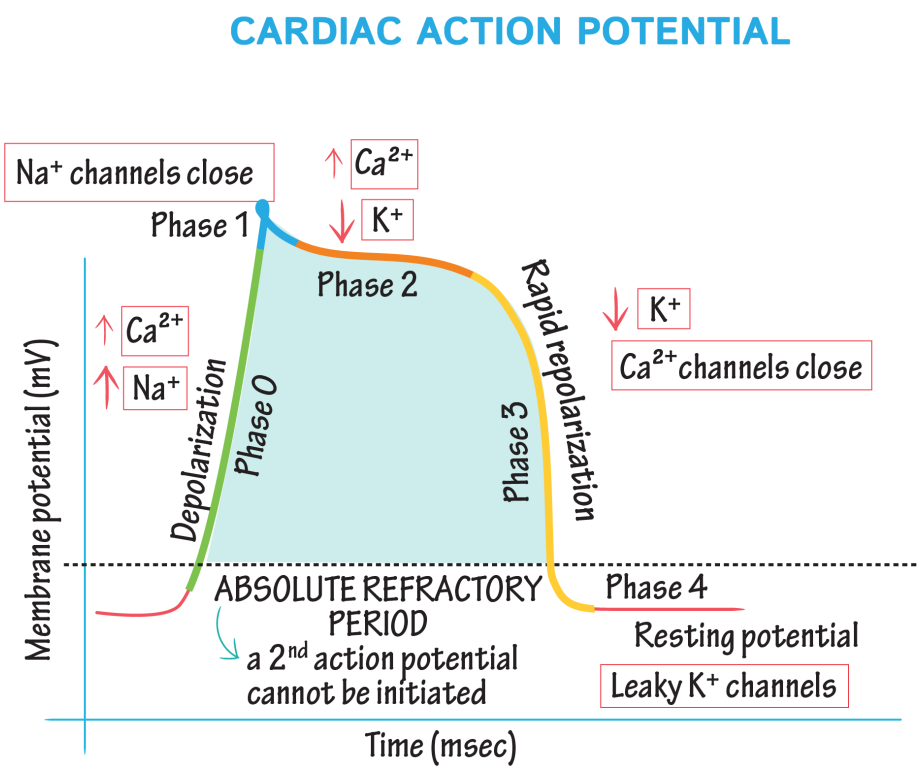


Figure 2.21: Illustration of the generation of cardiac action potential. Figure adapted from [55]

2.22 Initiation of contraction by an Action Potential (AP)

The generated AP in the cardiac contractile cells travels down to the transverse (T)-tubules, where the L-type Ca^2+ channels are located. The Ca^{2+} ions from the extracellular fluid (ECF) travel along these channels and diffuse into the cytosol triggering the opening of ryanodine Ca^{2+} release channels in the lateral sacs of the sarcoplasmic reticulum (SR). This process is called Ca-induced Ca-release and releases a massive amount of Ca^2+ from SR through ryanodine Ca^{2+} release channels into the cytosol. This release of Ca^{2+} ions is known as a Ca^2+ spark. The increased concentration of cytosolic Ca^{2+} initiates the contractile process. The cytosolic Ca^{2+} combines with the troponin-tropomyosin complex in thin filaments, which are pulled aside to form Ca^{2+} cross-bridge cycling and contraction between the thick and thin filaments. The excess Ca^{2+} in the cytosol is removed by processes such as the $Na^+ - Ca^{2+}$ exchanger in the plasma membrane and the sarcoendoplasmic reticulum calcium transport ATPase (SERCA) pump in the SR. This reinstates the blocking action of troponin and tropomyosin, causing the heart muscle cells to relax [54]. This process is also illustrated graphically in Figure 2.22.

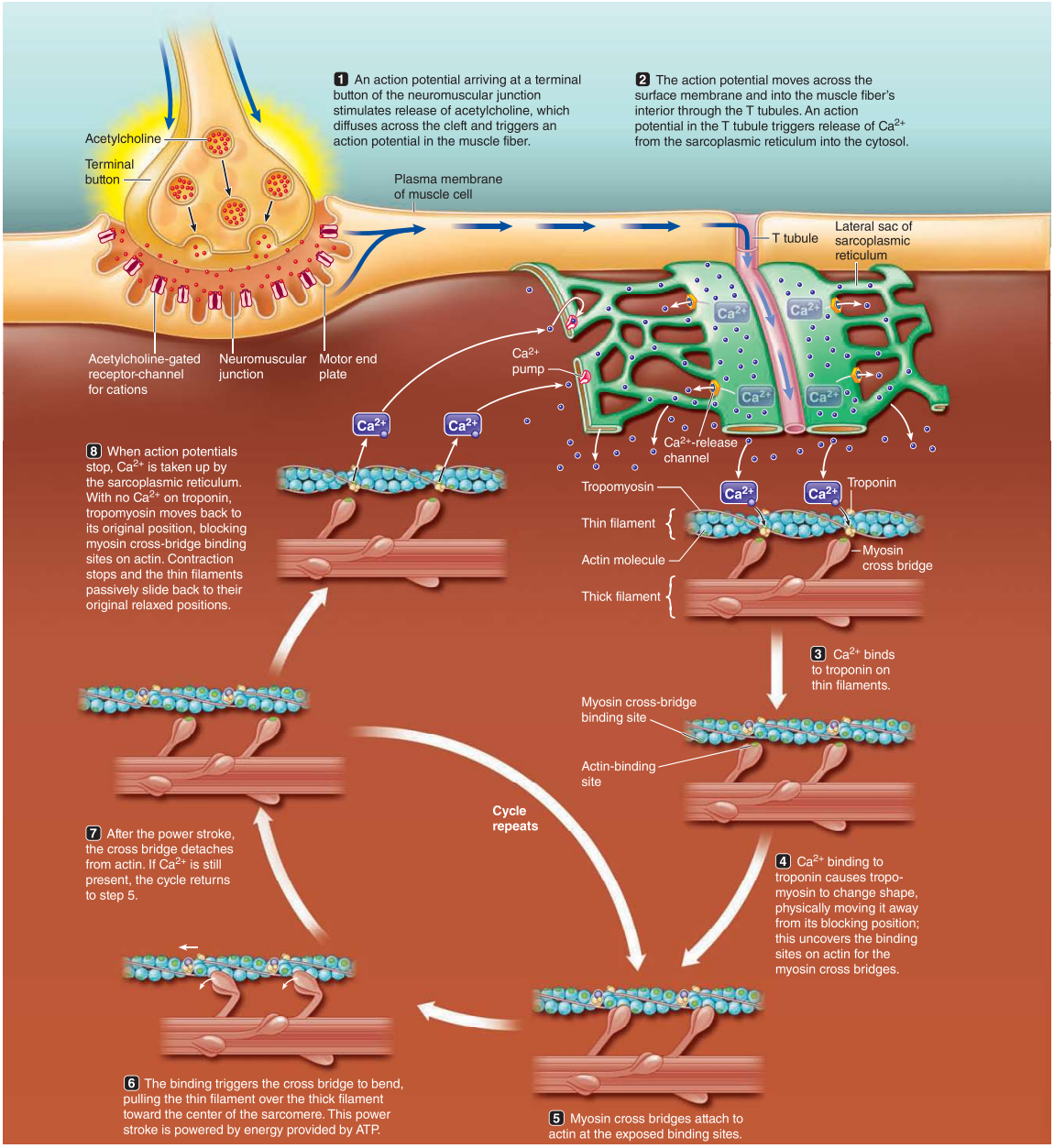


Figure 2.22: Illustration of excitation-contraction coupling and muscle relaxation. The steps 1-7 describe events coupling the neurotransmitter release and consequent electrical excitation of the muscle cell with muscle contraction. The cross bridge cycle returns to step 5 if the Ca^{2+} ions are still present in step 7. If the Ca^{2+} ions are no longer present, relaxation occurs as a result of step 8 [54]. Figure adapted from the book Human Physilogy [54][9 th Edition, Page 260] **Reprinted with Permission © Cengage Learning**

The following sections on model descriptions are based on [56],[57], [58],and[8] respectively.

2.23 The Pandit Model

Scientific works have revealed that there is a difference in AP waveforms in epicardial and endocardial cells in the left ventricle (LV) of the mammalian heart. This difference is termed as 'transmural heterogeneity', and occurs in feline, rabbits, mouse, rat, and humans. The electrophysiological properties in epicardial myocytes are characterized by a shorter AP duration (APD) than in endocardial myocytes.

Adult rats have been extensively used in LV studies of the electrical heterogeneity, not only under normal conditions but also in pathophysiological conditions such as diabetes, thyroid dysfunction, and myocardial infraction. It is observed that endocardial myocytes have longer APD in healthy adult rats making it an important inotropic (causing changes in the strength of contraction) variable. This APD has a more observable rate-dependent effect and higher peak overshoot in comparison to epicardial APD. The ionic mechanisms and their interactions are important to understand the electrical heterogeneity in adult rat LV.

Scientific literatures suggest that Ca^{2+} independent transient outward K^+ current ($\mathrm{I_t}$) is a key factor contributing to the differences in the endocardial and epicardial myocytes APD. The endocardial cells have a smaller density of ($\mathrm{I_t}$) and a slower recovery from inactivation kinetics than epicardial cells. Also, the endocardium in LV is reported to have a higher density of transmural gradient of Na^+ current ($\mathrm{I_{Na}}$). Transmural gradient is the differenece between the intracavitary pressure and the extracavitary pressure. The extracavitary pressure approximates to zero, thus the filling pressure usually equals the ventricular diastolic mean pressure. Pandit *et.al.* developed a mathematical model to analyze these differences in endocardial and epicardial cells by reconstructing the APs under normal conditions.

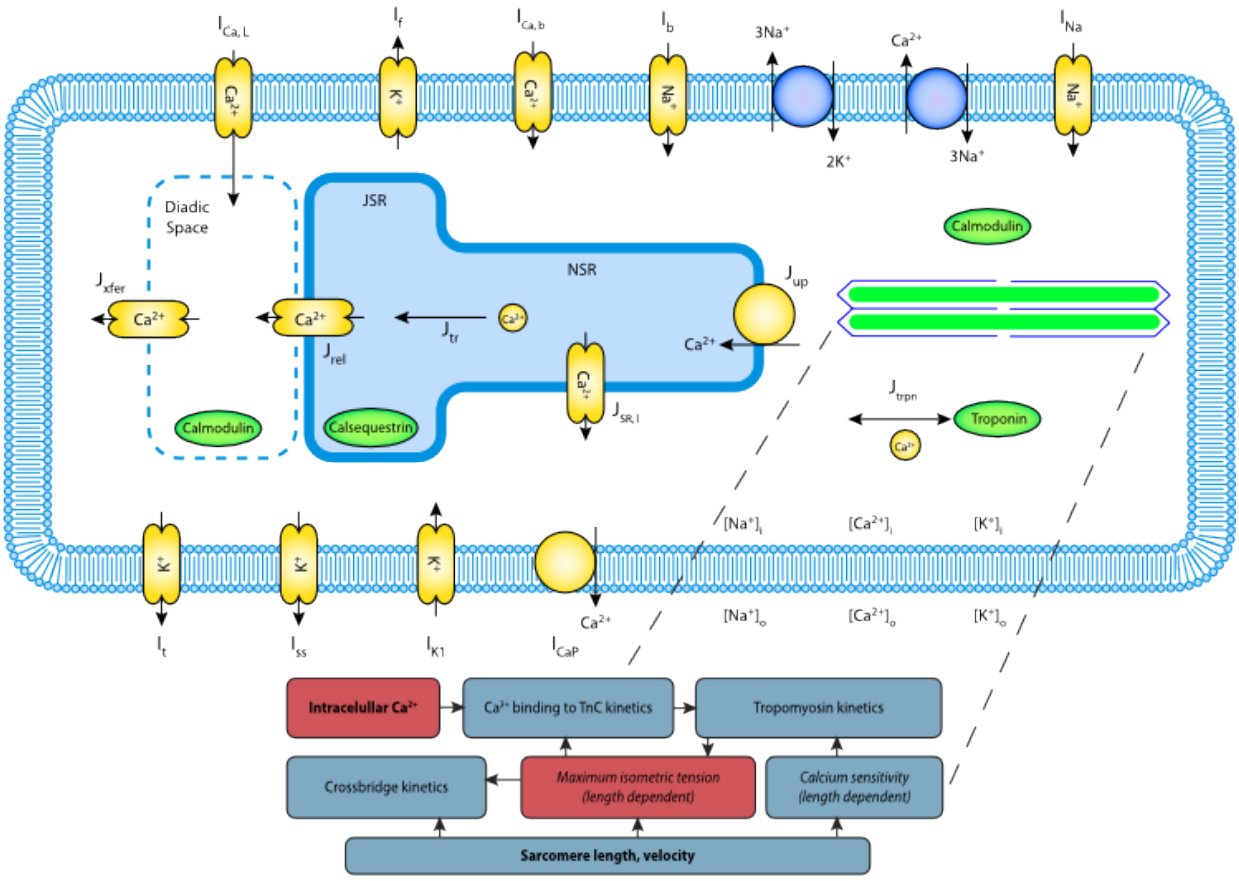


Figure 2.23: A schematic diagram of the CellML Model [59]

The mathematical formulation of the epicardial and endocardial cells of the rat LV is based on the Hodgkin Huxley model (1952). The electrical equivalent circuit of the Pandit model describing pumps, ion channels in sarcolemma and the Na^+ - Ca^{2+} exchanger in adult rat LV is shown in Figure 2.24 **A**. A fluid compartment model, Figure 2.24 **B**, is coupled together with the electrical equivalent circuit. The fluid compartment describes the changes in Na^+, Ca^{2+} , K^+ ions in the myoplasm, and the Ca^{2+} ions in the sarcoplasmic reticulum.

2.24 The Hinch Model

Most local control Ca^{2+} release models utilize stochastic simulation between L-type Ca^{2+} channels (LCCs) and ryanodine-sensitive Ca^{2+} release channels (RyRs). However, they are characterized by a high computational cost even in single-cell simulations. The Hinch model provides a simplified approach for modelling the local control of Ca^{2+} induced-calcium release (CICR) in cardiac ventricular myocytes.

The formulated model referred to as the LCC-RyR gating model consists of a low dimensional system of ODEs since it incorporates only the essential biophysical features from electromechanical (EC) coupling. This gives a system of model equations that can be simplified using timescale-based approximations. For example, Ca^{2+} in dyadic space equilibrates rapidly relative to the gating dynamics of LCCs and RyRs. Also, the model utilizes the Markov model in describing the cooperative behavior of LCCs and RyRs. The transition probabilities between the interacting states are solely a function of global variables enabling the calculation of collective calcium release units (CaRUs) behavior through ODEs. Despite its simplicity, the model accurately describes the main properties of CICR, which include graded release and voltage dependence of EC coupling gain. The EC gain is computed as peak SR Ca^{2+} release flux divided by the peak flux of Ca^{2+} across the cell membrane and quantifies the amplification provided by CICR.

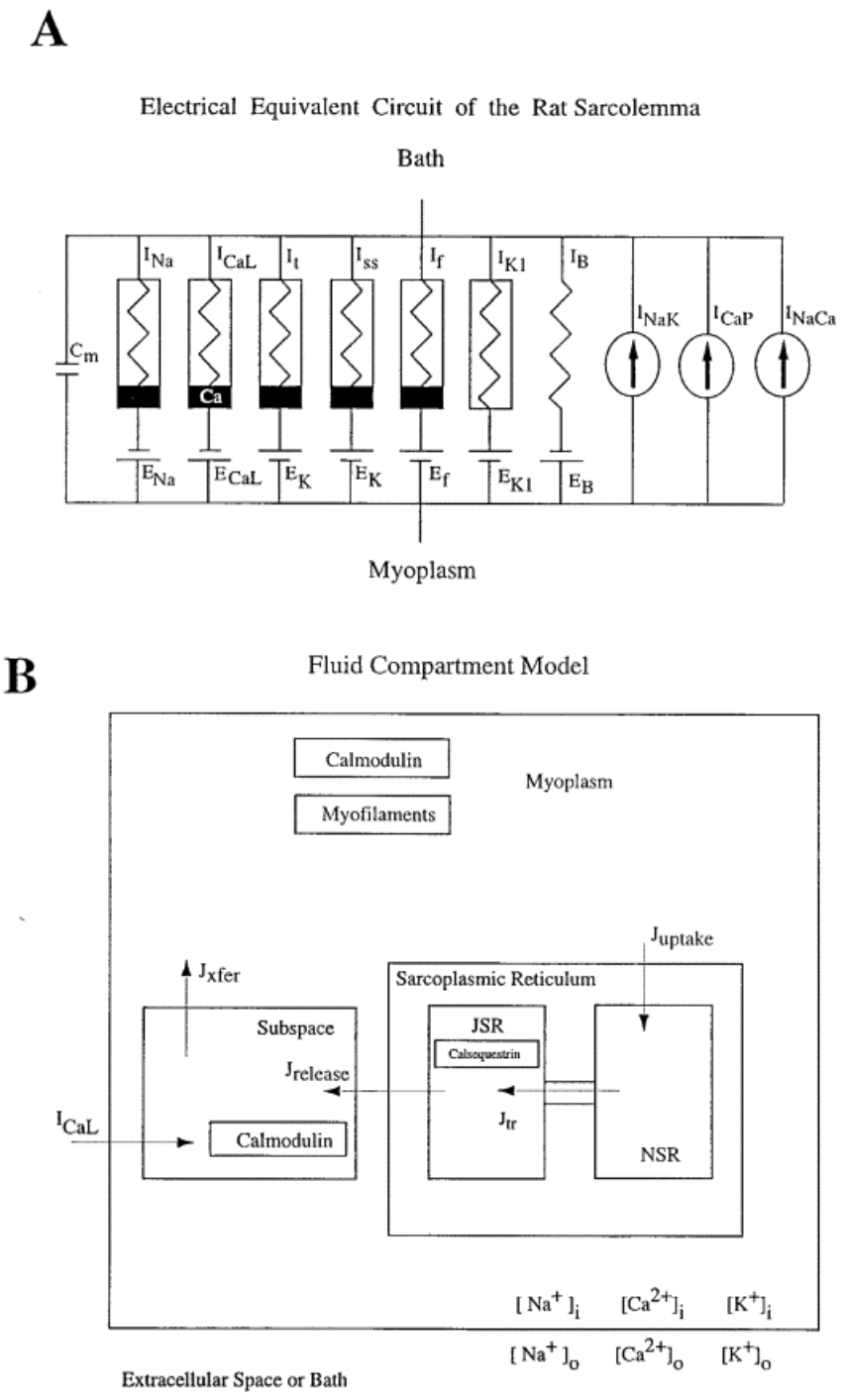


Figure 2.24: Electrical equivalent circuit (A) and fluid compartment model (B) of the rat cardiac cell [56]

2.25 The Niederer Model

This model formulates the phenomenon of cardiac myocyte relaxation. The relaxation and tension development at each heartbeat is governed by cellular mechanisms and takes place in sarcomere spatial scale. The interdigitated protein filaments of actin and myosin make up the sarcomere. The crossbridges that perturbate from myosin bind to actin. When these crossbridges undergo a conformational change, the bound crossbridges start to pull the filaments in the opposite direction, generating tension. This procedure is regulated by Ca^{2+} ions, which are locally free and by the intrinsic properties of the sarcomere.

A rise in the local $[Ca^{2+}]_{\mathrm{i}}$ causes contraction. The binding of Ca^{2+} ions and troponin C (TnC) leads to the movement of tropomyosin. This in turn exposes the actin binding sites, allowing the actin binding of the crossbridges whilst generating tension. The events such as the unbinding of Ca^{2+} and TnC, obstruction of the binding sites by tropomyosin, separation of crossbridges, and the drop in the tension value back to zero is followed after the $[Ca^{2+}]_{\mathrm{i}}$ is removed. A lot of research has been done on the process of tension generation , but the steps controlling the relaxation are not extensively studied.

Relaxation is measured as the time at which the peak tension value decays by fifty percent. The $[Ca^{2+}]_{\mathrm{i}}$ transient and intrinsic properties of the myofilaments are the determinants of relaxation. Relaxation is affected when the $[Ca^{2+}]_{\mathrm{i}}$ is altered or the simulation frequency is increased. The intrinsic properties of myofilaments that affect the relaxation are sarcomere length, tension dependent binding of Ca^{2+} to TnC, inhomogeneous sarcomere shortening, phosphorylation of troponin I, and crossbridges inhibiting tropomyosin returning to its resting state. Each of the parameters in the simulation of the Niederer et.al model are adapted from a number of articles from the literature and experiments.

The model of the active contraction is based on the structure suggested by Hunter et.al., illustrated in Figure 2.25. The inputs to the model are sarcomere length, velocity and intracellular $[Ca^{2+}]_{\mathrm{i}}$. Crossbridge kinetics, tropomyosin kinetics and Ca^{2+}binding

to TnC kinetics are the processes described by the differential equations. The model utilizes steady state and transient experimental data in describing the Ca^{2+} binding to TnC. Likewise, the model uses light-activated Ca^{2+} chelator experiments and Force - Ca^{2+} curves to expose the actin binding sites because of the movement of tropomyosin. The active tension is calculated as a product of available actin sites, maximum isomeric tension, and a sarcomere-velocity dependent scalar. The maximum isomeric tension is described by a linear function of sarcomere length and is produced when the length of muscle's sarcomeres are on plateau of the length-tension curve. The function parameters are maximum velocity, rapid length step and sinusoidal perturbation experimental results.

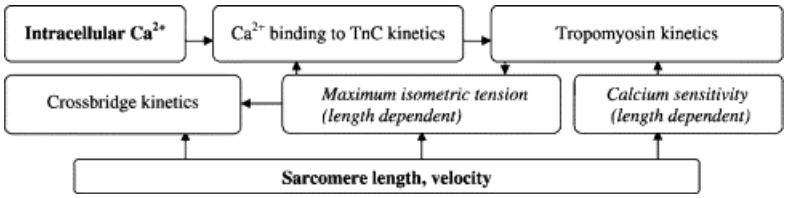


Figure 2.25: Flow diagram illustrating the relationships between the active contraction [58]

2.26 The Pandit-Hinch-Niederer (PHN) Model

Most of the scientific works are developed on the basis of previous experiments and results in the same field. In the computational field, the same is achieved by the use of accepted standards which describe and simulate the models. The cardiac model used for metamodelling in this paper is a combination of an electrophysiological model, a cellular calcium dynamics model, and a mechanics model, which together formulate the Pandit-Hinch-Niederer (PHN) model. Sarcolemmal K^+ and Na^+ currents are described by the Pandit endocardial model, the description of Ca^{2+} fluxes is derived from the Hinch model, while the binding mechanism of Ca^{2+} to troponin C and the tension development is related to the Niederer model.

