International Journal of Computer Applications (0975 - 8887)
Volume 184 - No.14, May 2022

Artificial Neural Network Comparison on hERG Channel Blockade Detection

Haibo Liu
ESR-fellow of the INSPIRE-ITN
NOTOCORD and Inria

Tessa De Korte
Leiden University Medical Center
2333 ZA Leiden, Netherlands

Sylvain Bernasconi
NOTOCORD part of Instem
78230 Le Pecq, France

Christophe Bleunven
NOTOCORD part of Instem
78230 Le Pecq, France

Damiano Lombardi
Research Scientist at Inria
Inria, 751012, Paris, France

Muriel Boulakia
professor at Laboratoire de Mathematiques de Versailles
78000, Versailles, France

ABSTRACT

This work will present a comparison of several Artificial Neural Network methods for a classification problem related to cardiac safety assessment. Given the extracellular field potential recorded by means of micro-electrode arrays, the aim is to determine whether a given chemical drug is altering the electrical activity of cardiomyocytes by disrupting the normal behavior of the hERG channels. To do so, this work has considered four different Neural Network methods and compared them in terms of accuracy and computational costs. The conclusion is that, among the tested architectures, the Multilayer Perceptron (MLP) and multivariate 1-dimensional Convolutional Neural Network (1D-CNN) give the most promising results.

General Terms

Neural networks, Classification

Keywords

Artificial Neural Networks, classification problems, cardiac safety assessment, safety pharmacology

1. INTRODUCTION

According to [1], cardiotoxicity has become one of the major causes of drug discontinuation in preclinical and clinical drug development. More specifically, as presented in the studies [2, 3], several non-cardiovascular drugs were withdrawn from clinical use from 1990 to 2001 because they were associated with QT interval prolongation by blocking ion channels. Therefore, it is crucial for the pharmaceutical industry to develop effective methods to study and identify the cardiotoxicity risk in drug development at an early stage. To do so, measuring the electrophysiology of human-induced pluripotent stem cell-derived cardiomyocytes (hiPSC-CMs) using multi-electrodes arrays (MEA) is a very promising technology that sets up high-throughput drug screening methods. Among the different ion channels, a special attention is paid to the human ether-a-go-go-related gene (hERG) channel and assessing drug effects on its activity is an important part of the cardiac safety risk assessment.
In this work, the aim is to evaluate to what extent a drug affects the hERG potassium channel by using Artificial Neural Network (ANN) methods on the field potential (FP) recordings of hiPSC-CMs obtained with MEA technology. In particular, given the electrograms recorded by MEA after adding a drug to a mono-layer of hiPSC-CMs, this work seeks to determine whether or not the drug can be classified as a hERG potassium blocker. By investigating different ANNs, this work expects to find the most suitable and effective methods to automatically identify which drugs cause hERG potassium channel blockade.
It is important to note that this classification problem has already been addressed in [4] through the use of a greedy classifier optimization strategy. This paper is a continuation of the previous work and aims to test Neural Networks that are known to be particularly effective and have the advantage that, compared to the methods developed in [4], the pre-processing step to extract features from the data is generally not needed for CNN methods.
The articles [5]-[6] presented several Neural Network methods that are able to improve the accuracy of different learning tasks for electrocardiogram (ECG) analysis. Among the proposed methods, MLP and CNN are the most popular Neural Network methods that have been widely used for classification and prediction purposes in different domains, including ECG arrhythmia classification. Neural Networks have also been widely tested in electroencephalography (EEG) research like brain computer interfaces, sleep analysis and seizure detection. Especially, CNN methods are used to detect and diagnose seizures based on EEG signals ([7, 8]).
In this article, Section 2 presents the experimental dataset that we have considered and the data pre-processing needed for MLP, one of the tested ANN methods. Then, Section 3 details the type of input and the architecture for the tested ANN methods: MLP, Univariate 1D-CNN, Multivariate 1D-CNN and 2D-CNN. At last, in Sections 4 and 5 we present the design and results of the classification tasks given by these four ANN methods and we compare them in terms of performances, data processing costs and network training costs.

2. PRESENTATION OF THE DATASET AND PRE-PROCESSING

2.1 Experimental dataset

This study has considered the same experimental data and setup as in the work of [4]. The experimental data is FP recordings of hiPSC-CMs obtained with MEA technology. MEA includes a two-dimensional arrangement of micro-electrodes that can monitor the extracellular electrical activity of the cultured cells (we refer to [9] for an overview on the background of MEA measurements of hiPSC-CM). More specifically, the experimental data correspond to FP signals before and after addition of 12 drugs, using 96-well MEA plates, each well containing 8 recording electrodes. Each drug has been added at four concentrations to four different wells using five replicates per concentration.

As mentioned in [4], when a drug is added to the hiPSC-CM, the FP can be altered. For instance, the drug can some certain ion channels of the cells, it can reduce the original amplitude or prolong the duration of each cardiac beat in the signals. However, there are big variabilities in the whole recording and each drug has a specific effect. Using the basic statistical tools to analyse those signals often faces difficulties and gives biased results. In this work, ANNs have been tested to have a more comprehensive and effective analysis on FP data set.

The drugs of the dataset are listed in Table 1 for each drug where specified the hERG potassium (K), Calcium (Ca) and Sodium (Na) IC50 values and the concentrations that have been tested. The last column of Table 1 summarizes the known effects of the drugs by classifying them as K, Ca or Na blockers or mixed blockers. When a tested drug has a predominant impact on K channel, it will be considered as a K blocker. Based on this information, all the 12 drugs, except Diltiazem, can be considered as pure or mixed K blockers. Indeed, Diltiazem is the only drug in this list with a K IC50 value that is much higher than the Ca IC50 and higher than the top test concentration and has been classified as a Ca blocker.

Each cardiac beat has been extracted from the recording sequences coming from each electrode. Since each beat may have a different length depending on the variability of the hiPSC-CMs and the recording conditions, all cardiac beats have been resampled using 0.08 ms time step to normalize each beat duration to 885.6 ms (11070 samples). For more details on the experimental data and setup, we refer to the work [4] which considered the same dataset.

Table 1. : Experimental data information

Drug	IC50 (μM)			Concentration (μM)				Type of blocker
	K	Ca	Na	#1	#2	#3	#4	
Loratadine	6.1	11.4	28.9	0.001	0.003	0.0095	0.03	K, Ca
Ibutilide	0.018	62.5	42.5	0.0001	0.001	0.01	0.1	K
Droperidol	0.06	7.6	22.7	0.03169	0.10014	0.31646	1.0	K
Mexiletine	62.2	125	38	0.1	1.0	10	100	Na , K
Dofetilide	0.03	26.7	162.1	0.0003	0.001	0.0032	0.01	K
Diltazem	13.2	0.76	22.4	0.01	0.1	1.0	10	Ca
Chlorpromazine	1.5	3.4	3	0.0951	0.3004	0.9494	3	K , Ca , Na
Clozapine	2.3	3.6	3	0.0951	0.3004	0.9494	3	K , Ca
Clarithromycine	32.9	>30	NA	0.1	1	10	100	K
Cisapride	0.02	11.8	337	0.0032	0.01	0.0316	0.1	K
Bepridil	0.16	1.0	2.3	0.01	0.1	1	10	K , Ca , Na
Azimilide	<1	17.8	19	0.01	0.1	1	10	K , Ca , Na

The information contained in this table comes from [4] and the references therein.

For each beat, a reasonable signal has to contain both depolarization and repolarization phases. Among all the collected cardiac beats, some signals may have an altered depolarization or repolarization phase compared to the majority of the signals and these abnormal signals have been removed from the dataset.

In the experiments, the drugs have been added after a few minutes of baseline recording so that each electrode provides two recordings: the time period corresponding to the baseline recording (prior to drug addition) is denoted by $\mathcal{P}_1$ and the time period corresponding to the post-addition recording (after drug addition) is denoted by $\mathcal{P}_2$. To construct the training set, all the beats in the period $\mathcal{P}_1$ and the beats obtained from the experiments of Diltiazem in the period $\mathcal{P}_2$ will be labeled as non-K blocker (this class is denoted by NO-K-blocker). On the other hand, the beats obtained when one of the 11 K blockers was added to the experiments in the period $\mathcal{P}_2$ will be labeled as K blocker (this class is denoted by K-blocker).