The XML based programming language (Cell ML) facilitates the reuse and combina-

tion of these models. A primary issue in creating ensemble models is model incompatibility. PHN describes three types of incompatibility issues, namely unit inconsistencies, structural inconsistencies and parameter inconsistencies.

Unit inconsistency is one of the simplest forms of inconsistencies that can be easily solved. It refers to the variables that vary in the dimension but have equivalent units. For example; the current obtained from the Pandit model is in nA whereas the PHN model uses μA. This issue is automatically resolved by the integration software PCEnv by converting the dimensions to equivalent units.

Structural inconsistencies reflect the various ways in which bilology is integrated in mathematical equations. For example, the ion fluxes and the whole-cell current both describe the movement of ions across the cell membrane but are not in equivalent units and cannot be converted into one another. This conversion requires the valance of the species (i.e. Ca^{2+}) and the volume of myocytes. The conversion equation can be described as :

$$I = j \times z \times F \times vol_{\text{myo}} \tag{2.26.1}$$

where the current (*'I'*) is in microamperes, vol_{myo} is in microlitres, F is the Faradays constant, the flux of calcium ions *(j)* in millimolar per milliseconds, and z is the valence for Ca^{2+} ions which is 2. Even so, the dependence of myocyte volume in this conversion leads to parameter inconsistency.

The parameter inconsistencies are caused by the differences in the mathematical equations used in several models. Although these equations may describe the same biological process, the units of the measurement may be different. From equation 2.26.1 it is evident that the conversion of fluxes to current is dependent on cell myoplasm volume. However, the Pandit model uses *picolitre* and the Hinch model uses *micrometrecube* as the unit to measure cell volume. Thus, it was necessary to choose a uniform value of the cell volume in the final PHN model. In comparison to other cellular ions in ventricular myocytes,

Ca^{2+} has significantly lower ion concentration. Consequently, it has a higher sensitivity to the volume of myoplasm, creating a strong dependence between the results of Ca^{2+} dynamics and the cellular dimensions in the Hinch model. Hence, the cell dimensions from the Hinch model was used as it accurately measured the intracellular Ca^{2+} concentration and this in turn regulated both cellular electrophysiology and contraction. Also, the membrane currents described in the Pandit model are whole-cell current and are independent of cell volume. This further aided the integration of these models.

The full set of modified equations of the PHN model are listed in the Appendix B.

3 Methods

3.1 Data generation

The cardiac membrane potentials analysed in this paper were simulated using the Pandit-Hinch-Niederer model. The python implementation of the model was obtained from the Physiome Project repository `https://models.physiomeproject.org`. The original implementation was modified to allow for varying the parameter values. 76 parameters (set as 'constants' in the original implementation) were varied by $\pm 50\%$ from their default values, using a Latin Hypercube sampling procedure. This resulted in 10,000 combinations of values of those 76 parameters. See Appendix A for a complete list of the varied parameters and their default values. However, six of the parameters were left unchanged and are described in Table 1. The membrane potential was evaluated over time for each set of parameter values to analyse the variance in the model output, resulting from the variation in the input parameters. Each of the 10,000 simulations was run for 250ms with a resolution of 1ms, resulting in a recording of the cardiac membrane potential over 250 timesteps. These simulations ran for 43 hours and 33 minutes on a computer with a processor powered by Intel® Core™ i5-8250U CPU@1.60GHz (8CPU) $\sim$ 1.8GHz, and a memory of 8GB.

Table 1: Default values in simulations of PHN Model

Parameter Description	Value
R in component membrane (mJ/mole K)	$-8314.5\ mJ/moleK$
T in component membrane (kelvin)	$295\ K$
F in component membrane (C/mole)	$96487\ C/mole$
Stimulation period in component membrane (ms)	$1e^{3}\ ms$
Stimulation duration in component membrane (ms)	$10\ ms$
Stimulation amplitude in component membrane (uA)	$-0.6e^{-3}\ ms$

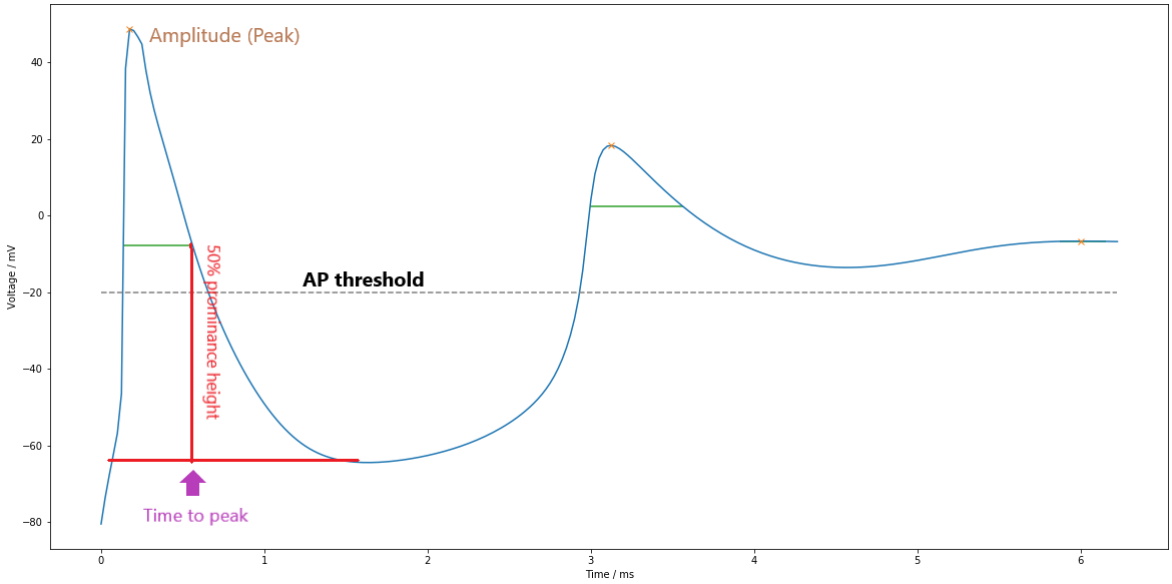


Figure 3.1: Illustration of the calculation of the aggregated phenotypes. The threshold for the action potential is set to -20mV while the width of an AP is calculated at 50% prominence height. Likewise, *Time to peak* and *Amplitude* are calculated at the maximum AP.

3.2 Classical Metamodelling

3.2.1 Classical Metamodelling of the time series data

Classical Metamodelling of the time series data using HC-PLSR

The parameter combinations that generated the AP were collected and further divided into a training (70%) and a testing set (30%). Feature engineering was used to increase the feature space by including cross-interaction terms, second order terms and sinus and cosine terms. A widely held belief is that the introduction of sinus and cosine terms might benefit the classical metamodelling problem taking care of abrupt non-linearities in the data [60]. Including cross-interaction terms may help identify the interactions between the main effects.

The training data was standardized to zero mean and unit standard deviation, forcing the features to form a standard normal distribution. An initial PLSR model, referred to as the global PLSR model, was calibrated with the training set, using ten-fold cross-validation. The PCs that explained one percent or more of the total validated $\mathbf{Y}$ variance were selected in the global PLSR model, while the rest of the PCs were considered as noise and therefore omitted. In HC-PLSR, the $\mathbf{X}$ scores obtained from the global PLSR model are utilized for clustering. The k-means clustering algorithm was used and the optimal number of clusters was determined by the elbow method. Depending on the optimal number of clusters, the equivalent number of PLSR models, referred to as local PLSR models, were fitted from the respective cluster samples with the same settings as that of the global model.

Prior to prediction with the test set, the cluster memberships of the test set samples must be determined by classification. For this purpose, classification models were developed using the training set. Several classification algorithms were used to train a classification model and examine the classification accuracy with the global $\mathbf{X}$ scores as the training data. These classification algorithms are listed below:

- Random Forests
- Support Vector Machines
- Logistic Regression
- K-Nearest Neighbors

The test set was standardized using the mean and the standard deviation from the training set. This resulting scaled test set was projected into the global PLSR model to obtain the **X** scores of the test set. These projected **X** score matrices were used to predict the cluster labels for the test samples, using the best classification model obtained. The final **Y**-prediction for the test set samples was based on the prediction from the local PLSR model that the sample was classified into.

Classical Metamodelling of the time series data using deep learning

The classical metamodelling of the generated time series AP data was done using FFN and CNN. RENT and SBS were used as feature selection techniques to identify and select the important features for the deep learning models. The total dataset was divided into a training (70%) set and a test set (30%). The training set was further divided into a validation set (20%) and an actual training set (80%). The data was standardized as described above.

CNNs are also powerful tools as they can look for spatial dependencies in the data, unlike FFNs. Adam was used as the default optimizer as it handles sparse gradients on noisy problems. The number of epochs, batch size, and the learning rate were tuned using the validation set. The coefficient of determination (R^2) was used as the performance evaluation metric, while the mean squared error was used as the loss function. To avoid overfitting, dropout was used in combination with early stopping and reducing the learning rate on the plateau. Likewise, the output from each of the layers was normalized using batch normalization, enabling the weights to take smaller values. The final performance of the model was evaluated on the test set that the model had never seen before.

The model architecture for the classical metamodelling of the PHN model using FFN is described in Appendix F Figure F.1. All hidden layers in FFN used ReLU as the

activation function and utilized Adam as an optimizer. A linear activation function was used in the output layer of all the metamodels, handling them as a regression problem. The FFN comprised of three fully connected dense layers, each with 232 neurons.

Likewise, the model architecture for the classical metamodelling of the PHN model using CNN is described in Appendix F Figure F.2. The network consists of four hidden layers in which the first three are 1D convolution layers while the last one is a dense layer with 500 neurons. The convolutional layers comprised of 128 filters and a kernel size of 10. Max pooling, average pooling, dropout, batch normalization and flatten were also introduced in between the layers. The activation function and the optimizer in both the models were the same as that of the FFN model.

3.2.2 Classical Metamodelling of the aggregated phenotypes

The classical metamodelling of the aggregated phenotypes was carried out using HC-PLSR, FFN and CNN. The **Y** matrix consisted of extracted features such as "time to peak", "width of the first AP", and "amplitude of the first AP" while all of the 76 parameters were used in the **X** matrix for a fair comparison between the model performances. The exact same methods and constraints were applied in this metamodelling process while using HC-PLSR as described in the section 3.2.1.

Likewise, for FFN, the choice of activation function, optimizer, loss function, and the evaluation metric were the same as that of the classical metamodelling of time series data using FFN. Batchnormalization, dropout and callabcks were also implemented to reduce overfitting. The architecture of the FFN model used for metamodelling of the aggregated phenotypes is shown in Appendix F Figure F.3. It consists of a single dense layer with 15 neurons.

Similarly, the model architecture for the metamodelling of aggregated phenotypes using CNN is shown in Appendix F Figure F.4. It consists of a single 1D convolutional layer, and a dense layer with four neurons. The convolutional layer utilized two filters

and a kernel size of two. The choice of activation function, loss function, optimizer and the intermediate layers were identical to the model used in the classical metamodelling of the time series data using CNN.

3.3 Inverse Metamodelling

3.3.1 Inverse Metamodelling of the time series data

The inverse metamodelling of the time series data was carried out using HCPLSR, FFN and CNN. The **Y** matrix consisted of the 76 parameters that were varied, while the **X** matrix was the time series data. The inverse metamodelling process using HC-PLSR followed the same methods and constraints as introduced in the classical metamodelling except for the cross-interaction terms, which were not included here.

The inverse metamodelling of the time series data using FFN followed the similar setup to that introduced in section 3.2.1. The network architecture is shown in Appendix F Figure F.5. It consisted of three densely connected hidden layers with 216 neurons each, with ReLU as the activation function while the output layer comprised of 76 neurons with linear activation.

The inverse metamodelling using CNN (Appendix F Figure F.6) composed of five hidden layers among which the last two layers were dense layers with 200 and 500 neurons whereas the first three layers were 1D convolutional layers with 128 filters and a kernel size of 10. The convolutional layer was same padded, confirming that the data size was not reduced. The same intermediate layers used in the classical metamodelling of time series data using CNN were utilized. A fully connected layer with seventy six neurons was introduced to link the feature map with the output layer.

3.4 Permuted Feature Importance

The feature importance was calculated from multiple permutations for stability and the adjustment of random effects. Finally, the mean and the standard deviation of the importance of each of the features were listed.

4 Results

4.1 Data Generation

Out of 10,000 simulations, only 5344 simulations generated a membrane potential. A wide range of differently behaving membrane potentials were generated through the combination of sampled parameters as illustrated in Figure 4.1 **A**. The mean and the standard deviation of the 5344 simulations are also shown in Figure 4.1 **B**. The time for the first AP generation for all the parameter combinations was around 0.673 ms.

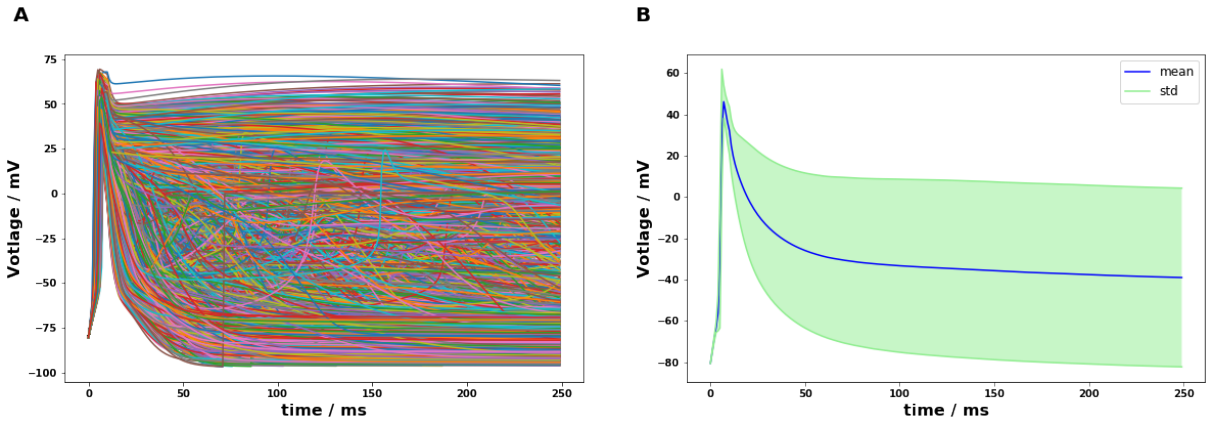


Figure 4.1: Plot of membrane potentinals generated from PHN model. A) The 5344 membrane potentials simulated from the PHN model. **B)** The mean and standard deviation of the 5344 membrane potentials simulated from the PHN model, blue: mean and green: standard deviation.

4.2 Classical Metamodelling

4.2.1 Classical Metamodelling of the time series data using HC-PLSR

The global PLSR model trained with the features selected from RENT and SBS presented a significant decrease in model performance in comparison to using all available features. Likewise, there was no improvement in the model performance with the introduction of

sinus, cosinus, and cross-interaction terms. Hence, all the features were used to train the global PLSR model. The optimal number of PCs was found to be six (Appendix D Figure D.1) as the seventh PC failed to explain the total cross-validated $\mathbf{Y}$ variance with an increment of one percent. The first six PCs explained 68% of the total cross-validated variance in the $\mathbf{Y}$-matrix.

Since the elbow method was unsuccessful in determining the distinct number of clusters, a number of HC-PLSR models were trained with clusters ranging from four to eight. Using prediction accuracy as a basis for selection, the optimal number of clusters and the value of the silhouette coefficient taking the global $\mathbf{X}$-scores as the input matrix was found to be five and 0.12 respectively. The elbow plot and the silhouette plots are illustrated in Figure 4.2. The clustering of the $\mathbf{X}$-scores with the five cluster centroids is plotted in Figure 4.3. We can observe that there is no clear cluster pattern, and this might subdue the performance of HC-PLSR which aims to identify simulations with similar behavior and model them separately.

The final accuracy of the HC-PLSR model was 0.73, with an increase of 4% compared to the initial global PLSR model. The global regression coefficients and the local regression coefficients for the five PLSR models fitted within each cluster are plotted in Figure 4.4 A and Figure 4.4 B-F. The ten most significant features were selected by sorting the standardized regression coefficients. The ten features were selected because the constraints introduced (described in the later sections) in selecting the important features using FFN and CNN resulted in similar number of features. It can be observed that regression coefficients of the fourth cluster are different compared to the other clusters. Similarly, *"Na_o"* and *"g_Na"* have high coefficient values at the initial stages of the AP generation, while *"g_D"* and *"N"* play an important role in describing the latter stages of the AP. The parameters *"g_t"*, *a_endo"*, and *"V_myo"* have a negative coefficient value, indicating a negative correlation between them and the time series AP.

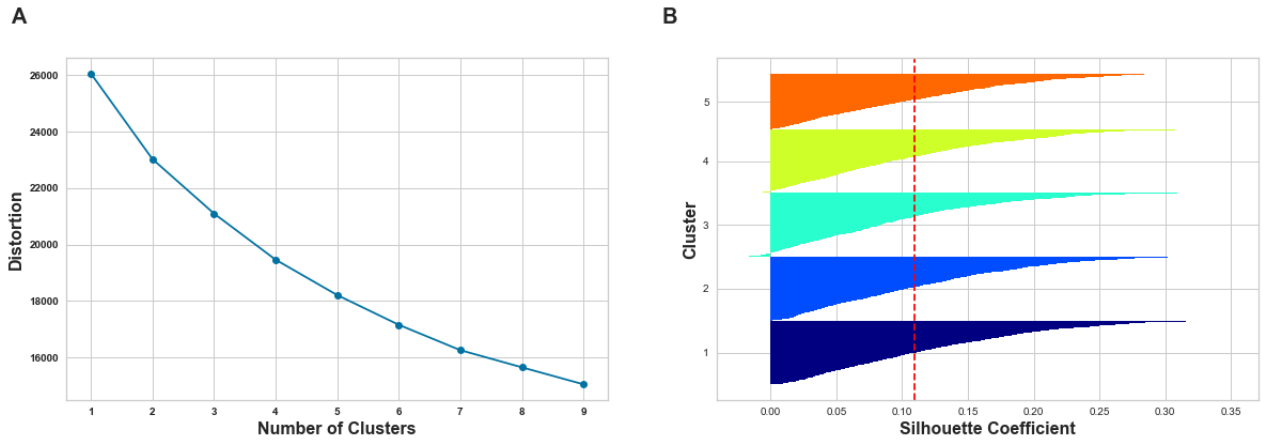


Figure 4.2: Elbow and silhouettee plots for the classical metamodelling of PHN model using HC-PLSR. A) Elbow plot considering upto nine clusters. **B)** The silhouettee plot for five clusters.

4.2.2 Classical Metamodelling of time series data using FFN

The initial FFN models when trained with the features selected from SBS achieved a test accuracy of 0.82 while the test accuracy was 0.78 for the features gathered from RENT. Moreover, it outclassed the initial model trained with all the features by 6%. The SBS selected 49 features from the 76 features to enhance the model performance. The selected features are colour-listed in Appendix A. Appendix G Figure G.1 illustrates the training plots used in hyperparameter tuning of the FFN model in classical metamodelling. Thus, the final model was trained with 49 features selected from SBS for 200 epochs and achieved a test set prediction accuracy of 0.83.

The feature importance ranking of the 49 variables was calculated using five permutations for each of the features using the feature importance permutation approach described in the Section 2.5. The features were sorted based on the their importance, and from this set, the features with minimal difference in the absolute value of the importance between them were selected (Figure 4.5). From the figure, it is clear that the effect of *"g_t"* is the highest in determining the initial AP, followed by *"del_VL"* and *"V_myo"*.

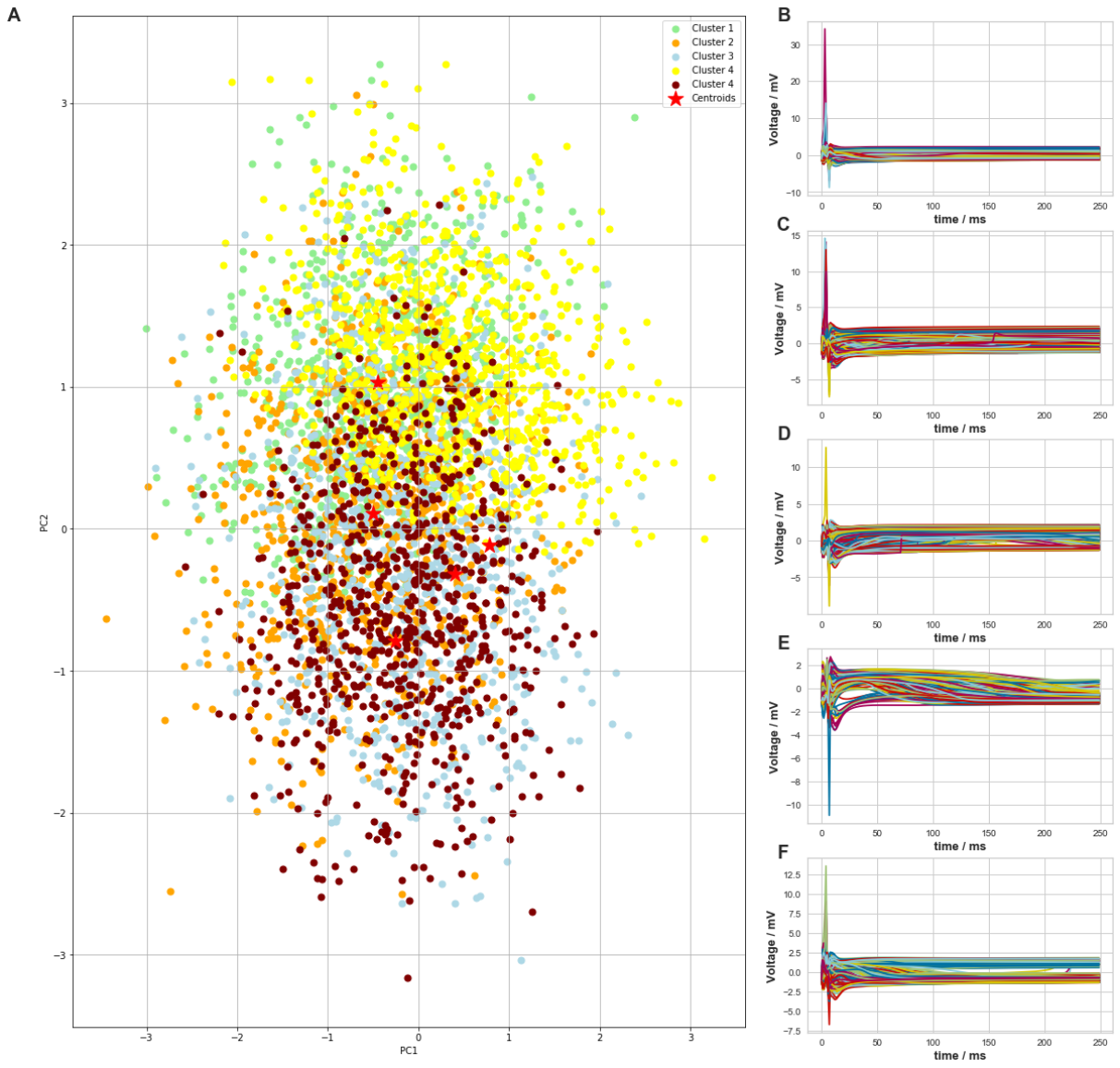


Figure 4.3: Clustering results from HCPLSR metamodelling of the PHN model using five clusters. **A)** Scatter plot of the X-scores from the global metamodel. The red stars indicate the cluster centroids and the samples are coloured according to their cluster membership. Cluster1=green, cluster2=orange, cluster3=blue, cluster4=yellow, cluster5=brown. **B-F)** illustrates the nature of the cluster membrane potentials for clusters 1-5, respectively.

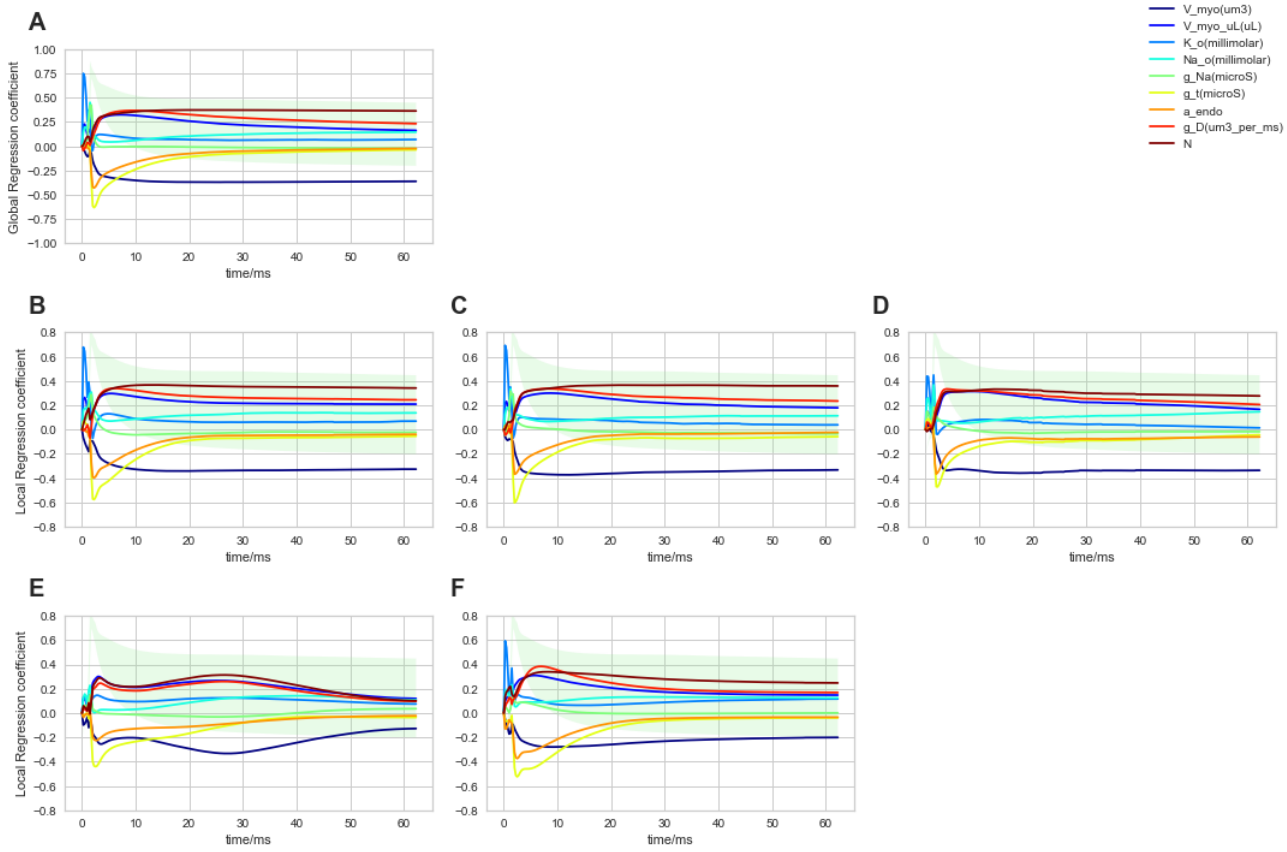


Figure 4.4: Global and local regression coefficients for the input parameters of the PHN model. The regression coefficients illustrate the sensitivity of the input parameters with time. **A)** shows the global regression coefficients. **B-F)** shows the local regression coefficients for cluster 1-5, respectively. The standard deviation of the membrane potentials belonging to each individual clusters is plotted off scale in the background for comparison.

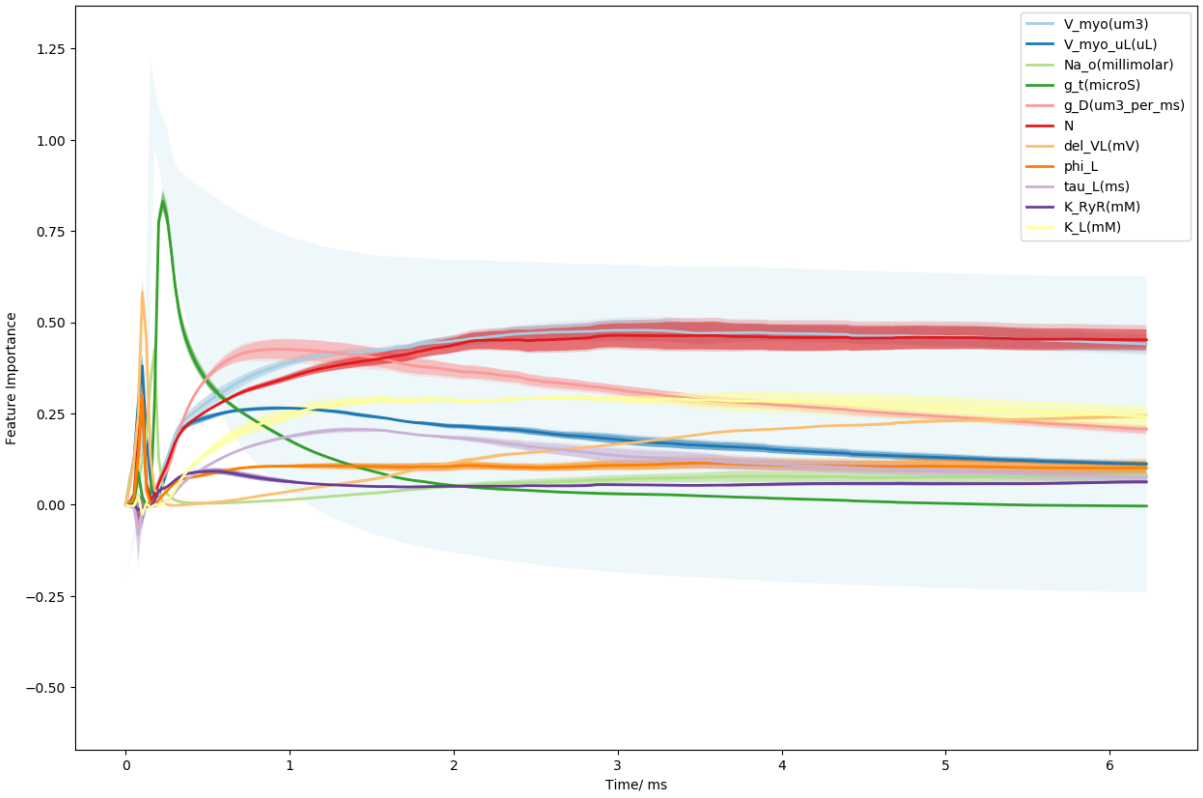


Figure 4.5: Feature importance for the input parameters of the PHN model using FFN. The feature importance is calculated with five permutations of each input variable. The standard deviation of the simulated membrane potentials is plotted off scale in the background for comparison.

4.2.3 Classical Metamodelling of the time series data using CNN

The 49 features selected from SBS were again used to train the CNN model. Figure Appendix G G.2 illustrates the training plots used in hyperparameter tuning of the CNN model in classical metamodelling. Thus, the final model was trained for 20 epochs and achieved a test set prediction accuracy of 0.93.

The feature importance of the 49 variables was calculated using five permutations for each of the features using the feature importance permutation approach described in the Section 2.5. From this, the features having a weight coefficient greater than 0.05 were gathered (Figure 4.6). The threshold was determined using the same approach as described in the previous section. It can be observed that the volume of the sarcoplasmic reticulum is the most important feature in guiding the initial AP, followed by i_NaK_Max and $''g\_NCX''$.

4.2.4 Classical Metamodelling of the aggregated phenotypes using HC-PLSR

The **X**-scores were chosen as the clustering basis for k-means clustering for the classical metamodelling of aggregated phenotypes using HC-PLSR. The model was trained using all available features as the feature selection techniques did not improve the model prediction accuracy. The first four PC's were chosen (Appendix D Figure D.2) since the fifth PC failed to increase the explained variance of the first four PCs by more than one percent. The first four PCs explained 60% of the total cross-validated variance in **Y**. Likewise, the elbow method Figure(4.7**(A)**) failed to provide the optimal number of clusters. As a result, cluster numbers ranging from four to ten were tried before landing into the optimal number of clusters which was six. The prediction accuracy achieved from each trial of the cluster number was the basis in choosing the optimal number of clusters.

In selecting the important features, a constraint was introduced such that the features having at least 40% of the highest regression coefficient values were selected. This different requirement was essential since all of the 76 features were used in training the

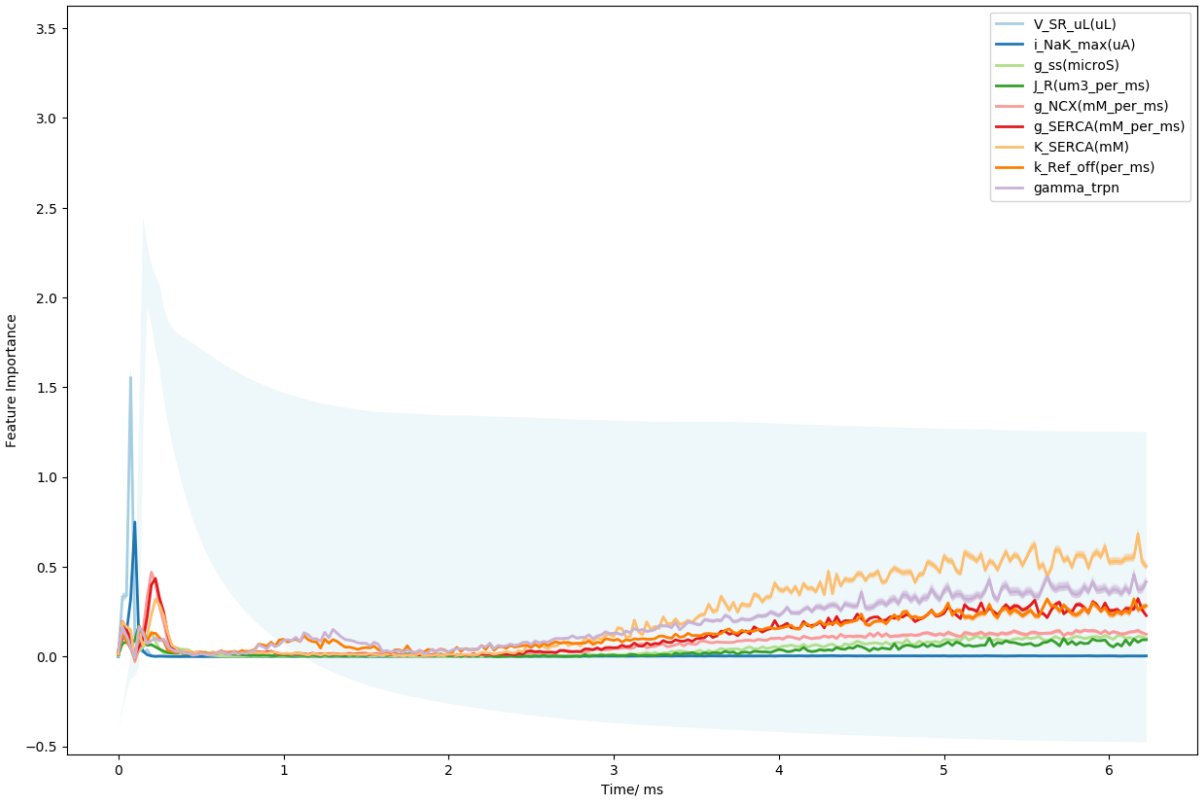


Figure 4.6: Feature importance for the input parameters of the PHN model using CNN. The feature importance is calculated with five permutations of each input variable. The standard deviation of the simulated membrane potentials is plotted off scale in the background for comparison.

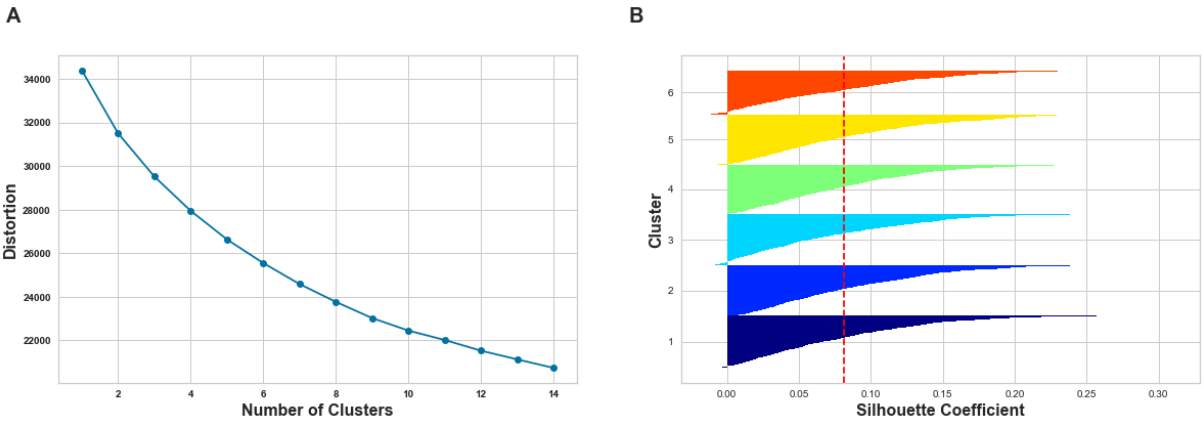


Figure 4.7: Elbow and silhouettee plots for the classical metamodelling of PHN model using HC-PLSR. A) Elbow plot considering upto fourteen clusters. **B)** The silhouettee plot for six clusters.

metamodel. This resulted in a more evenly distributed coefficient values without an observable significant decrease between them, which could be used as a basis for selection. The regression coefficients for the main effects of the extracted aggregated phenotypes are shown in Figure 4.8. The results indicate that *"g_t"* is an important parameter in determining all three phenotypes while *"Na_o"* guides both the time to the first AP and the amplitude of the first AP. Similarly, the width of the AP is highly sensitive to *"K_RyR"* and *"g_D"*.

4.2.5 Classical Metamodelling of the aggregated phenotypes using FFN

Appendix G Figure G.3 illustrates the training plots used in hyperparameter tuning of the FFN model in classical metamodelling. The model was trained for 500 epochs with a batch size of 64. The feature importance of the input variables was calculated with the same method as elaborated in the Section 2.5 and is illustrated in Figure 4.9. The results suggest that *"Na_o"* and *"g_Na"* are the important determinants of the time to the first AP and the amplitude of the first AP, while *"g_D"* and *"K_RyR"* mainly govern the AP width.

4.2.6 Classical Metamodelling of the aggregated phenotypes using CNN

Appendix G Figure G.4 illustrates the training plots used in hyperparameter tuning of the FFN model in classical metamodelling. The model was trained for 79 epochs with a batch size of 64. The feature importance of the input variables was calculated with the same method as elaborated in the Section 2.5 and is illustrated in Figure 4.10. The findings imply that *"Na_o"* and "g_Na" are primarily responsible for time of AP and its amplitude while *"g_t"* is important in determining all of three extracted phenotypes. *"K_RyR"* is another factor that influences the width of the first AP.

The test set prediction accuracies comparing the performance of FFN, global PLSR and HC-PLSR in predicting the aggregated phenotypes are listed in Table 2.

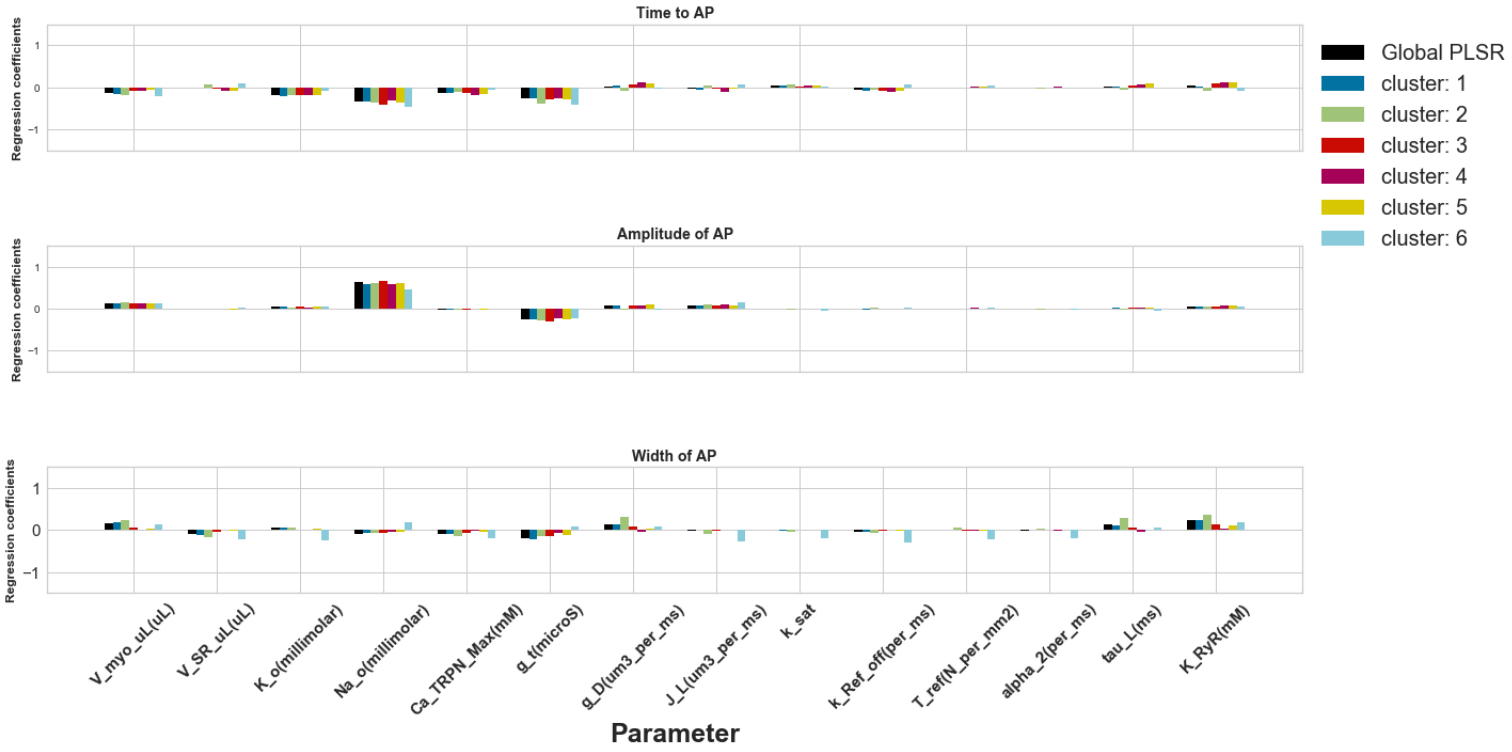


Figure 4.8: Global and local regression coefficients for the main effects of the aggregated phenotypes extracted from the PHN model. The regression coefficients show the sensitivity of the input parameters. **The top figure** shows the regression coefficients for the time taken for the membrane potential to reach the first AP. **The middle figure** shows the regression coefficients for the amplitude reached by the first AP. **The bottom figure** shows the regression coefficients for the width of the first AP. The regression coefficients are coloured according to their respective clusters.

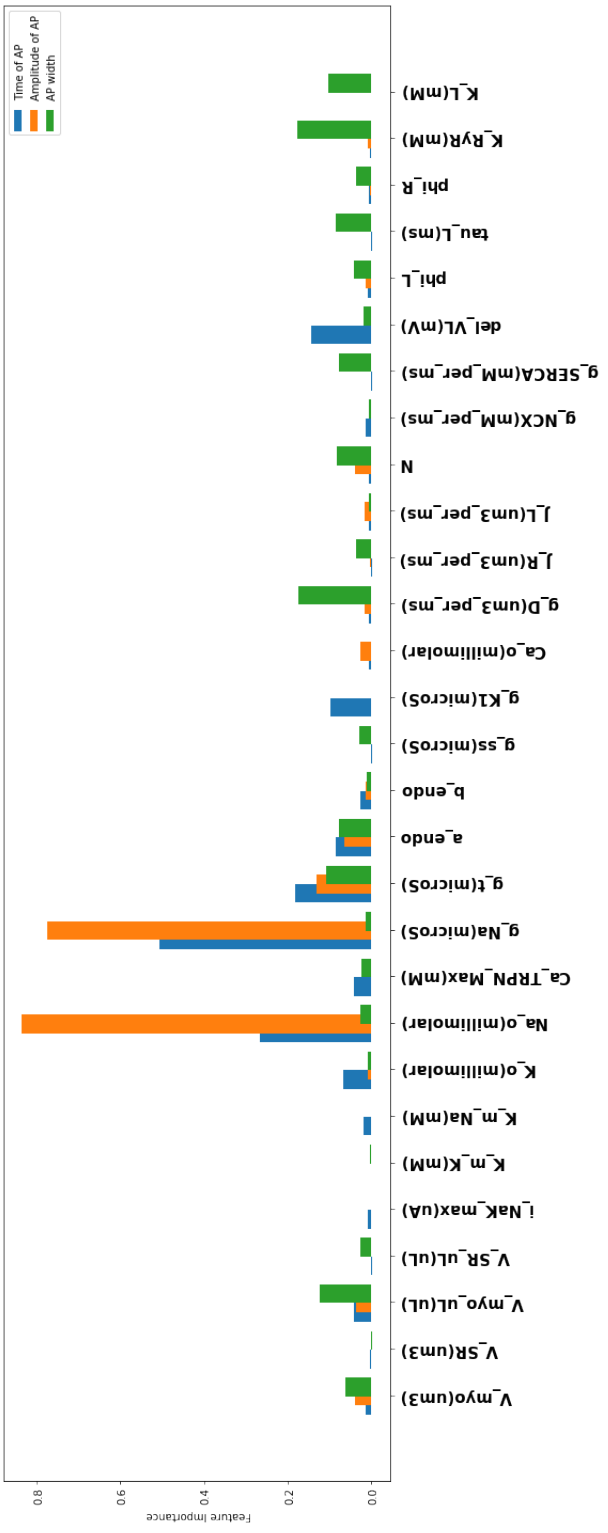


Figure 4.9: Feature importance for the input parameters of the aggregated phenotypes extracted from the first AP in the PHN model using FFN. The feature importance is calculated as the mean of five permutations for each input variable.