2.2 Pre-processing for MLP method

Compared to CNN methods that can directly use the raw signal, MLP method requires a pre-processing step to extract some markers or features from the signal that will be used as input. This data pre-processing step represents an additional computational cost but it can significantly reduce the input size compared to the raw signals and, by this way, it can substantially speed up the training phase. To facilitate the feature extraction process, we have separated the signal into two phases: each beat of the signal has been split into a depolarization phase with a duration of 25.6 ms, and a repolarization phase with a duration of 860 ms.

To make the computation of the features more accurate, the impact of signal noise has been mitigated by using Gaussian filtering, following [10]. Since the signal to noise ratio for the repolarization phase of the signal is larger, a Gaussian kernel with a larger standard deviation for the repolarization phase (its value is equal to 40) than for the depolarization phase (its value is equal to 2.5) have chosen.

In summary, pre-processing step has extracted from each beat 64 features distributed into 23 features for the depolarization phase, 37 features for the repolarization phase, and 4 features for the entire signal. The features are listed in Tables 2 and 3 and some of them are displayed in Figure 1 and 2. The following notations are used in the tables:

—the covariance matrix used to compute Maximum Eigenvalue for depolarization phase (DEV) and Maximum Eigenvalue for repolarization phase (REV) is the matrix given by: for all $1 \leq i, j \leq 8$

$$C_{ij} = \frac{1}{N} \sum_{k=1}^{N} (e_i^{(k)} - \bar{e}_i)(e_j^{(k)} - \bar{e}_j)^T$$

where N corresponds to the number of time steps, $e_i^{(k)} \in \mathbb{R}$ is the FP signal recorded by the i-th electrode and $\bar{e}_i$ is the mean value of e_i

—Dw is the filtered signal restricted to the depolarization phase (that corresponds to the time interval delimited by DD in Figure 1) whereas Rw is the part of the filtered signal restricted to the repolarization wave. Dw is defined on $[0, t_D]$ and Rw is defined on $[{t_R}^1, {t_R}^2]$.

—A_D (resp. A_R) is the area under the curve of Dw (resp. of Rw):

$$A_D = \left| \int_0^{t_D} Dw(t)dt \right| \text{ and } A_R = \left| \int_{{t_R}^1}^{{t_R}^2} Rw(t)dt \right|$$

The third and sixth columns of Table 2 and the third column of Table 3 give the feature indexes. The DEV and REV correspond to one scalar feature. The other features listed in the tables have been extracted from each signal and, in a given well and at a fixed beat, the average, maximum, minimum, and standard deviation of the features over the different electrodes of the well have been computed. Then, these four values have been stored in the entries. In addition, Duration DD and Arrival time at the center DCT minimum values have been removed considering they have too extreme values when they are computed on a large number of beats.

Table 2. : Features for the depolarization and repolarization signals

Depolarization Phase			Repolarization Phase		
Features Name	**Methodology**	**Index of features**	**Features Name**	**Methodology**	**Index of features**
Maximum Eigenvalue (DEV)	Maximum eigenvalue of the covariance matrix C	1	Maximum Eigenvalue (REV)	Maximum eigenvalue of the covariance matrix C	24
Amplitude (DA)	$DA = \max(Dw) - \min(Dw)$	2, 3, 4, 5	Amplitude (RA)	$RA = \max(Rw) - \min(Rw)$	25, 26, 27, 28
Duration (DD)	Duration of Dw	6, 7, 8			
Amplitude in the Center (DC)	Value of Dw at the time where the area under the curve reaches $0.5A_D$	17, 18, 19, 20	Amplitude in the Center (RC)	Value of Rw at the time where the area under the curve reaches $0.5A_R$	49, 50, 51, 52
Maximum Slope ($Dsmax$)	$Dsmax = \max_{[0,t_D]} Dw'$	9, 10, 11, 12	Maximum Slope ($Rsmax$)	$Rsmax = \max_{[0,t_R]} Rw'$	33, 34, 35, 36
Minimum Slope ($Dsmin$)	$Dsmin = \min_{[0,t_D]}$Dw'	13, 14, 15, 16	Minimum Slope ($Rsmin$)	$Rsmin = \min_{[0,t_R]} Rw'$	37, 38, 39, 40
Arrival Time at the Center (DCT)	Time where the area under the curve reaches $0.5 \times A_D$	21, 22, 23	Arrival Time for Maximum Amplitude (RCT)	Time when Rw reaches its maximum value	29, 30, 31, 32
			25% of the area under the curve (RCT0.25)	Time where the area under the curve reaches $0.25A_R$	41, 42, 43, 44
			50% of the area under the curve (RCT0.5)	Time where the area under the curve reaches $0.5A_R$	45, 46, 47, 48
			75% of the area under the curve (RCT0.75)	Time where the area under the curve reaches $0.75A_R$	53, 54, 55, 56
			90% of the area under the curve (RCT0.9)	Time where the area under the curve reaches $0.9A_R$	57, 58, 59, 60