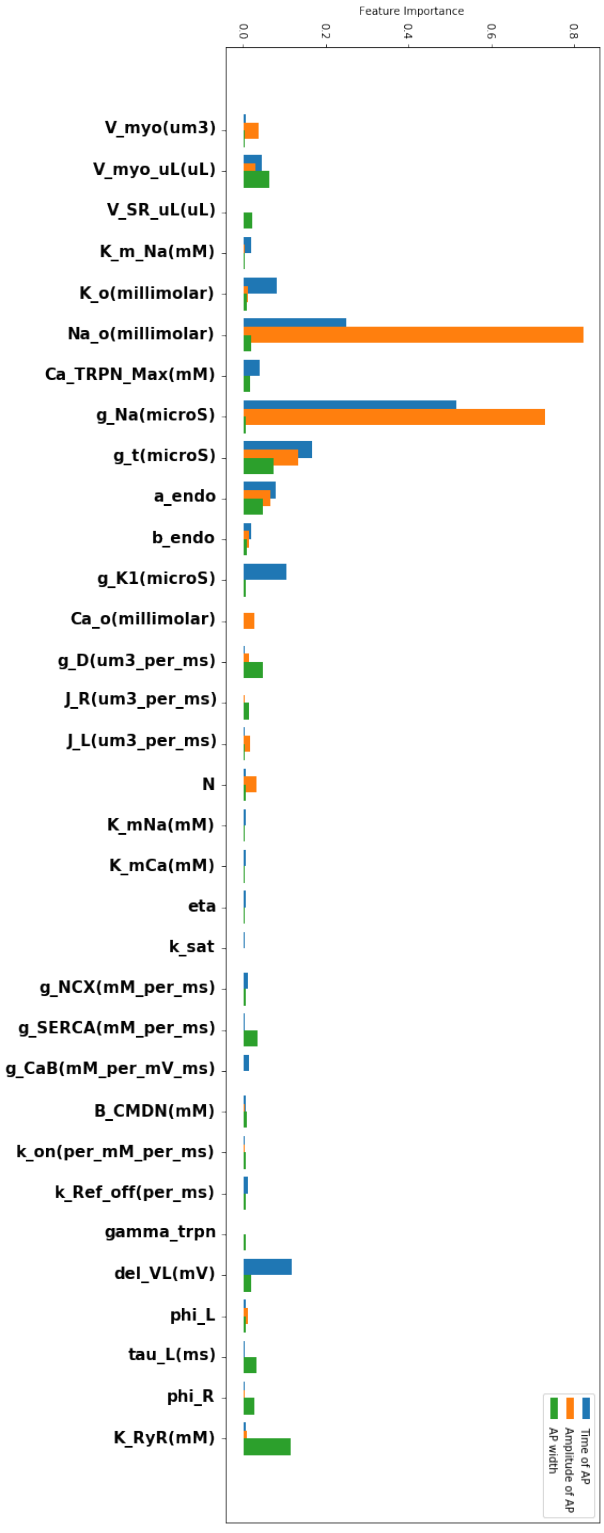


Figure 4.10: Feature importance for the input parameters of the aggregated phenotypes extracted from the first AP in the PHN model using CNN. The feature importance is calculated as the mean of five permutations for each input variable.

Table 2: Prediction results (R^2) for metamodelling of the aggregated phenotypes extracted from the PHN model using FFN, CNN, PLSR and HC-PLSR.

	FFN	CNN	Global PLSR	HC-PLSR
time to peak	0.6738	0.6857	0.6822	0.6096
width of first AP	0.9521	0.9218	0.9135	0.8999
amplitude of first AP	0.3972	0.2659	0.2438	0.1696

4.3 Inverse Metamodelling

4.3.1 Inverse Metamodelling of the time series data using HC-PLSR

The response matrix to the global PLSR model in the inverse metamodelling comprised the set of all the 76 features that were varied. The ideal number of PCs was determined as 10 (Appendix D Figure D.3) using the same method described in earlier sections, explaining 35% of the total cross-validated variance in $\mathbf{Y}$. Following this, the optimal number of clusters and the value of the silhouette coefficient taking the global $\mathbf{X}$-scores as the input matrix, and using the Elbow method, were two and 0.58 respectively. The clustering of the X-scores with the two cluster centroids is plotted in Figure 4.11. The elbow plot and the silhouette plots are illustrated in Figure 4.12. Hence, two local PLSR models were trained, and the final results are presented in Table 3.

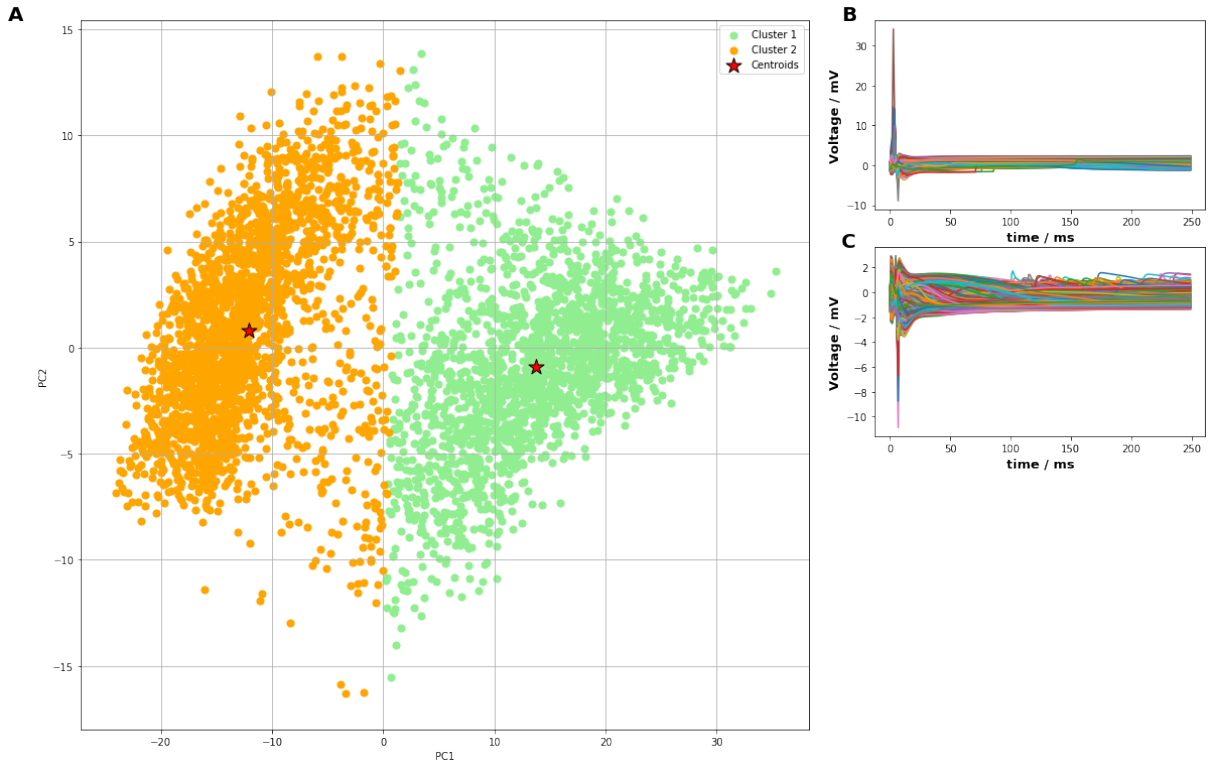


Figure 4.11: Clustering results from the inverse HC-PLSR metamodelling of the PHN model using two clusters. **A)** Scatter plot of the X-scores from the global metamodel. The red stars indicate the cluster centroids and the samples are coloured according to their cluster membership. cluster1=green, cluster2=orange, **B-C)** illustrates the nature of the cluster membrane potentials for clusters 1 and 2, respectively.

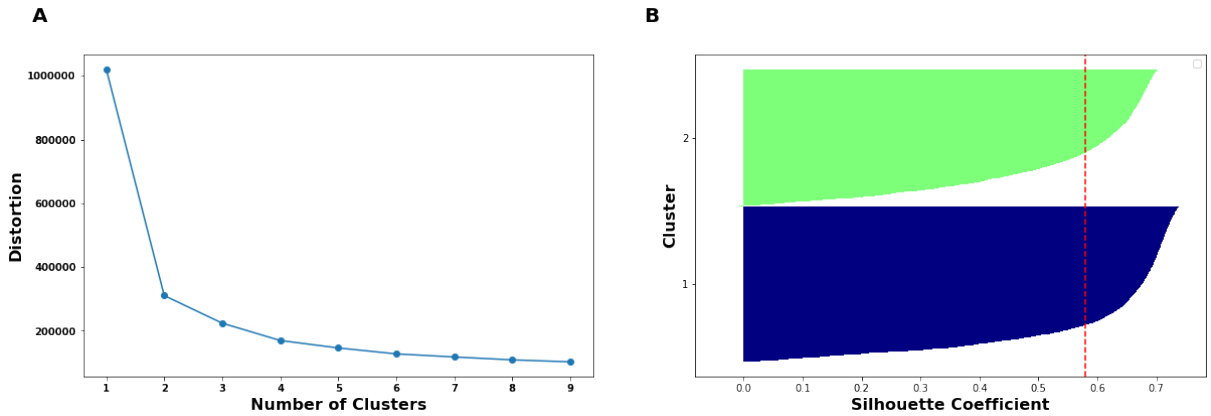


Figure 4.12: Elbow and silhouettee plots for the inverse metamodelling of PHN model using HC-PLSR. A) Elbow plot considering upto nine clusters. **B)** The silhouettee plot for two clusters.

Table 3: Prediction accuracies (R^2) for inverse metamodelling of the PHN model using HC-PLSR. The red color indicates a decrease in prediction accuracy while the green color indicates an increase.

Parameter	Global PLSR	HC-PLSR
V_myo	0.138	0.108
K_o	0.820	0.821
Na_o	0.354	0.357
V_myo_uL	0.157	0.157
g_Na	0.417	0.440
g_t	0.470	0.494
a_endo	0.208	0.212
g_D	0.215	0.206
N	0.146	0.080

4.3.2 Inverse Metamodelling of the time series data using FFN and CNN

The response matrix to the FFN and the CNN model in the inverse metamodelling comprised of all the initial parameters that were varied. Most of the variables selected from the FFN model have higher prediction accuracy than the important features selected from the high-performing CNN model. The results of inverse metamodelling for the variables selected from the FFN and CNN are listed in Table 4.

Table 4: Prediction accuracies (R^2) for inverse metamodelling of the PHN model using FFN and CNN.

Parameter	Prediction Accuracy (FFN)
V_myo	0.1717
phi_L	0.0572
Na_o	0.4435
V_myo_uL	0.2339
g_t	0.5248
K_L	0.1466
g_D	0.2394
N	0.1643
del_VL	0.2463
tau_L	0.1421
K_RyR	0.1580

Parameter	Prediction Accuracy (CNN)
V_SR_uL	0.0234
i_NaK_max	0.0046
g_ss	0.0046
J_K	0.0049
g_Ncx	0.0087
g_SERCA	0.0371
k_SERCA	0.0012
K_Ref_off	0.0243
gamma_TRPN	0.0042
-	
-	

5 Discussion

5.1 Classical Metamodelling of the time series data

A good prediction accuracy (0.69) achieved from the global PLSR model suggests that the PHN model exhibits soft non-linear behavior, since this can be modelled using the nonlinear terms included in the set of independent variables. Nonetheless, the slightly increased accuracy (4%) achieved by using HC-PLSR shows that in order to better emulate the model outputs, some abrupt non-linearities must be taken into consideration. Since a clear cluster pattern was not identified, it diminished the efficiency of HC-PLSR in modelling these abrupt non-linearities. However, the slight increase in the prediction accuracy from HC-PLSR and the lower prediction accuracy achieved with the PLSR compared to FFN and CNN indicates that there are complex non-linearities present. The best HC-PLSR model achieving a test set accuracy of 0.73 was surpassed by FFN (0.84) and CNN (0.93) model indicating that CNN is more efficient at emulating the PHN model.

The k-means clustering used in HC-PLSR, to some extent, was able to differentiate the divergent behavior of the PHN model. Figure 4.4 shows how the parameter space was divided into subspaces (B-F) using clustering. This further increased the interpretability of the importance of the input parameters to different model behavior. It is a well known fact that the initial AP is caused by the sodium current and followed by the repolarization of the membrane potential with the potassium current. These mechanics are confirmed by the local regression coefficients of the input parameters in Figure 4.4 (B-F), which illustrates the influence of $''Na\_o''$ which is governed by the somatic conductance of sodium $''g\_Na''$, is increasing with the membrane potential. When inspecting these regression coefficients closely, their value are lower in clusters Figure 4.4 (D-E) in comparison to other clusters, which explains the reason behind their lower peak AP. The parameter $''g\_t''$ is a component associated with the Ca independent transient outward K current which

explains the early phase shaping of the AP [61]. It is responsible for the prolongation or the shortening of APD with a high correlation between it and prominence of Phase-1 notch [61]. Likewise, dyadic space conductance ($''g\_D''$) and the number of release units ($''N''$) are associated components in CaRU which describe the behavior of the membrane potential during the final stages.

The permuted feature importance calculated from the FFN and CNN model were not able to provide a thorough explanation of the model behaviors, since it is based on the complete dataset. As a result, it falls short of characterizing highly non-linear input-output correlations and revealing additional patterns of covariation, unlike cluster-based approaches to regional metamodeling. The generalized overview provided by the permuted feature importance of the FFN model is somewhat similar to the global regression coefficients Figure 4.4 A. It can be elucidated from the feature importances that $''g\_t''$ is the most important parameter that explains the behavior leading to the initial AP. They also indicate that $''del\_VL''$ (width of opening potentials), a component in CaRU transition, $''Na\_o''$ and the volume of myocytes (uL) are the other main parameters influencing the generation of the AP.

However, the best emulating model recognized the volume of the sarcoplasmic reticulum $''V\_SR''$ and $''i\_NaK\_max''$ which is a component related to the sodium potassium pump as the significant parameters which guide the initial phase of the AP. Likewise, the components $''g\_NCX''$ (component in sodium-calcium exchanger), $''g\_SERCA''$ and $''k\_SERCA''$ are important in describing the latter phase of the AP. It is apparent that CNN focuses in predicting the latter stages of the AP, which is often difficult to predict.

5.2 Classical Metamodelling of the aggregated phenotypes

Overall, FFN achieved higher prediction score for the extracted phenotypes from the PHN model. However, the CNN model achieved a slightly better score (by 1%) in predicting "time to peak". All models performed well in predicting the "width of the first AP"

(FFN being the best). However, all of them struggled in predicting the "amplitude of first AP" with HC-PLSR making predictions that were worse than the calculated average amplitude of the first AP. From the results, it is apparent that **XY** relationships are too complex for all the methods to make predictions regarding the first AP amplitude.

Both the feature importance permutation from FFN and CNN the local regression coefficients identified *"g_Na"* as the most critical parameter in predicting the time until the first AP, followed by *"Na_o,"* *"g_t"* and *"del_VL"* (width of opening potentials). Not surprisingly, the amplitude of the initial AP was mainly explained by *"Na_o"* and *"g_Na"* in all three methods. While the HC-PLSR was unable to predict the width of AP accurately, the permuted feature importance in FFN and CNN recognized *"K_RyR"* (half concentration of activation) as the most important feature followed by *"g_D"* and *"V_myo"*.

5.3 Inverse Metamodelling of the time series data

The inverse metamodelling using HC-PLSR had difficulties predicting *"N"* and *"V_myo"*, but it did better than the global PLSR model in predicting other important variables. Most of the sensitive parameters from HC-PLSR and FFN were overlapping. The FFN model outperformed the HC-PLSR model in predicting these important common parameters. The prediction ability of CNN was worst of all in predicting the features selected from the permuted feature importance. The prediction accuracy for all variables was below 1%.

The sloppiness of the models might explain the low prediction accuracies in inverse metamodelling. Even though a parameter is considered important, it might be that several different combinations of parameter values can give similar outputs. An increase in a parameter value and a decrease in an other parameter value might balance the changes. The outputs can still be similar for different simulations resulting in a many to one problem in predicting back to the parameters. There is no efficient method to fix

this problem. However, one can try to identify manifolds or the ranges of the parameter values that generate similar outputs instead of directly predicting them.

5.4 Comparison between HC-PLSR and Deep Learning

Considering the emulation capacities, it is evident that the deep learning models (both FFN and CNN) surpass the HC-PLSR model. While it may well be true that increasing the number of clusters in the HC-PLSR model might achieve a greater prediction ability, this eventually leads to an increased computational cost due to a longer clustering time and a large number of PLSR models that must be calibrated. In addition, when dealing with experimental data, with the increase in the number of clusters, the number of observations required to model each cluster's behavior increases simultaneously. However, when using simulated data, the total number of observations is usually high. Also, difficulties arise in explaining the model behavior when the number of clusters is high. Similarly, deep learning models are unable to explain its predictions and interpret in terms of human intuition. It is also worth noting that the deep learning models used in this study could be improved further.

HC-PLSR's strength lies in revealing regional differences in the model sensitivity to various input parameters. This is aided by exploring regression coefficients or the loading plots from the local and global modelling. The local models also help identify the operative domains of the parameter space more efficiently. However, the behavioral insights presented by the HC-PLSR metamodelling might be restricted by its ability to emulate highly complex models with high granularity such that it hinders the local interpolation in the parameter space [62]. Hence, an important question arises on the amount of information that HC-PLSR is unable to explain. Another downside with the PLSR subspace methodology is that it fails to generate informative component plots in situations where the mechanistic model does not have a robust input-output relationship but only a wide range of of equally important relationships [62].

When comparing the time consumption for the sensitivity analysis, HC-PLSR has an advantage as the PLSR coefficients that are examined are predicted simultaneously. Usually, the PLSR modelling is carried out with standardized variables to ensure that the regression coefficients for different input variables are comparable. However, the permuted feature importance method permutes and analyzes each variable at a time, and becomes computationally demanding with the increase in the number of input parameters.

6 Conclusion

The results in this thesis indicate that deep learning metamodels are highly effective compared to the HC-PLSR at emulating the PHN model, which is a complex non-linear mathematical model. Likewise, the findings show that HC-PLSR is an efficient tool for model analysis as it divides the original parameter space into subspaces. This allows for a detailed investigation of the input parameters that influence the model outputs without prior knowledge about the model through the scores and the loadings plots. Despite their higher emulation capacity, deep learning methods require adjustments relating to a prior knowledge (data augmentation, encoding input vectors, and use of pre-trained models) of the model output to achieve similar insights.

6.1 Further Works

This thesis utilizes a crisp clustering algorithm for the implementation of the HC-PLSR model. One can explore and implement clustering methods like FCM [63] and Robust Agglomeration Algorithm (RCA) to determine optimal number of clusters [64], which might produce better clustering results allowing for the metamodelling of complex models.

As more and more high-dimensional measuring devices are used in computational biology, the size and complexity of the mathematical models are ever-increasing. An extension of HC-PLSR termed as N-way HC-PSLR [4] is another efficient approach that combines several regional N-way PLSR models. A benefit of this approach is that it enables the modelling of multiple state variables at a time. Hence, it is a useful tool for multivariate metamodelling of spatiotemporal models.

Similarly, clustering techniques could be used in combination with deep learning such that it too generalizes on the local parameter space, thereby increasing its interpretability. The deep embedding approach described by [65] could be utilized for subspace analysis.

References

[1] D. Sterratt, B. Graham, A. Gillies, and D. Willshaw, *Principles of computational modelling in neuroscience*. Cambridge University Press, 2011.

[2] K. Tøndel, U. G. Indahl, A. B. Gjuvsland, J. O. Vik, P. Hunter, S. W. Omholt, and H. Martens, "Hierarchical cluster-based partial least squares regression (hc-plsr) is an efficient tool for metamodelling of nonlinear dynamic models," *BMC Systems Biology*, vol. 5, no. 1, pp. 1–17, 2011.

[3] K. Tøndel, S. A. Niederer, S. Land, and N. P. Smith, "Insight into model mechanisms through automatic parameter fitting: a new methodological framework for model development," *BMC systems biology*, vol. 8, no. 1, pp. 1–20, 2014.

[4] K. Tøndel, U. G. Indahl, A. B. Gjuvsland, S. W. Omholt, and H. Martens, "Multi-way metamodelling facilitates insight into the complex input-output maps of nonlinear dynamic models," *BMC systems biology*, vol. 6, no. 1, pp. 1–21, 2012.

[5] J. P. Kleijnen, "Design and analysis of simulation experiments," in *International Workshop on Simulation*. Springer, 2015, pp. 3–22.

[6] G. Li, H. Rabitz, P. E. Yelvington, O. O. Oluwole, F. Bacon, C. E. Kolb, and J. Schoendorf, "Global sensitivity analysis for systems with independent and/or correlated inputs," *The journal of physical chemistry A*, vol. 114, no. 19, pp. 6022–6032, 2010.

[7] D. J. Fonseca, D. O. Navaresse, and G. P. Moynihan, "Simulation metamodeling through artificial neural networks," *Engineering applications of artificial intelligence*, vol. 16, no. 3, pp. 177–183, 2003.

[8] J. R. Terkildsen, S. Niederer, E. J. Crampin, P. Hunter, and N. P. Smith, "Using physiome standards to couple cellular functions for rat cardiac excitation–contraction," *Experimental physiology*, vol. 93, no. 7, pp. 919–929, 2008.

[9] D. Gorissen, K. Crombecq, I. Couckuyt, and T. Dhaene, "Automatic approximation of expensive functions with active learning," in *Foundations of Computational, Intelligence Volume 1*. Springer, 2009, pp. 35–62.

[10] H. Martens, K. Tøndel, V. Tafintseva, A. Kohler, E. Plahte, J. O. Vik, A. B. Gjuvsland, and S. W. Omholt, "Pls-based multivariate metamodeling of dynamic systems," in *New Perspectives in Partial Least Squares and Related Methods*. Springer, 2013, pp. 3–30.

[11] L. Yu, S. Wang, and K. K. Lai, "A neural-network-based nonlinear metamodeling approach to financial time series forecasting," *Applied Soft Computing*, vol. 9, no. 2, pp. 563–574, 2009.

[12] R. A. Kilmer, A. E. Smith, and L. J. Shuman, "An emergency department simulation and a neural network metamodel," *Journal of the society for health systems*, vol. 5, no. 3, pp. 63–79, 1997.

[13] R. N. Gutenkunst, J. J. Waterfall, F. P. Casey, K. S. Brown, C. R. Myers, and J. P. Sethna, "Universally sloppy parameter sensitivities in systems biology models," *PLoS Comput Biol*, vol. 3, no. 10, p. e189, 2007.

[14] P. Geladi and B. R. Kowalski, "Partial least-squares regression: a tutorial," *Analytica chimica acta*, vol. 185, pp. 1–17, 1986.

[15] M. Haenlein and A. M. Kaplan, "A beginner's guide to partial least squares analysis," *Understanding statistics*, vol. 3, no. 4, pp. 283–297, 2004.

[16] S. Raschka, *Python machine learning*. Packt publishing ltd, 2015.

[17] K. Tøndel, J. O. Vik, H. Martens, U. G. Indahl, N. Smith, and S. W. Omholt, "Hierarchical multivariate regression-based sensitivity analysis reveals complex parameter interaction patterns in dynamic models," *Chemometrics and Intelligent Laboratory Systems*, vol. 120, pp. 25–41, 2013.

[18] A. Florian, "An efficient sampling scheme: updated latin hypercube sampling," *Probabilistic engineering mechanics*, vol. 7, no. 2, pp. 123–130, 1992.

[19] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[20] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[21] G. Palermo, P. Piraino, and H.-D. Zucht, "Performance of pls regression coefficients in selecting variables for each response of a multivariate pls for omics-type data," *Advances and applications in bioinformatics and chemistry: AABC*, vol. 2, p. 57, 2009.

[22] S. Raschka and V. Mirjalili, "Python machine learning: Machine learning and deep learning with python," *Scikit-Learn, and TensorFlow. Second edition ed*, 2017.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[24] M. Pontil and A. Verri, "Properties of support vector machines," *Neural Computation*, vol. 10, no. 4, pp. 955–974, 1998.

[25] P. Schiilkop, C. Burgest, and V. Vapnik, "Extracting support data for a given task," in *Proceedings, First International Conference on Knowledge Discovery & Data Mining. AAAI Press, Menlo Park, CA*, 1995, pp. 252–257.

[26] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[27] ——, "Some properties of splitting criteria," *Machine Learning*, vol. 24, no. 1, pp. 41–47, 1996.

[28] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[29] 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.

[30] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[31] M. Dash and H. Liu, "Feature selection for classification," *Intelligent data analysis*, vol. 1, no. 1-4, pp. 131–156, 1997.

[32] S. Raschka, "Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack," *The Journal of Open Source Software*, vol. 3, no. 24, Apr. 2018. [Online]. Available: http://joss.theoj.org/papers/10.21105/joss.00638

[33] A. Jenul, S. Schrunner, K. H. Liland, U. G. Indahl, C. M. Futsaether, and O. Tomic, "Rent – repeated elastic net technique for feature selection," 2021.

[34] A. Jenul, "Rent - repeated elastic net technique for feature selection," *Zenodo*, 2021. [Online]. Available: https://zenodo.org/badge/DOI/10.5281/zenodo.4588447.svg

[35] A. Blum, A. Kalai, and J. Langford, "Beating the hold-out: Bounds for k-fold and progressive cross-validation," in *Proceedings of the twelfth annual conference on Computational learning theory*, 1999, pp. 203–208.

[36] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," 2020.

[37] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[38] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, "The analysis of a simple k-means clustering algorithm," in *Proceedings of the sixteenth annual symposium on Computational geometry*, 2000, pp. 100–109.

[39] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[40] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[41] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[42] C. Stangor, J. Walinga *et al.*, *Introduction to psychology.* BCcampus, BC Open Textbook Project, 2014.

[43] F. Chollet *et al.*, *Deep learning with Python.* Manning New York, 2018, vol. 361.

[44] C. C. Aggarwal *et al.*, "Neural networks and deep learning," *Springer*, vol. 10, pp. 978–3, 2018.

[45] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, vol. 2, no. 11, p. e7, 2017.

[46] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[47] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[48] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[49] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *arXiv preprint arXiv:1706.02515*, 2017.

[50] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[51] 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.

[52] 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: http://tensorflow.org/

[53] F. Chollet, "Building autoencoders in keras," *The Keras Blog*, vol. 14, 2016.

[54] L. Sherwood, *Human physiology: from cells to systems*. Cengage learning, 2015.

[55] Drawittoknowit.com, "Physiology glossary: Cardiac muscle action potential," Apr 2021. [Online]. Available: https://www.drawittoknowit.com/course/physiology/glossary/physiological-process/cardiac-muscle-action-potential

[56] S. V. Pandit, R. B. Clark, W. R. Giles, and S. S. Demir, "A mathematical model of action potential heterogeneity in adult rat left ventricular myocytes," *Biophysical journal*, vol. 81, no. 6, pp. 3029–3051, 2001.

[57] R. Hinch, J. Greenstein, A. Tanskanen, L. Xu, and R. Winslow, "A simplified local control model of calcium-induced calcium release in cardiac ventricular myocytes," *Biophysical journal*, vol. 87, no. 6, pp. 3723–3736, 2004.

[58] S. Niederer, P. Hunter, and N. Smith, "A quantitative analysis of cardiac myocyte relaxation: a simulation study," *Biophysical journal*, vol. 90, no. 5, pp. 1697–1722, 2006.

[59] H. Nielesens, *Schematic diagram of the CellML model*. Physiome Model Repository, Aug 2010. [Online]. Available: https://models.physiomeproject.org/workspace/terkildsen_niederer_crampin_hunter_smith_2008/rawfile/cebc92eb074b5acf345f88c53ca883169675bca3/terkildsen_2008.png

[60] L. S. Aiken, S. G. West, and R. R. Reno, *Multiple regression: Testing and interpreting interactions*. sage, 1991.

[61] J. L. Greenstein, R. Wu, S. Po, G. F. Tomaselli, and R. L. Winslow, "Role of the calcium-independent transient outward current i to1 in shaping action potential morphology and duration," *Circulation research*, vol. 87, no. 11, pp. 1026–1033, 2000.

[62] K. Tøndel and H. Martens, "Analyzing complex mathematical model behavior by partial least squares regression-based multivariate metamodeling," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, no. 6, pp. 440–475, 2014.

[63] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, no. 7, pp. 773–780, 1989.

[64] H. Frigui and R. Krishnapuram, "A robust competitive clustering algorithm with applications in computer vision," *Ieee transactions on pattern analysis and machine intelligence*, vol. 21, no. 5, pp. 450–465, 1999.

[65] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*. PMLR, 2016, pp. 478–487.

A Default Parameter Values

Table 5: The 76 parameters that were varied using LHS and their default values. The features selected using SBS are highlighted in purple.

Parameter	Default Value
V_myo in component cell geometry (um^3)	25.85e^3
V_SR in component cell geometry (um^3)	2.098e^3
V_myo_uL in component cell geometry (uL)	25.85e^{-6}
V_SR_uL in component cell geometry (uL)	2.098e^{-6}
i_NaK_max in component sodium-potassium pump (uA)	0.95e^{-4}
K_m_K in component sodium-potassium pump (mM)	1.5
K_m_Na in component sodium-potassium pump (mM)	10
K_o in component standard ionic concentrations (millimolar)	5.4
Na_o in component standard ionic concentrations (millimolar)	140
Ca_TRPN_Max in component troponin (mM)	70e^{-3}
g_Na in component sodium current (microS)	0.8
g_t in component Ca independent transient outward K current (microS)	0.035
a_endo in component Ca independent transient outward K current (dimensionless)	0.583
b_endo in component Ca independent transient outward K current (dimensionless)	0.417
g_ss in component steady state outward K current (microS)	0.007
g_K1 in component inward rectifier (microS)	0.024
g_f in component hyperpolarisation activated current (microS)	0.00145
f_Na in component hyperpolarisation activated current (dimensionless)	0.2
g_B_Na in component background currents (microS)	0.00008015
g_B_Ca in component background currents (microS)	0.0000324
g_B_K in component background currents (microS)	0.000138
E_Ca in component background currents (millivolt)	65
Ca_o in component standard ionic concentrations (millimolar)	1.2
g_D in component CaRU (um^3/ms)	0.065
J_R in component CaRU (um^3/ms)	0.02
J_L in component CaRU (um^3/ms)	9.13e^{-4}

Parameter	Default Value
N in component CaRU (dimensionless)	5000
K_mNa in component Na-Ca Exchanger (mM)	87.5
K_mCa in component Na-Ca Exchanger (mM)	1.38
eta in component Na-Ca Exchanger (dimensionless)	0.35
k_sat in component Na-Ca Exchanger (dimensionless)	0.1
g_NCX in component Na-Ca Exchanger (mM/s)	$38.5e^{-3}$
g_SERCA in component SERCA (mM/ms)	$0.45e^{-3}$
K_SERCA in component SERCA (mM)	$0.5e^{-3}$
g_pCa in component Sarcolemmal Ca pump (mM/ms)	$0.0035e^{-3}$
K_mpCa in component Sarcolemmal Ca pump (mM)	$0.5e^{-3}$
g_CaB in component Background Ca current (mM_per_mV_ms)	$2.6875e^{-8}$
g_SRl in component SR Ca leak current (per_ms)	$1.8951e^{-5}$
k_CMDN in component calmodulin Ca buffer (mM)	$2.382e^{-3}$
B_CMDN in component calmodulin Ca buffer (mM)	$50e^{-3}$
k_on in component troponin (per_mM_per_ms)	100
k_Ref_off in component troponin (/ms)	0.2
gamma_trpn in component troponin (dimensionless)	2
alpha_0 in component tropomyosin (/ms)	$8e^{-3}$
alpha_r1 in component tropomyosin (/ms)	$2e^{-3}$
alpha_r2 in component tropomyosin (/ms)	$1.75e^{-3}$
n_Rel in component tropomyosin (dimensionless)	3
K_z in component tropomyosin (dimensionless)	0.15
n_Hill in component tropomyosin (dimensionless)	3
Ca_50ref in component tropomyosin (mM)	$1.05e^{-3}$
z_p in component tropomyosin (dimensionless)	0.85
beta_1 in component tropomyosin (dimensionless)	-4
beta_0 in component filament overlap (dimensionless)	4.9
T_ref in component length independent tension (N/mm^2)	56.2
a in component Cross Bridges (dimensionless)	0.35
A_1 in component Cross Bridges (dimensionless)	-29
A_2 in component Cross Bridges (dimensionless)	138
A_3 in component Cross Bridges (dimensionless)	129
alpha_1 in component Cross Bridges (/ms)	0.03
alpha_2 in component Cross Bridges (/ms)	0.13

Parameter	Default Value
alpha_3 in component Cross Bridges (/ms)	0.625
V_L in component CaRU Transitions (mV)	-2
del_VL in component CaRU Transitions (mV)	7
phi_L in component CaRU Transitions (dimensionless)	2.35
t_L in component CaRU Transitions (ms)	1
tau_L in component CaRU Transitions (ms)	650
t_R in component CaRU Transitions (ms)	1.17
tau_R in component CaRU Transitions (ms)	2.43
phi_R in component CaRU Transitions (dimensionless)	0.05
theta_R in component CaRU Transitions (dimensionless)	0.012
K_RyR in component CaRU Transitions (mM)	$41e^{-3}$
K_L in component CaRU Transitions (mM)	$0.22e^{-3}$
a in component CaRU Transitions (dimensionless)	14
b in component CaRU Transitions (dimensionless)	14
c in component CaRU Transitions (dimensionless)	0.01
d in component CaRU Transitions (dimensionless)	100
tau_s_ss in component steady state outward K current s_ss gate (second)	2.100

B PHN Model Equations

The original model equations for the individual models can be found in the individual papers. Here, only the corrected model equations that gave rise to a combined PHN model are enlisted.

B.1 Corrected Pandit Equations

The initial conditions for the Pandit endocardical cell model are described in Figure B.1.

State Variable	Description	Initial Value
V_m	Membrane potential	−79.565681 mV
m	I_{Na} activation gating variable	0.004825174
h	I_{Na} fast inactivation gating variable	0.641759447
j	I_{Na} slow inactivation gating variable	0.641671606
d	$I_{Ca,L}$ activation gating variable	2.579718 e^{-6}
f_{11}	$I_{Ca,L}$ fast inactivation gating variable	0.999944746
f_{12}	$I_{Ca,L}$ slow inactivation gating variable	0.999944746
Ca_{inact}	Ca^{2+}-inactivation gating variable	0.985762725
r	I_t activation gating variable	0.002362996
s	I_t fast inactivation gating variable	0.87713552
s_{slow}	I_t slow inactivation gating variable	0.410011002
r_{ss}	I_{ss} activation gating variable	0.003126432
s_{ss}	I_{ss} inactivation gating variable	0.2965757
y	I_f inactivation gating variable	0.003051783
$[Na^+]_i$	Intracellular Na^+ concentration	10.10341842 mM
$[K^+]_i$	Intracellular K^+ concentration	137.4335936 mM
$[Ca^{2+}]_i$	Myoplasm Ca^{2+} concentration	1.319981172 e^{-4} mM
$[Ca^{2+}]_{NSR}$	NSR Ca^{2+} concentration	7.75677853 e^{-2} mM
$[Ca^{2+}]_{SS}$	Restricted subspace Ca^{2+} concentration	1.44390143 e^{-4} mM
$[Ca^{2+}]_{JSR}$	JSR Ca^{2+} concentration	7.72445851 e^{-2} mM
P_{C1}	Fraction of channels in state P_{C1}	0.537662547
P_{C2}	Fraction of channels in state P_{O1}	0.46178896
P_{O1}	Fraction of channels in state P_{O2}	0.0005512441
P_{O2}	Fraction of channels in state P_{C2}	3.483581253 e^{-9}
l_{trpn}	Concentration of Ca^{2+}-bound low-affinity troponin sites	0.00816340866 mM
h_{trpn}	Concentration of Ca^{2+}-bound high-affinity troponin sites	0.139650948 mM

Figure B.1: Initial conditions for the Pandit endocardial cell model

$$\tau_{\mathrm{m}} = \frac{0.00136}{\frac{0.32(V_{\mathrm{m}}+47.13)}{1-\mathrm{e}^{-0.1(V_{\mathrm{m}}+47.13)}} + 0.08\mathrm{e}^{\frac{-V_{\mathrm{m}}}{11}}} \tag{B.1.1}$$

$$\tau_h = \begin{cases} 0.0004537\left(1+\mathrm{e}^{\frac{-(\gamma_{\mathrm{a}}+10.66)}{11.1}}\right) & \text{if } V_{\mathrm{m}} \geq -40\mathrm{mV} \\ \frac{0.00349}{0.135\mathrm{e}^{\frac{-(\mathrm{ra}+50)}{0.5}}+3.56\mathrm{c}^{0.079\ \mathrm{V_m}+310000\mathrm{e}^{0.35\ \mathrm{V_{tn}}}}} & \text{otherwise} \end{cases} \tag{B.1.2}$$

$$\tau_{f_{11}} = 0.105e^{-\left(\frac{V_m+45}{12}\right)^2} + \left(\frac{0.04}{1+\mathrm{e}^{\frac{-V_{\mathrm{m}}+25}{25}}}\right) + \left(\frac{0.015}{1+\mathrm{e}^{\frac{V_{\mathrm{m}}+75}{25}}}\right) + 0.0017 \tag{B.1.3}$$

$$\tau_{f_{12}} = 0.041\mathrm{e}^{-\left(\frac{V_{\mathrm{m}}+47}{12}\right)^2} + \left(\frac{0.08}{1+\mathrm{e}^{\frac{V_{\mathrm{m}}+55}{-5}}}\right) + \left(\frac{0.015}{1+\mathrm{e}^{\frac{V_{\mathrm{m}}+75}{25}}}\right) + 0.0017 \tag{B.1.4}$$

$$i_{\mathrm{K1}} = \left(\frac{48}{\mathrm{e}^{\frac{V_{\mathrm{m}}+37}{25}}+\mathrm{e}^{\frac{V_{\mathrm{m}}+37}{-25}}}+10\right)\left(\frac{0.001}{1+\mathrm{e}^{\frac{V_{\mathrm{m}}-(E_{\mathrm{K}}+7.77)}{-17}}}\right)+\frac{g_{\mathrm{K1}}\left(V_{\mathrm{m}}-(E_{\mathrm{K}}+1.73)\right)}{\left(1+\mathrm{e}^{\frac{1.613F(\ \mathrm{V_m}-(E_{\mathrm{K}}+1.73)}{RT}}\right)\left(1+\mathrm{e}^{\frac{K_0-0.998}{-0.124}}\right)} \tag{B.1.5}$$

$$I_{\mathrm{NaK}} = \bar{I}_{\mathrm{NaK}}\left(\frac{1}{1+0.1245\mathrm{e}^{\frac{-0.1\ \mathrm{V_m}F}{RT}}+0.0365\sigma\mathrm{e}^{\frac{-V_{\mathrm{W}}F}{RT}}}\right)\left(\frac{K_{\mathrm{o}}}{K_{\mathrm{o}}+K_{\mathrm{m_K}}}\right)\left(\frac{1}{1+\left(\frac{K_{\mathrm{m_N}}}{\mathrm{Na_i}}\right)^{1.5}}\right) \tag{B.1.6}$$

$$\frac{\mathrm{d}V_{\mathrm{m}}}{\mathrm{d}t} = \frac{-\left(I_{\mathrm{Na}}+I_{\mathrm{CaL}}+I_{\mathrm{t}}+I_{\mathrm{SS}}+I_{\mathrm{f}}+I_{\mathrm{K1}}+I_{\mathrm{B}}+I_{\mathrm{NaK}}+I_{\mathrm{NaCa}}+I_{\mathrm{CaP}}-I_{\mathrm{stim}}\right)}{C_{\mathrm{m}}} \tag{B.1.7}$$

where, $\mathrm{I_{stim}}$ is 0.6 nA for 5 ms.

B.2 Corrected Hinch Equations

$$\frac{\mathrm{d}\left[\mathrm{Ca}^{2+}\right]_{\mathrm{i}}}{\mathrm{d}t} = \beta_{\mathrm{i}}\left(I_{\mathrm{LCC}} + I_{\mathrm{RyR}} - I_{\mathrm{SERCA}} + I_{\mathrm{SR,1}} + I_{\mathrm{NCX}} - I_{\mathrm{pCa}} + I_{\mathrm{CaB}} + I_{\mathrm{TRPN}}\right) \tag{B.2.1}$$

$$[\mathrm{TRPN}]_0 = \frac{k^-_{\mathrm{TRPN}}[B]_{\mathrm{TRPN}}}{k^-_{\mathrm{TRPN}} + k^+_{\mathrm{TRPN}}\left[\mathrm{Ca}^{2+}\right]_{\mathrm{i},0}} \tag{B.2.2}$$

B.3 Corrected Niederer Equations

$$z_{\max} = \left(\frac{\alpha_0}{\left(\frac{Ca_{\mathrm{TRPN}\ 50}}{TRPN_{\mathrm{tot}}}\right)^{n_{\mathrm{Hill}}} - K_2}\right)\frac{1}{\alpha_{r1} + K_1 + \frac{\alpha_0}{\left(\frac{Ca_{\mathrm{TRPN}\ 50}}{\mathrm{TRPN}\ \mathrm{tot}}\right)^{n_{\mathrm{Hill}}}}} \tag{B.3.1}$$

C Cardiac Computational Modelling

The following report is for the coursework **DAT 390 Data Science Seminar** in NMBU. The report describes various cardiac physiological processes and sheds light on several efforts in cardiac modelling, which was a preparation of this Master's thesis.

Cardiac Computational Modelling

Ashesh Raj Gnawali
Data Science Section, Faculty of Science and Technology, Norwegian University of Life Sciences
ashesh.raj.gnawali@nmbu.no

ABSTRACT

In this paper, I review some of the relevant theories and literature in cardiac modelling and metamodelling. The review summarizes the cellular models and the whole organ models as well as highlights their applications in treatment predictions, drug discovery and testing. The article outlines the development of cardiac models and the intrinsic need of these models to shorten the diagnostic and therapeutic strategies in clinical applications along with cost savings.

1 INTRODUCTION

Cardiovascular diseases are the principal reason behind mortality claiming around 17.9 million life every year globally. [1]. Twenty-one percent of the Norwegian population lives with a type of cardiovascular disease or has a risk of developing the disease and 1.1 million of them use therapeutic drugs to treat or prevent the disease [2]. The diagnostic, as well as the therapeutic assessment of the patient, is still done by statistical studies which retrieves information from a large group of patients who have common pathologies [3]. Since every patient has a unique disease profile, the choice of treatment obtained from this model is thus not necessarily optimal.

Numerous cellular and mathematical models are developed for various parts of the heart. These models together form anatomically detailed whole organ models with applications in drug discovery and testing. They are also used to simulate cardiac defibrillators [4]. Simulating whole organ cardiac bio-physiology requires a large amount of medical imaging and diagnostic data in order to build a robust computational model. These along with revised cell, tissue and organ models as well as the significant advancement in computing devices has boosted the simulation of patient-specific models in determining the optimal treatment. Unlike the models that are based on statistics, these models are based on physics and physiology which facilitate uncovering diagnostic details that may have remained hidden thereby aiding in predicting patient-specific treatment[5].

Simulating whole organ cardiac biophysiology requires developing a complete model that simulates electrophysiology, blood flow mechanism, and muscular contraction [6]. The anatomical structure of the human heart is derived from in vivo medical images namely computed tomography (CT) and magnetic resonance (MRI). Improvement in mathematical modelling in conjunction with computational advancements enables us to simulate tightly coupled complex mechanisms that encompass feedback mechanisms to perform virtual heart studies [3]. These studies aid in treatment planning, patient risk assessment, and virtual heart clinical trials.

The mathematical models are represented by a system of ordinary differential equations which are governed by the parameters that describe the underlying processes in the heart. These models produce dynamic (spatio) temporal outputs. A patient-specific simulation is obtained by feeding patient-specific parameters to the model such that the results from the simulation agree with the clinical measurements. [5]. **Figure 1** illustrates how a computation model works in cardiology. Generally, a model is considered reliable if it replicates the validation data and can predict efficiently as well as successfully simulate the physiology of the individual considering the measurements that are provided[5]. Besides, this model can also be used for further enhancement of diagnosis as well as planning of treatment, and thereby contribute to increasing time and cost savings in clinical and preclinical trials[5].