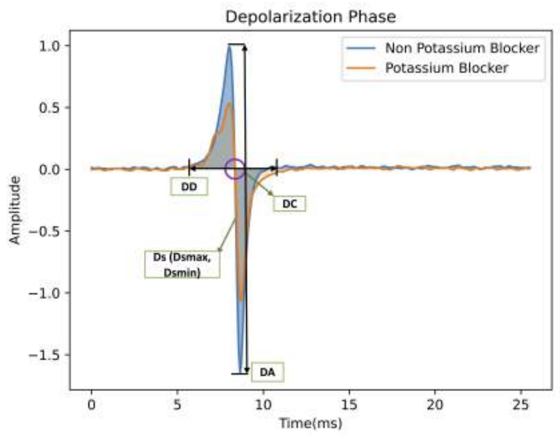

Fig. 1: A selection of depolarization phase features

Table 3. : Features for the whole signal

Features Name	Methodology	Index of features
Field potential duration (FPD)	The duration from beginning of depolarization wave to the end of the repolarization wave	61, 62, 63, 64

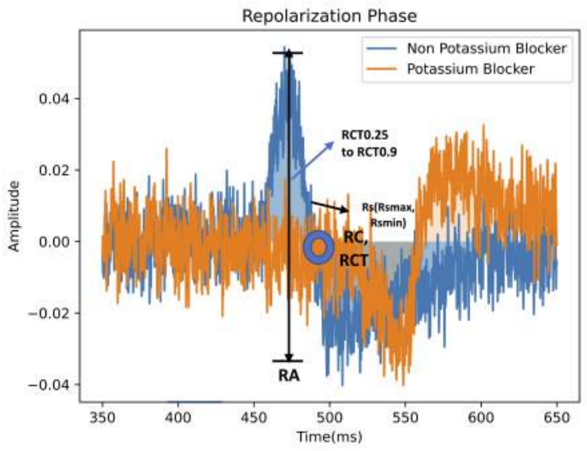

Fig. 2: A selection of repolarization phase features

Due to the fact that the orders of magnitude of the features widely vary, the features have been rescaled to ensure that the statistical distribution of the input data is roughly in the same range. Since our goal is to detect the impact of drugs on the signal, the idea is to rescale the features extracted from beats corresponding to period $\mathcal{P}_1$ or $\mathcal{P}_2$ by features corresponding to period $\mathcal{P}_1$.
More precisely, considering a feature coming from one beat taken in period $\mathcal{P}_1$, it is rescaled by dividing the similar feature computed from another beat in period $\mathcal{P}_1$ in the same well. The arrays containing these rescaled features will be labeled as NO-K-blocker. On the other hand, considering a feature coming from one beat taken in period $\mathcal{P}_2$, it is rescaled by dividing the similar feature computed from another beat in period $\mathcal{P}_1$ in the same well. An array containing these features will be labeled as NO-K-blocker if it is computed from beats extracted from Diltiazem experiments whereas an array containing these features will be labeled as K-blocker if it is computed from beats extracted from K-blocker experiments.

3. ARTIFICIAL NEURAL NETWORKS METHODS

This study focuses on the classification between K or non-K blockers of the drugs listed in Table 1. To do so, four types of ANNs have tested: MLP, Univariate 1D-CNN, Multivariate 1D-CNN and 2D-CNN. In this section, the architecture of these ANN methods will be explained.

3.1 Methods used in ANN

Before presenting the different ANNs, the important techniques that have been used are listed. We start by detailing the activation functions:

(1) Since Rectifier Linear Unit (ReLU) has been used to improve Boltzmann Machines in [11], ReLU becomes a commonly used activation function [12]:

$$ReLU(x) = max(x, 0). \tag{1}$$

(2) The leaky ReLU function has