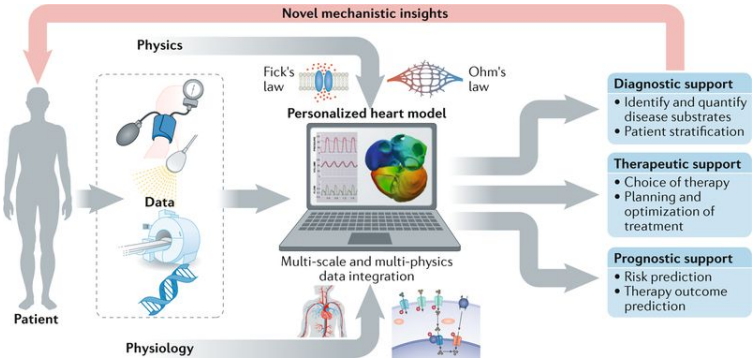


Figure 1: Working of a computational model in cardiology, adapted from Niederer et.al(2019)

A computational model of a human heart combines diagnostic data obtained from electrocardiography, genetics, blood-pressure, MRI and electrocardiography[5]. Such personalized data from a patient enables running model simulations as a 'virtual-patient' that provides insights in determining the disease optimal treatment[5]

.

Despite the emerging needs of precision medicine in cardiology, patient-specific modelling is a complex task that involves a team of experts from several disciplines. A team of clinical researchers provide the anatomical and physiological details of the heart and validate the model predictions[3]. Likewise, the professionals from fields like physics, mathematics, and engineering formulate equations for heart modelling and develop software to solve and simulate those models in normal as well as pathological heart conditions[3].

In this paper different types of cardiac models and their applications will be described. The paper further provides insights on metamodelling as a data-driven approach for analysing the behavior of such mechanistic models that can aid in the construction and validation of the models.

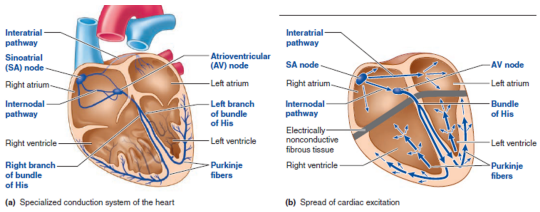


Figure 2: Electrical conduction system of the heart illustrating the flow of cardiac excitation, adapted from[8]

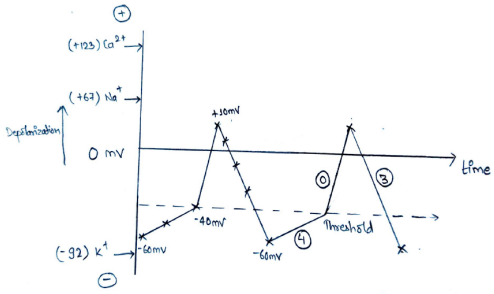


Figure 3: Graphical illustration of generation of action potential in pacemaker cells

2 THEORY

2.1 Electrical System of the Heart

The Right Atrium (RA), Right Ventricle (RV) along with the Left Atrium (LA) and Left Ventricle (LV) make up the four chambers of the heart. Blood flows through all the chambers and then is pumped to the rest of the body aided by the coordinated contraction and expansion of the heart. A cell is usually negatively charged. However, it can gain positive charge by depolarization which is a process shifting from a more negative membrane potential to a positive potential. Depolarization causes the heart muscles to contract[7].

Referring to **Figure 2**, in the sinoatrial (SA) node, there are unique cells that can depolarize themselves without any assistance. This ability of the cells is called automaticity. Once the cell is self-depolarized, it spreads out the depolarization waves to neighboring cells through the junctions that connect them. As the process continues, the neighboring cells also get depolarized. This depolarization initiated in the SA node is then transferred to the left atrium along the interatrial pathway called Bachmann's bundle. This depolarization in RA and LA occurs in a coordinated way. The internodal tracks guide the signal to the atrio-ventricular (AV) node which is the sole electrical connection between the atria and the ventricles. The conduction in the AV node is slow creating a delay in contraction between atria and ventricles by one-tenth of a second enabling the blood circulation through the atria to ventricles and then throughout the heart in a coordinated way[8].

The signal from the SA node is carried by the Bundle of His which is breaks into right and left bundle. The latter additionally divides into left posterior fascicle and left anterior fascicle that reaches down the septum and to the tip of ventricular chamber and back to the atria. Small fibers called Purkinje fibers emerge from the bundle of His and spreads the electrical signal in all directions[7].

2.2 Action potential in pacemaker cells

Pacemaker cells having the property of automaticity are responsible for the beating of the heart in a certain rhythm and a certain pace. There are three groups of pacemaker cells in the heart, each present in the SA node, AV node, and on Bundle of His and Purkinje fibers. Each of these sets of pacemaker cells are included in the electrical conduction system associated with the heart and are responsible for the appropriate pacing of the heart[9].

Calcium, Potassium, and Sodium ions move in and out of the cells which help us determine the voltage of a cell. For instance, if calcium is the only ion moving in and out of a cell, it would have a cell potential of 123mV. This gives the information about the cell potential if calcium was the only ion that could permeate the cell. Likewise, if the sodium ion and potassium ion were only the ions that could permeate the cell, the cell membrane potential would be 67mV and -92mV which is their resting potential respectively. In reality, the cells are permeable to multiple ions and not just permeable to a single ion. Thus, depending on how permeable a cell is to different ions we get the information about the membrane potential[9].

Figure 3 shows a graphical illustration of the generation of the action potential in pacemaker cells. Let us assume that a pacemaker cell is predominantly permeable to sodium ions and say its voltage is -60mV. Now, the sodium ions from outside the cells start moving inside as the cell is permeable to sodium ions. This will start increasing the cell potential and would eventually be close to 67mV which is the resting potential of sodium ion. But something interesting happens in between the process, as the cell potential increases it will hit a threshold at -40mV. This threshold is for a new type of ion which leads to the opening of voltage-gated calcium channels. Now, the calcium ions start pouring into the cell and since it is the ion, which the cell is dominantly permeable, the cell membrane potential would rise to the resting potential of the calcium ion which is even larger than that of the sodium ion. As the membrane potential increases quickly towards a positive value, and let's say it reaches 10mV, the voltage-gated calcium channels are now shut, and the potassium channels open, making it the dominant permeable ion. The potassium ions will escape from the cell following the direction of the concentration gradient, reducing the cell potential. Thus, the membrane potential decreases, reaches -60mV and stops. It does not decrease further because the potassium gated channels are shut at this point. Here, we have only the sodium ions entering the cell just like when the process started. Thus, this whole process happens each time the heart beats.

This process is divided into 3 phases which are Phase 4, Phase 0, and Phase 3. The action potential at Phase 0 in pacemaker cells grows slower than that of the action potential of the muscle cells and is hence referred to as "slower action potential". Phase 0 and 4 indicate the slow depolarization while phase 3 depicts repolarization. Repolarization can be described as the reduction of the cell membrane potential back to its most negative voltage just after depolarization[9].

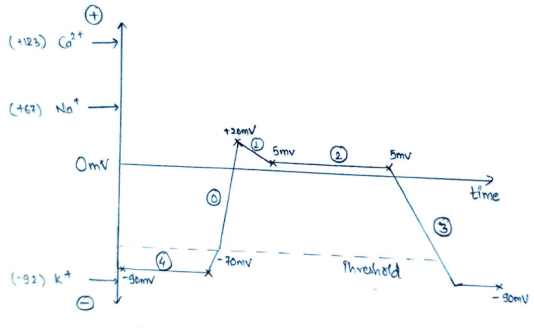


Figure 4: Graphical illustration of generation of action potential in cardiac myocytes

2.3 Action Potential in Cardiac Myocytes

The contraction of the heart is controlled by the heart muscle cells called cardiac myocytes. The trigger for the contraction of the heart is calcium. Similar to the action potential in pacemaker cells, the main ions that influence the heart muscle cells are sodium, calcium, and potassium.

Referring to **Figure 4** ,let us assume that a cell is dominated by potassium ions and the ions are leaving the cell causing a negative membrane potential at around -90mV. Now, as a neighboring cell undergoes depolarization, this depolarization wave also transmits to this cell and a little bit of sodium and calcium ions start to leak into the cell. This results in the membrane potential becoming a little more positive. Now, it is -70mV up from -90mV.

At this point, new channels start opening and these channels are sodium channels. As the sodium ions start flowing into the cell, they drive the membrane potential to a very positive value somewhere around 67mV. These voltage-gated sodium channels close just as quickly as they opened. Thus, as the cell membrane potential is quite positive, we could say that the channels have caused a depolarization. No more sodium ions are flowing in but the potassium ions are still leaving the cell. At this point, new voltage-gated potassium channels open because of depolarization. Now, that the cell potential is positive (about 20mV) and the potassium voltage-gated channels have opened, the membrane potential will go down at around 5mV, and if it continued this way it would reach -90mV. However, an interesting development takes place at this point, the calcium ions start leaking into the cell. As both events are happening simultaneously, both potassium ions leaving and calcium ions entering the cell, we get a flat line depicting that the membrane potential stays somewhat the same. The voltage-gated calcium channels close just as suddenly as they opened, blocking the flow of calcium ions. As this flow of calcium ions was only responsible for keeping the membrane potential somewhat constant, now due to the leaving of the potassium ions the membrane potential decreases and goes back to somewhere around -90mV. These voltage-gated potassium channels also close at this point and we finally get back to our initial state waiting for another cycle[10].

All these processes are named with different stages. State 4 describes the baseline negative state when the muscle cell is relaxed. The action potential at -70mV is called a threshold and when it gets to 20mV we call it state 0. State-1 is the point where just the voltage-gated potassium channels open. State-2 is where they're balanced with calcium channels and state-3 is when only the voltage-gated potassium channels are open. And then the state-4 is following again in the next cycle[10].

As the state 0 occurs quickly compared to how an action potential develops in a pacemaker cell due to the quick voltage-gated sodium channels, this action potential is also called a fast action potential.

2.4 Sodium Calcium Exchanger

The intracellular calcium levels in cardiac cells range from $10^{-7}$to 10^{-5}M whereas the extracellular concentration is about 2mM. Thus, in order to maintain this concentration, it is necessary to remove the calcium that entered the cell during the action potential to avoid the accumulation of calcium which can cause cellular dysfunction. This mechanism of removing the calcium from the cell is called sodium-calcium exchanger. As sodium and calcium can move freely in the sarcolemma(cell membrane of a muscle fiber), in the process, for each calcium ion three sodium ions are exchanged releasing a small electric potential[11].

The movement of ions(in or out) is guided by the cell membrane potential. In the case of depolarization(positive membrane potential), the sodium ion leaves the cell as the calcium ion enters the cell. In contrast, during repolarization, the calcium ions are carried out of the cell by the exchanger while sodium ions enter the cell[11].

2.5 The Electrocardiogram (ECG)

The ECG **(Figure 5)** illustrates the electrical activity of the heart. It provides information on the alteration in voltage with time. The recording of the ECG by electrodes, involves only the activity present in the body fluids which subsequently reaches the body surface through the cardiac impulse.

An ECG has three waveforms: The P wave, QRS complex and the T wave.The SA node sets off the waves of depolarization which move across the atria. This signal travels further to the AV node. The P-wave represents the atrial depolarization resulting in a subsequent contraction of the atria. The conduction in the AV node is slow because the diameter of the AV nodal cells is small. Another reason for the slow conduction is the presence of calcium channels in the AV node which are naturally slow moving compared to that of sodium channels that are specific to the conduction system in the ventricles. This time delay is of great importance since it allows the ventricles to fill blood as well as provide time for contraction and relaxation of the atria.

As the SA node fires, spreading waves of depolarization across the atria, the P-wave is obtained. The atrial contraction is followed immediately after 100ms of the P-wave. The time taken by the signals to travel from the SA node to the AV node is represented by P-Q. The depolarization of the interventricular septum is described by the Q-wave. The PR segment indicates the time taken from the beginning of atrial depolarization until ventricular depolarization.

The fast-acting sodium channels are responsible for quick depolarization in the Bundle of His and this wave of depolarization is further carried down to the branches and Purkinje fibers. This leads to the depolarization of ventricular cells. All of this process is seen as a QRS complex in ECG and it describes the ventricular

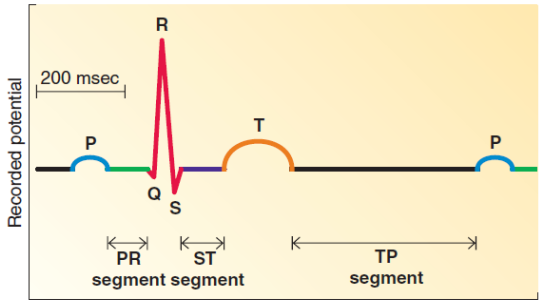


Figure 5: Electrocardiogram waveforms , adapted from[8]

depolarization. The time taken by the ventricles in contraction and pumping of the blood is represented by the S-T segment whereas the T-wave reflects the repolarization of ventricles.The ST segment represents a period where there is no net current,i.e., there aren't any large electrical vectors in any directions. As repolarization occurs slower than depolarization, the T wave is flatter and longer than the QRS complex. The T wave reflects ventricular repolarization. Likewise, the TP segment explains the time during which the ventricles are relaxing and filling.

2.6 Cellular Models

Cell type models combine measurements and models of protein functions into a physically and physiologically coerce framework [5]. In the past 30 years modelling of mammalian cardiac myocytes has shifted from combination of large amounts of data from multiple species to data from specific species [12].

The newer cellular models include many sub-cellular organelles like diadic space, sacroplasmic reticulum, mitochondria or simply focus on individual ion transporters [13].The primary function of the sarcoplasmic reticulum is the regulation of calcium ion concentration in the cytoplasm and controlling muscular contraction and relaxation. The calcium found in the human body (calcium carbonate and calcium phosphate) helps to make teeth and bones. Thus, higher concentration of calcium in the cells leads to the hardening of the cellular structures which eventually causes cell death. The cytoplasmic volume in the cellular region, where the cell membrane and a cellular organelle is in close vicinity, generally 10-12nm, is referred to as a dyadic space. The space aids in ionic signaling. Mitochondria are responsible for the production of ATP (adenosine triphosphate) which aids in powering the metabolic processes in the cell. Oxidative phosphorylation is a process that converts the consumed food into a form of energy that a cell can utilize. Mitochondria also plays a role in apoptosis (cell death) in deciding which of the cells to destroy (aged cells/broken cells) by releasing cytochrome C. This release of cytochrome C activates a major enzyme "caspase" that is responsible for destroying cells.

Nobel et.al have developed a a model consisting of 26 ODE's that describe the rate at which the dominant cellular ion K^+, Na^+, Ca^{2+} concentration varies[14]. Also, models with K^+, Na^+, Ca^{2+}, including physiological processes like pH regulation and Ca^{2+} homeostasis have been developed [15].Calcium homeostasis refers to the maintenance of a consistent concentration of calcium ions within the extracellular fluid. The extracellular calcium ions are kept constant by maintaining a constant Ca^{2+} concentration in the plasma. Keeping plasma Ca^{2+} concentration is important for cell adhesion, cardiac contractility, muscle contraction, and blood clotting[16].

The behavior of ion channels, transporters, pumps and buffers can be uncovered by cellular models. These properties are evaluated by mathematical models and are used in examining the changes in channel function due to mutation, drugs or any other physiological changes [5]. Biophysical cardiac cell models are used effectively to simulate Ca^{2+} dynamics, electrophysiology in the sinoatrial node as well as in myocytes in atria and ventricles [5].

2.6.1 Effects of protein Mutation. Protein mutations in cells are often associated with the risk of arrhythmia. To link the causative relationship between the disease and mutation it is essential to find the mechanistic relation between mutation and change in protein function [5].

The successful identification of a channel mutated protein, and linking it to a specific protein function does not guarantee that a particular mutation is the cause of a disease. [5]. Thus, linking the change in protein function to arrhythmia and ECG morphologies is challenging[5].

In order to overcome it, the biophysical models capable of understanding complex cellular electrophysiology and models that represent fluctuation in kinetic channels are combined together, the result of which is sufficient to demonstrate that certain channel mutations can explain observed cellular and clinical phenotypes [5]. Under further advancement, the models are capable enough to mechanistically explain channel mutations and specific clinical manifestation of channelopathies resulted by multiple channel mutation or genetic mosaicism [17]. Mosaicism is existence of cells with different genetic component than the other cells in the body. It occurs due to mutation in early phase of development. Recently, models to predict mutation effect on sarcomeric protein and change in tension development have also been developed [5].

Current models enable us to express gene variation which is fundamental for heart condition and diseases such as heart failure[18]. The models are now advanced enough to reach the genetic level by constructing mutation effects which are originally indicated by alternation in protein function. For instance, during the formulation and validation of the models of wild-type and mutant Na^+ channel, a three amino acid ΔKPQ mutation was simulated that affected channel inactivation which was also connected to a long QT[19]. It was observed that the failure of inactivation resulted in mutant channel reopening from inactivated state causing a continuous Na^+ flow in the mutant cell. This led to longer repolarization period and early after-depolarization which is also the same in bradycardia-related arrhythmogenic episodes in LQT3 patients while they are asleep or relaxed[18].

2.6.2 Electrophysiological and Mechanical Models. The patch-clamp method is an experimental method that is used to measure biological as well as physical attributes of cardiac muscle cells. This method aids the development of computational models for each myocyte.[3]. These models are based on the mathematical formulation of Hodgkin-Huxley which reveals that there exists a system of equations that describe the cellular action potential provided that

these equations model the kinetics of individual ion channels,pumps and exchangers including their electrical interactions[20]. Over the years, ventricular models, atrial myocytes models and Purkinge fiber models which describe the cardiac electrical activity have been developed[3].

The excitation-contraction coupling must be considered in 3D cardiac computational models that focus on electromechanical simulations[3]. Whenever the calcium inside a cell increases, a contraction in the cardiac tissue is initiated that induces an electrical activation which further affects the action potential through stretch activated currents[21].

Assessing and Predicting Drug Action

Models that simulate the perturbing protein activity and it's formation are applicable on a large scale to predict the drug action[18]. This is because receptor, channel, transporter, and enzyme proteins when acted on by the drugs, affect the channels differently[18]. Present-day models are now able to detect the response of the ion channels for the applied drug through the dose-response relationship which gives an estimation of functioning channels, and this model further can be incorporated in whole-cell models[5]. Models can predict the drug-drug interaction, an underlying mechanism such as Na/Ca exchange, and how a drug can have multiple actions. This is because the drugs that act on only one receptor are rare, which is particularly true for Na/Ca exchange[5].

Simulating drugs for multiple actions comes with a bright side that open possibilities for the discovery of antiarrhythmic drugs [18]. These simulations can also predict the efficiency of the drug or risk associated with them and effects of a certain compound and it's corresponding metabolites on human electrophysiology [5]. Approaches such as Comprehensive In vitro Proarrhythmia(CiPA) [5] are now proven effective in screening cardiac toxicity in humans.This method is used in the analysis of cardiac toxicity to evaluate the risk of Torsade de Pointes(TdP) which is recognized by the elongation of the QT interval. TdP leads to abnormal heart rhythm and can even cause sudden death. The goal of CiPA is to develop a model of ventricular cells to examine cardiac response and cardiac toxicity risk and formulate a metric for TdP risk assessment. With the advancement in these approaches, cardiac model simulations can predict the heart's response to compounds or drugs [21].

2.7 Whole Organ Models

Arrhythmias are highly dependent on cellular and molecular mechanisms, and are fatal due to their actions at organ level[22].There has been a lot of debate among the researchers on deciding the best approach for incorporating cellular models into whole organ models, whether it should be a bottom-up, top-down or a combination of these[23][24]. A general agreement is that it should be "middle-out" which means that we should start modelling when large biological data is available and then only jump or fall back to other levels[18].

In order to reconstruct the complete electrical and mechanical behavior of the whole heart, anatomically detailed models of ventricles with fibre orientations and sheet structures are combined with cellular models[18].Cardiac modelling has benefited from data rich cellular levels and the 3D geometry modelling of the whole

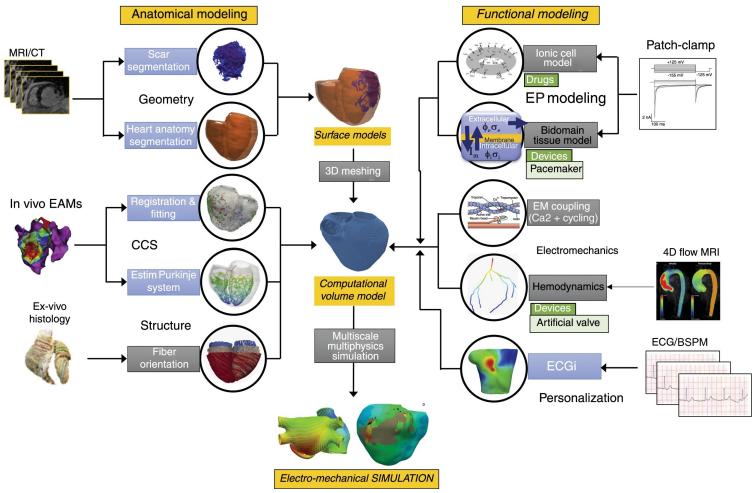


Figure 6: Schematic illustration for building a 3D cardiac computational model, adapted from Bragard et.al(2020)

The process is initiated by constructing a 3D cardiac geometry of the heart from MRI and CT images, followed by the generation of cardiac conduction system, myocardial structure and biophysical modelling leading to the multiscale multihphysics simulation. Patient specific models can be designed by utilizing the ECG of the patient. Pointers indicate the stream of clinical as well as biologial data into the ultimate computational model and electromechanical simulations. The blue boxes indicate the methods that enable us to specify patient specific personalized properties in the models whereas the grey boxes represent population based properties[3].

organ[18] The geometrical modelling utilizes MRI and CT images, from which the volume of various internal structures of the heart are extracted.

The images acquired from MR or CT are utilized in building 3D heart models[3].High resolution ex-vivo images provide more precise anatomical reconstruction of the heart as compared to in-vivo. Although ex-vivo images are much more informative, during the tissue preparation while acquiring histologial slices, the shape,size and volume of the cardiac structures may be altered because of distortion as a consequence of the slicing process in histological sections[6]. In the absence of anatomical data and when the simplicity of the model is of main priority rather than building a realistic model, population based data are found to be useful. These data are generally the measurements of cardiac wall thickness or the chamber volumes. [6].

As there is a large number of cells in the heart, it is difficult to model tissues by creating models for each and every cell. Due to this the heart is visualized in a such a way that it is composed of two continuous spaces which can be broken down into intracellular and extracellular domains. The geometries associated with two domains are way too complex and cannot be portrayed correctly.The biodomain concept assumes that the geometry of the two domains are overlapping. Hence, accepting that both the domains fill the heart muscle volume completely, the biodomain model does not primarily take the accuracy into account. A complex biodomain

model is reduced into a simple monodomain model by making non-physiological assumptions such as equal anisotropy ratios. Bidomain and monodomain models when coupled with cellular models form multiscale models that simulate the tissue propagation[25].

Also, it is very important to incorporate characteristics such as cardiac fiber orientation and pathologies that influence myocardial structure in the whole organ models. Thus, mathematical models that explain the property of electrical excitation or mechanical deformation to it's neighbors is essential[3]. Finally, these models also include a cardiac conduction system which is modelled to work at a functional level or are derived from ex-vivo images[26]. The pipeline for building a 3D cardiac computational model is presented in **Figure 6** One of the advanced models that has been used recently includes the Purkinjie network using inverse estimation that is based on electro-anatomical based data[27].

3 SOME CLINICAL APPLICATIONS

Arrythmia is a pathology denoted by the irregular heart rhythms which can be tachycardia (too fast) and bradycardia (too slow). The main cause of arrhythmia is the coronary artery disease which leads to myocardial ischemia (insufficient blood flow to provide adequate oxygenation). Having less oxygen in cardiac myocytes causes them to depolarize resulting in altered impulse formation or modified impulse condition. Altered impulse formation triggers partial or complete blockage of electrical conduction in the heart. This not only affects the automaticity of the pacemaker cells but also initiates action potential on different locales of the heart other than the SA node. Likewise, antiarrhythmic drugs along with the electrolyte(Ca^+, Na^+) imbalances can cause arrhythmias[11].

Computational models ease understanding the relationship between atrial activation patterns and the characteristics of electrograms recorded by catheters. These models also predict improved electrogram biomarkers that describe atrial fibrillation(AF) dynamics and it's drivers. Computational models also aid in studying electrophysiological differences between various patients and also in patient specific remodelling to AF initiation, and maintenance which guide treatment[3].

Cardiac ablation is a treatment procedure that is used for restoring the optimal heart rhythm. In this process, the tissue in the heart that causes abnormal heart rhythm is scarred. Catheters are inserted in the groin region through arteries and veins which are threaded to the heart. This helps in supplying energy (heat or cold) to modify the tissue in the heart that causes arrhythmia[28]. However, this method of treatment depends on several elements , for instance, the AF type which can be paroxysmal(occurs occasionally and stops spontaneously), persistent or permanent as well as factors like age, atrium size, use of antiarrhythmic drugs and the frequency that the patient has undergone ablation procedure. Thus, it is very important to select correct patients and identify the required size of generated lesions in order to prevent atrium impairment[3].

Boyle et.al carried out an experiment with 10 patients who had persistent AF symptoms and atrial fibrosis for targeted optimum ablation. Gadolinium is one of the contrast agents that help enhance the MR images for increased visibility of internal structures of the body. The authors utilized each patients late gadolinium enhancement MR imaging scans and created personalized computational atria models thereby locating the regions maintaining the AF as well as the sites that might become the possible sources of the AF in future[29].

The computational models are also used to enhance the technology behind pacemakers and defibrillators. It is now proven that fiber orientations and membrane kinetics are important to predict the result of defibrillation. However, the most precisely optimized personal electrical device is still a work in progress. The simulations show that the defibrillation success rate increases with increasing shock strength and traces a typical logistic curve[3].

4 METAMODELLING

A metamodel or surrogate model is a model of a model. An original model which is distinguished by high complexity and high computational cost(e.g. a computational model of the heart) is replaced by a metamodel. The input-output data from a complex model after a large number of simulations is used to calibrate a metamodel, which is a data-driven approximation of the original model based on e.g. regression. In cardiac modelling, the simulations are set up by varying the input parameters of the model and (spatio-)temporal outputs are generated from the simulations that resemble real quantities that can be measured on the myocytes or whole heart, and that represent the cardiac functionality.The metamodelling can be of either the classical or the inverse type. A classical metamodel predicts the output by taking the input parameters as functions whereas in inverse metamodelling, the input parameters of the mathematical models are predicted from the output of a complex model. The metamodelling can be carried out using various supervised and unsupervised machine learning techniques.

Metamodelling is used in estimation of model sensitivity (the degree of dependency of the model output on variations in the different input parameters), visualization and analysing the parameters as well the output space[30]. Metamodels also help identifying the hidden patterns of co-variance in the data[31]. The aim is to analyse and overview the model behaviour under different input conditions, as well as estimation of the parameter values that allow the model to replicate measured data (parameter fitting). The latter is crucial for practical use of the computational models, e.g. in the development of patient-specific models. Metamodelling can also be used for model reduction, by identification of redundant model components (e.g. by sensitivity analysis, where model components controlled by parameters which the model is insensitive to are removed). Moreover, metamodelling is widely used for reducing computational demand, by using the less computationally demanding metamodels as surrogates for the original, more complex models. Metamodelling has a variety of application areas, including fire stations, hospitals, manufacturing systems, risk assessment, military, and manufacturing systems[32].

5 DISCUSSION

The massive advancements in technology and increased research have now opened new possibilities for the application of biophysical models in clinical use. These advancements include high computing infrastructures, open-access medical databases, and increased development in open-source software. Computational models of the heart encompass the majority of the biophysical complexities of an

individuals heart and therefore is an aspect to be understood in designing medical devices as well as in therapies. Simulating models with virtual patients and therapies can not only save time but also minimize the developmental cost of innovation in pharmaceutical industries.

Although simulation models come up with several advantages, these models are still in the start-up phase for their application in diagnosis and treatment. We only discussed models that predict the short-term responses to treatments like termination of arrhythmia and prolonging action potential. However, the cardiologist's primary interest is in knowing how the patient responds to the given treatment in the long run rather than the patient's immediate response. Thus, it is essential to predict the remodeling of the heart after treatment such that we can also make long-term estimations associated with the clinical outcomes which will tremendously enhance the cardiological simulations. Cardiac remodelling is a technique that maintains the cardiac output during conditions such as abnormal loading and depressed contractility.

Several factors are responsible for restricting the application of biophysical models into a more common and efficient clinical practice. To predict realistic clinical results, biophysical models need pre-existing clinical information and a lot of data. In most of the cases, the data required is not available for most of the clinical applications as well as cannot be measured accurately in most of the clinical context. Similarly, most of the data in hand are not suited for model design, and data curation and annotation is a long, tedious, and time-consuming process. Thus, improvisation in existing methods which consequently lead to the translation of preclinical measurements into the patient-specific model is required. This will enable us to create a more robust model that fits with a number of pathologies and provides reliable predictions when making clinical decisions. Metamodelling with various machine learning approaches can be used to emulate dynamic biological models that aid in sensitivity analysis and variable selection.

To tackle these challenges and broaden the application of simulation models in cardiology, technological advancements in the clinical field to ensure proper harmonized, anonymized and standardized data is a must. Biomedical engineers, therefore, should be incorporated in the clinical units. The availability of high quality and relevant patient-specific data will add value to the simulations. The development of heart models demands a multi-disciplinary approach involving expertise in numerical analysis, computer science, cardiac electrophysiology, and image processing.

6 CONCLUSION

The computational models are still in their infancy period, but are much more economic and efficient compared to that of the classical statistical studies in treatment prediction and therapeutic assessment. As long as the clinicians are ready to embrace the support from the advancements of the models, the future of computational tools in cardiology is bright and holds more interesting human-related tasks.

REFERENCES

[1] W. H. Organization, ""cardiovascular diseases (cvd's) key facts"," "May" "2017".
[2] N. I. of Public Health, ""cardiovascular diseases"," "January" "2020(updated)".
[3] J. R. Bragard, O. Camara, B. Echebarria, L. G. Giorda, E. Pueyo, J. Saiz, R. Sebastián, E. Soudah, and M. Vázquez, "Cardiac computational modelling," *Revista Española de Cardiología (English Edition)*, 2020.
[4] D. Noble, "Modelling the heart: insights, failures and progress," *Bioessays*, vol. 24, no. 12, pp. 1155–1163, 2002.
[5] S. A. Niederer, J. Lumens, and N. A. Trayanova, "Computational models in cardiology," *Nature Reviews Cardiology*, vol. 16, no. 2, pp. 100–111, 2019.
[6] A. Lopez-Perez, R. Sebastian, and J. M. Ferrero, "Three-dimensional cardiac computational modelling: methods, features and applications," *Biomedical engineering online*, vol. 14, no. 1, p. 35, 2015.
[7] K. Academy, "Electrical system of the heart | circulatory system physiology | nclex-rn | khan academy." https://www.youtube.com/watch?v=7K2icszdxQc&ab_channel=khanacademymedicine, Oct 2012.
[8] L. Sherwood, *Human physiology: from cells to systems*. Cengage learning, 2015.
[9] K. Academy, "Action potentials in pacemaker cells | circulatory system physiology | nclex-rn | khan academy." https://www.youtube.com/watch?v=OQpFFiLdE0E&ab_channel=khanacademymedicine, Oct 2012.
[10] K. Academy, "Action potentials in cardiac myocytes | circulatory system physiology | nclex-rn | khan academy." https://www.youtube.com/watch?v=rIVCuC-Etc0&ab_channel=khanacademymedicine, Oct 2012.
[11] D. P. Johnson, "Cv physiology." https://www.cvphysiology.com/Cardiac%20Function/CF023, March 2016.
[12] M. Fink, S. A. Niederer, E. M. Cherry, F. H. Fenton, J. T. Koivumäki, G. Seemann, R. Thul, H. Zhang, F. B. Sachse, D. Beard, *et al.*, "Cardiac cell modelling: observations from the heart of the cardiac physiome project," *Progress in biophysics and molecular biology*, vol. 104, no. 1-3, pp. 2–21, 2011.
[13] N. Smith, P. Mulquiney, M. P. Nash, C. P. Bradley, D. Nickerson, and P. Hunter, "Mathematical modelling of the heart: cell to organ," *Chaos, Solitons & Fractals*, vol. 13, no. 8, pp. 1613–1621, 2002.
[14] D. Noble, A. Varghese, P. Kohl, and P. Noble, "Improved guinea-pig ventricular cell model incorporating a diadic space, ikr and iks, and length-and tension-dependent processes.," *The Canadian journal of cardiology*, vol. 14, no. 1, pp. 123–134, 1998.
[15] J. Heijman, P. G. Volders, R. L. Westra, and Y. Rudy, "Local control of β-adrenergic stimulation: effects on ventricular myocyte electrophysiology and ca2+-transient," *Journal of molecular and cellular cardiology*, vol. 50, no. 5, pp. 863–871, 2011.
[16] J. J. Feher, *Quantitative human physiology: an introduction*. Academic press, 2017.
[17] J. R. Priest, C. Gawad, K. M. Kahlig, K. Y. Joseph, T. O'Hara, P. M. Boyle, S. Rajamani, M. J. Clark, S. T. Garcia, S. Ceresnak, *et al.*, "Early somatic mosaicism is a rare cause of long-qt syndrome," *Proceedings of the National Academy of Sciences*, vol. 113, no. 41, pp. 11555–11560, 2016.
[18] D. Noble, "Modeling the heart–from genes to cells to the whole organ," *Science*, vol. 295, no. 5560, pp. 1678–1682, 2002.
[19] C. E. Clancy and Y. Rudy, "Linking a genetic defect to its cellular phenotype in a cardiac arrhythmia," *Nature*, vol. 400, no. 6744, pp. 566–569, 1999.
[20] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.
[21] P. J. Hunter, A. D. McCulloch, and H. Ter Keurs, "Modelling the mechanical properties of cardiac muscle," *Progress in biophysics and molecular biology*, vol. 69, no. 2-3, pp. 289–331, 1998.
[22] D. Noble, "Modeling the heart," *Physiology*, vol. 19, no. 4, pp. 191–197, 2004.
[23] G. R. Bock and J. A. Goode, *Complexity in biological information processing*, vol. 239. John Wiley & Sons, 2001.
[24] G. R. Bock and J. A. Goode, *'In silico'simulation of biological processes*, vol. 247. John Wiley & Sons, 2003.
[25] M. Potse, B. Dubé, J. Richer, A. Vinet, and R. M. Gulrajani, "A comparison of monodomain and bidomain reaction-diffusion models for action potential propagation in the human heart," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 12, pp. 2425–2435, 2006.
[26] R. Sebastian, V. Zimmerman, D. Romero, and A. F. Frangi, "Construction of a computational anatomical model of the peripheral cardiac conduction system," *IEEE transactions on biomedical engineering*, vol. 58, no. 12, pp. 3479–3482, 2011.
[27] F. Barber, I. García-Fernández, M. Lozano, and R. Sebastian, "Automatic estimation of purkinje-myocardial junction hot-spots from noisy endocardial samples: A simulation study," *International journal for numerical methods in biomedical engineering*, vol. 34, no. 7, p. e2988, 2018.
[28] M. F. for Medical Education and Research, "Cardiac ablation." https://www.mayoclinic.org/tests-procedures/cardiac-ablation/about/pac-20384993.
[29] P. M. Boyle, T. Zghaib, S. Zahid, R. L. Ali, D. Deng, W. H. Franceschi, J. B. Hakim, M. J. Murphy, A. Prakosa, S. L. Zimmerman, *et al.*, "Computationally guided personalized targeted ablation of persistent atrial fibrillation," *Nature biomedical engineering*, vol. 3, no. 11, pp. 870–879, 2019.
[30] D. Gorissen, I. Couckuyt, E. Laermans, and T. Dhaene, "Multiobjective global surrogate modeling, dealing with the 5-percent problem," *Engineering with Computers*, vol. 26, no. 1, pp. 81–98, 2010.
[31] H. Martens, K. Tøndel, V. Tafintseva, A. Kohler, E. Plahte, J. O. Vik, A. B. Gjuvsland, and S. W. Omholt, "Pls-based multivariate metamodeling of dynamic systems," in *New Perspectives in Partial Least Squares and Related Methods*, pp. 3–30, Springer,

2013.
[32] R. A. Kilmer, A. E. Smith, and L. J. Shuman, "An emergency department simulation and a neural network metamodel," *Journal of the society for health systems*, vol. 5, no. 3, pp. 63–79, 1997.

D Explained Y Variance Plots

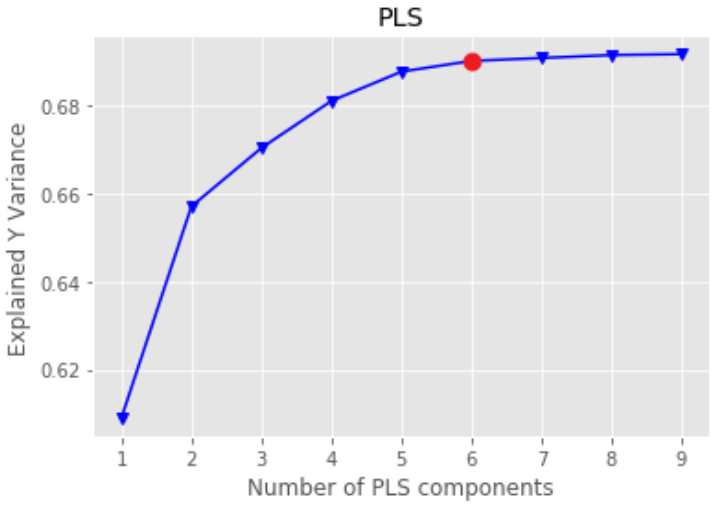


Figure D.1: Explained Y variance plot with increasing number of principal components for the classical metamodelling of the time series data.

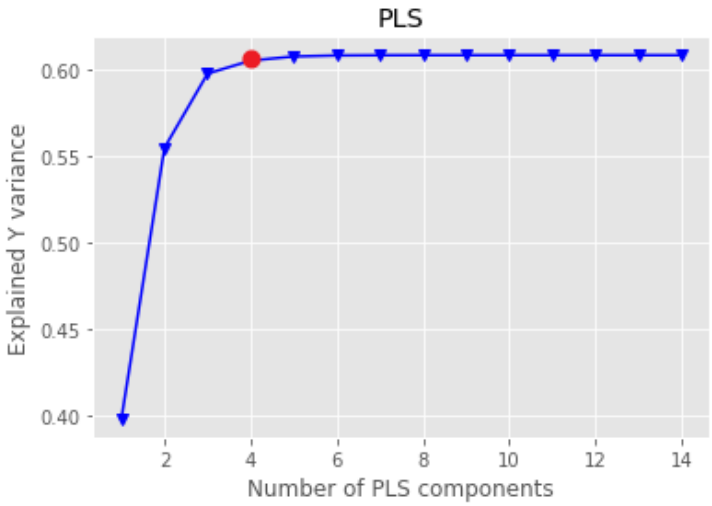


Figure D.2: Explained Y variance plot with increasing number of principal components for the classical metamodelling of the aggregated phenotypes.

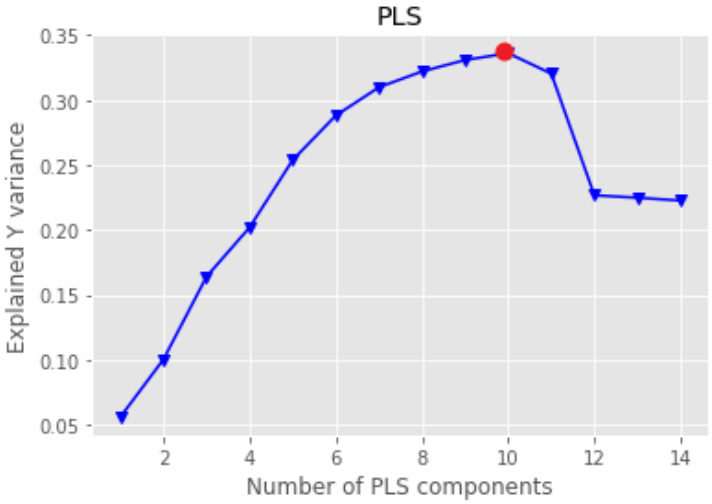


Figure D.3: Explained Y variance plot with increasing number of principal components for the inverse metamodelling of the time series data.

E Validation Accuracy Scores Table

Table 6: Validation accuracy scores achieved by various classification algorithms in different types of metamodelling using HC-PLSR

Type of Metamodelling \ Algorithm	LR	RF	DT	KNN	SVM
Classical Metamodelling	0.994	0.986	0.790	0.833	0.894
Classical Metamodelling of Aggregated Phenotypes	0.386	0.425	0.338	0.361	0.420
Inverse Metamodelling	0.998	0.996	0.995	0.968	0.972

F Network Architectures

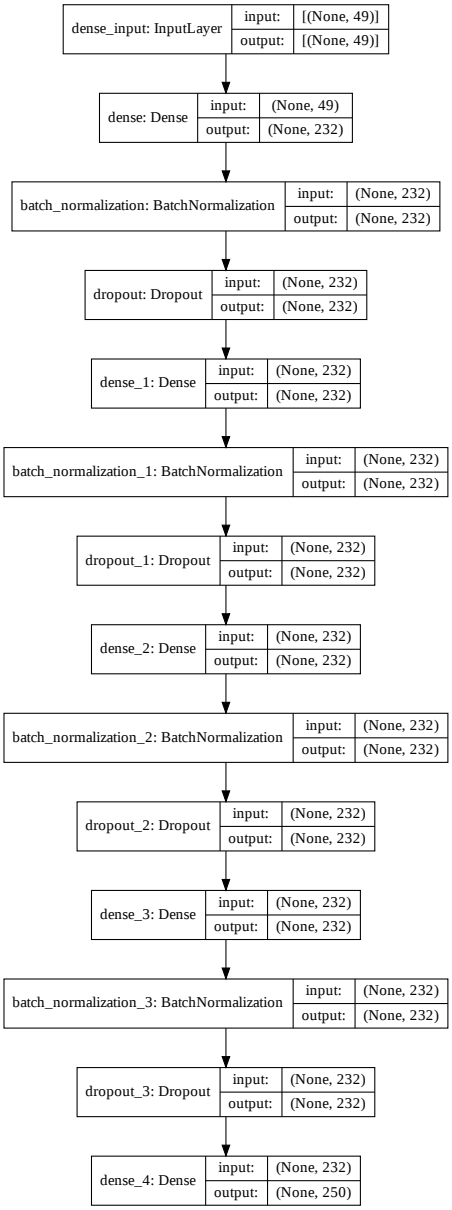


Figure F.1: Network architecture of the FFN model used in the classical metamodelling of the PHN model. Left box: Type of layer. **middle:** shape of input/output data.

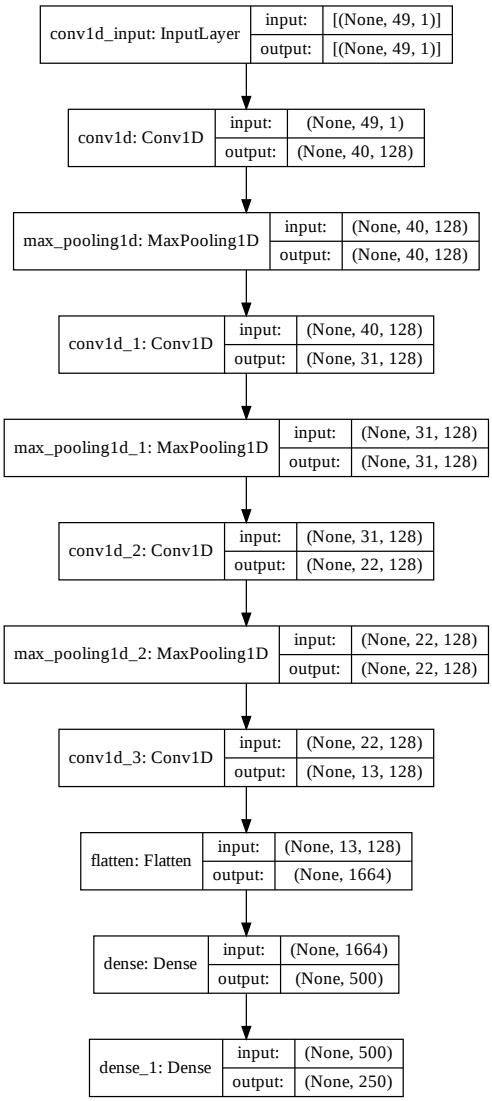


Figure F.2: Network architecture of the CNN model used in the classical metamodelling of the PHN model. Left box: Type of layer. **middle:** shape of input/output data.

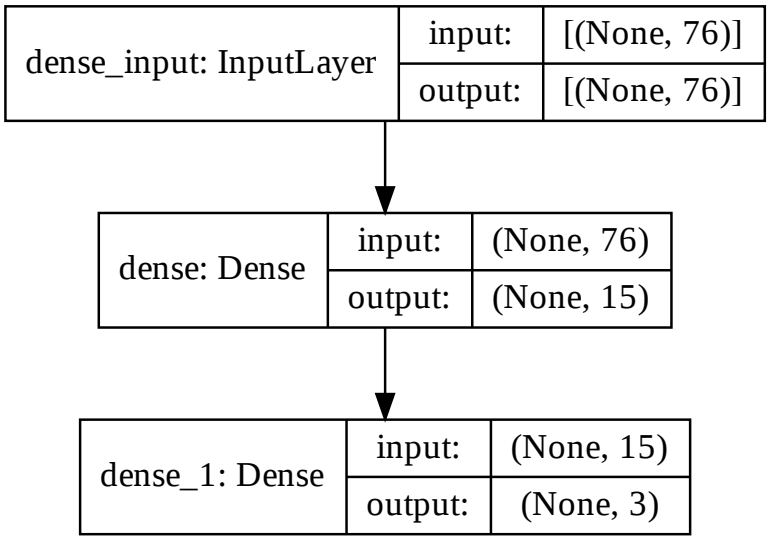


Figure F.3: Network architecture of the FFN model used in the classical metamodelling of the aggregated phenotypes of the PHN model. Left box: Type of layer. **middle:** shape of input/output data.

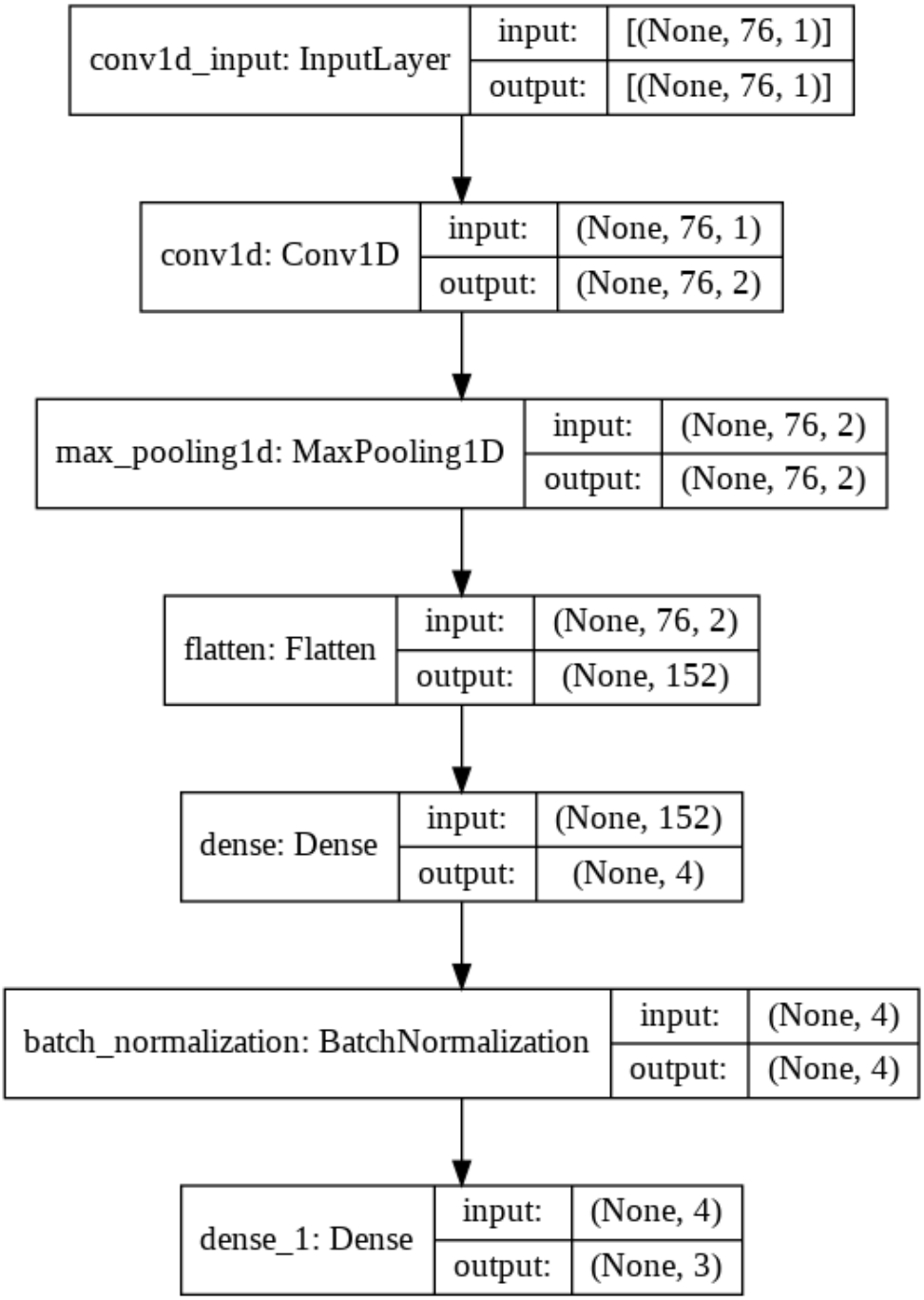


Figure F.4: Network architecture of the CNN model used in the classical metamodelling of the aggregated phenotypes of the PHN model. Left box: Type of layer. **middle:** shape of input/output data.

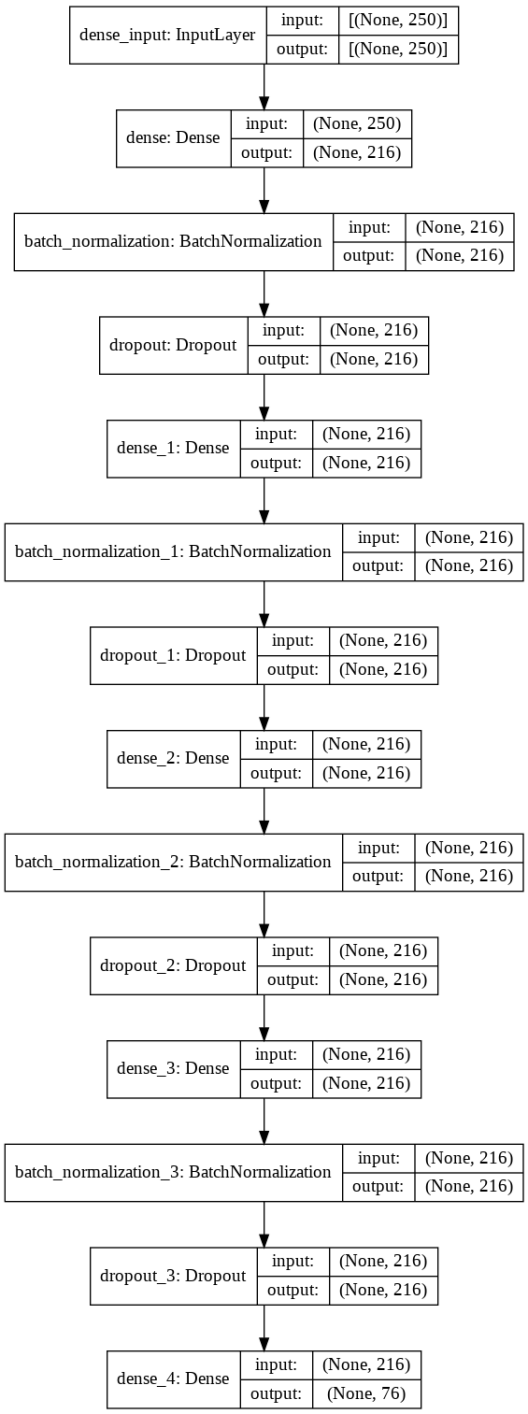


Figure F.5: Network architecture of the FFN model used in the inverse metamodelling of the PHN model. Left box: Type of layer. **middle:** shape of input/output data.

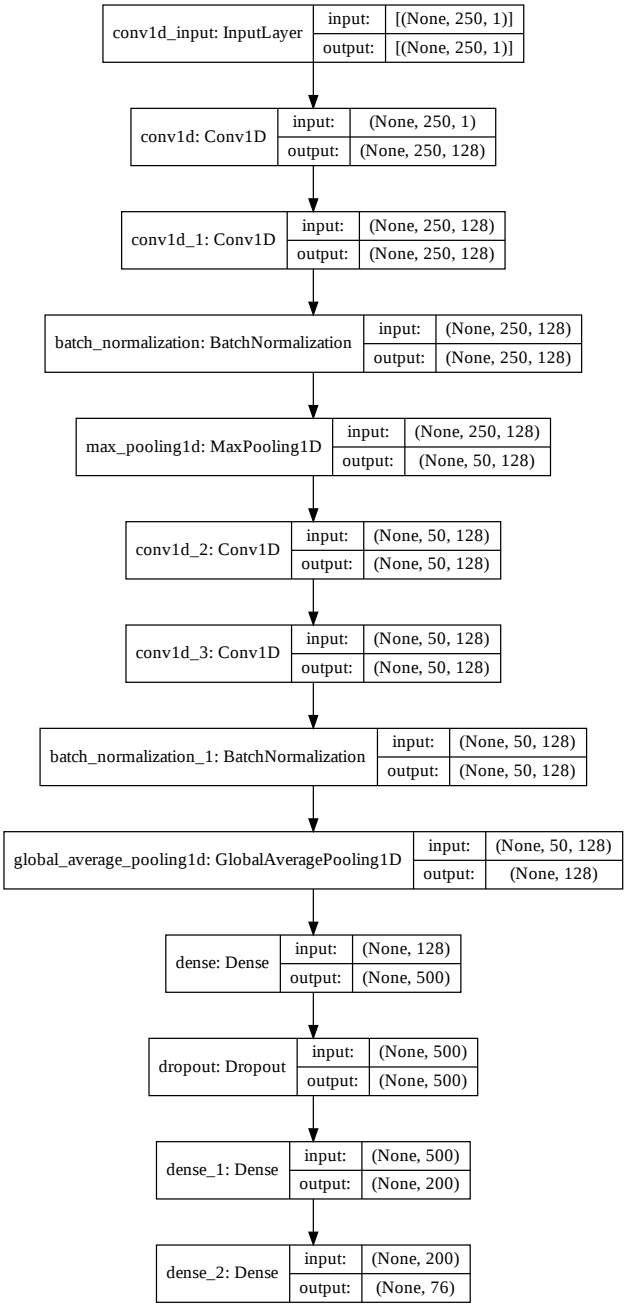


Figure F.6: Network architecture of the CNN model used in the inverse metamodelling of the PHN model. Left box: Type of layer. **middle:** shape of input/output data.

G Training Plots

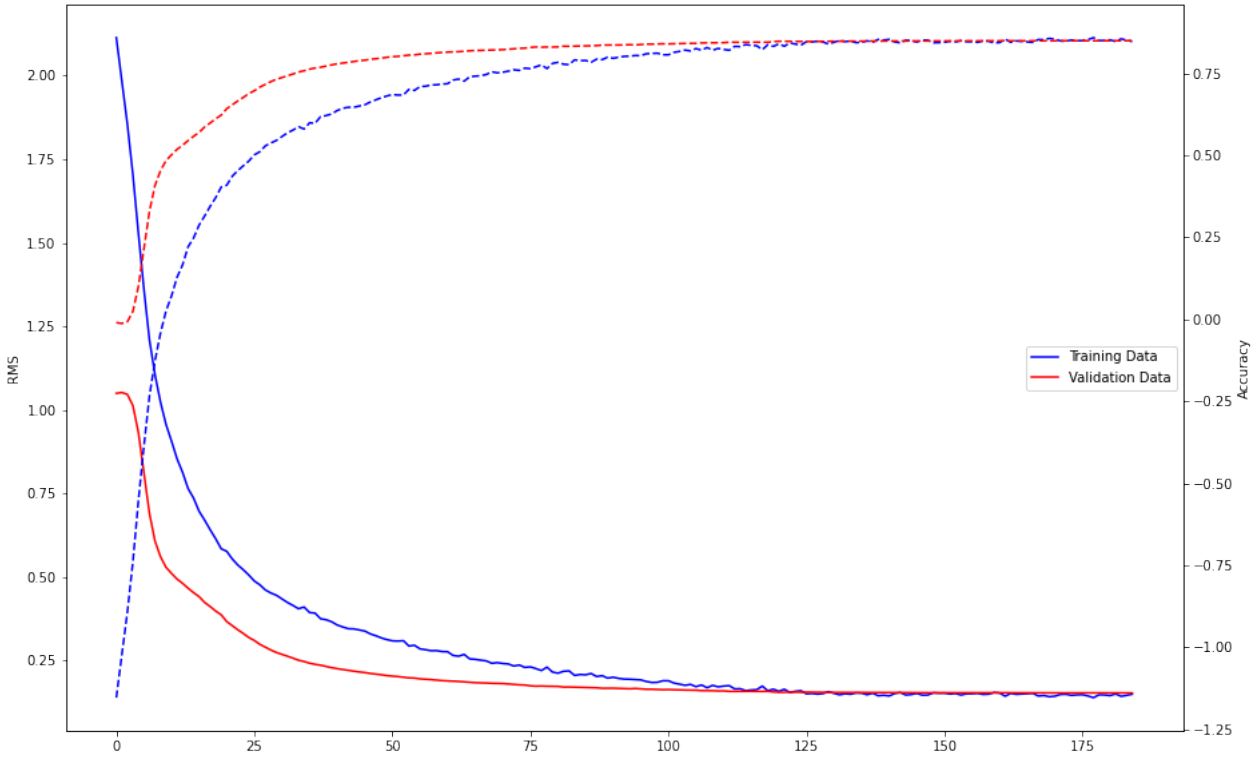


Figure G.1: Training plots from the training of the FFN used in the classical metamodelling of the PHN model.

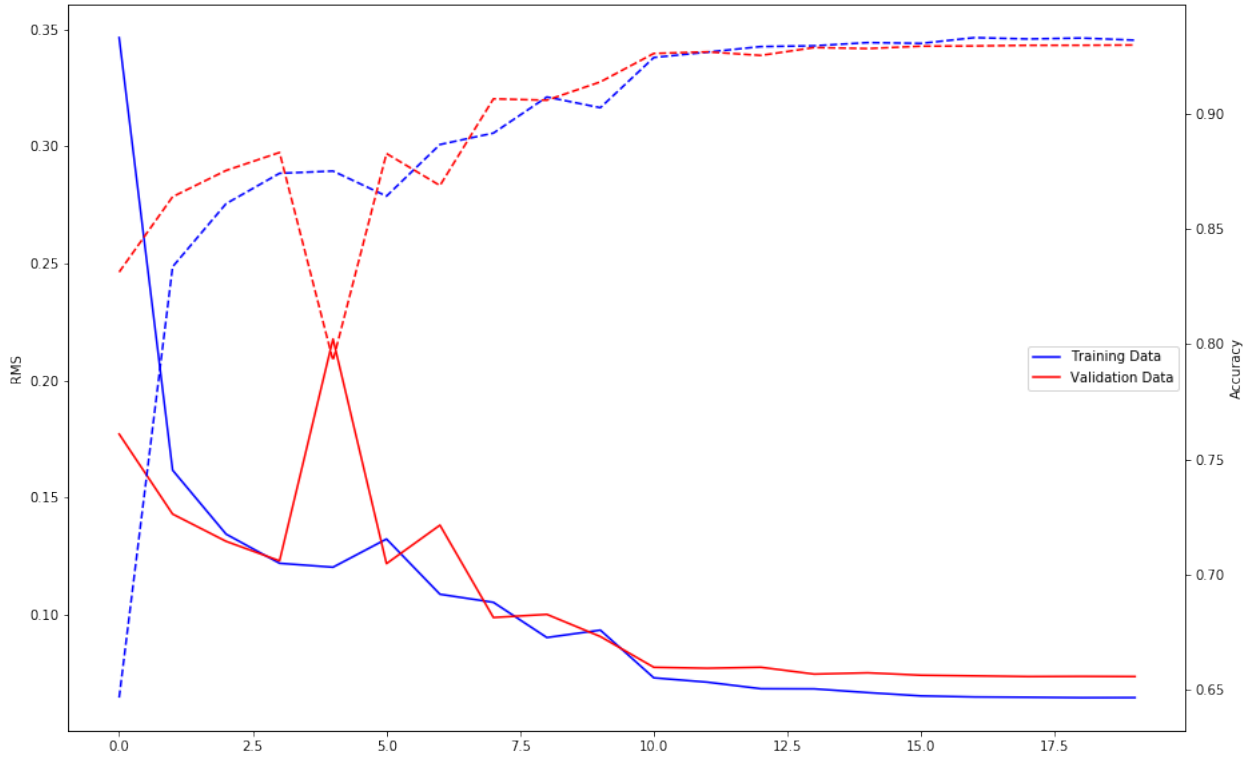


Figure G.2: Training plots from the training of the CNN used in the classical metamodelling of the PHN model

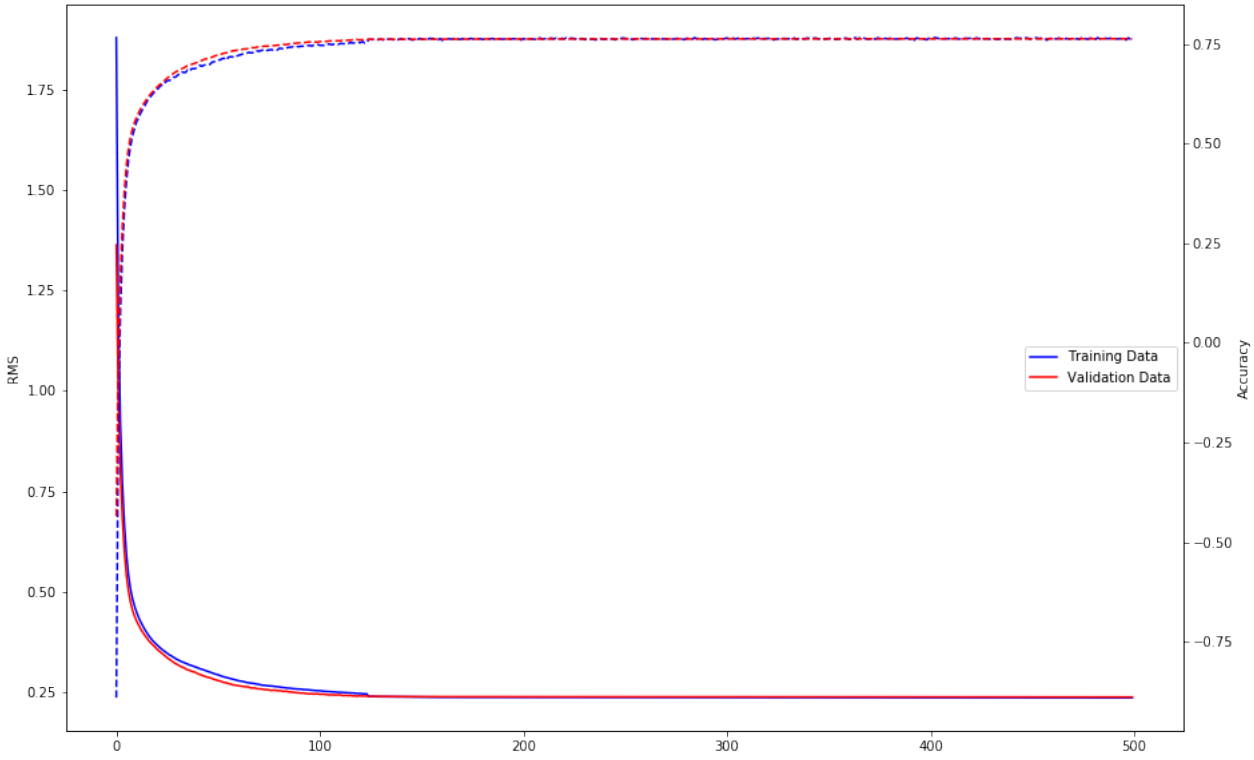


Figure G.3: Training plots from the training of the FFN used in the classical metamodelling of the aggregated phenotypes of the PHN model

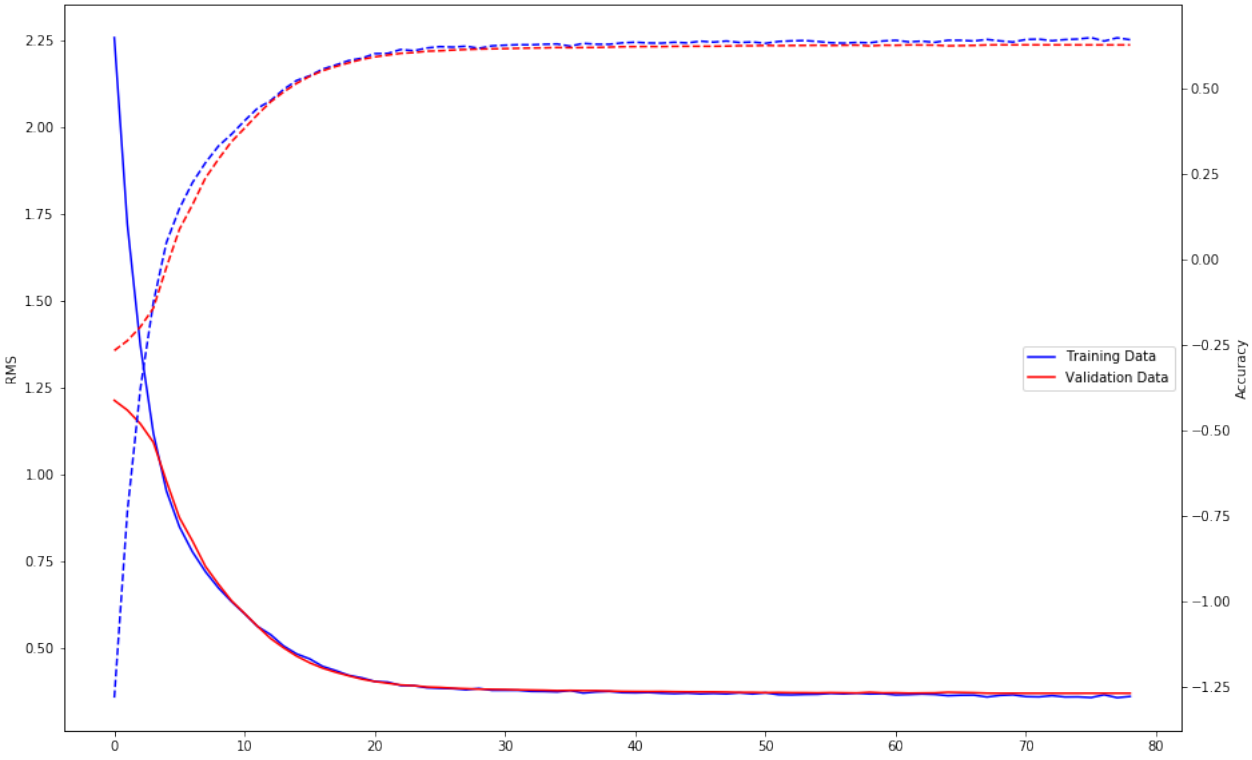
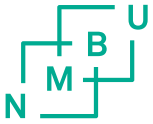


Figure G.4: Training plots from the training of the CNN used in the classical metamodelling of the aggregated phenotypes of the PHN model



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway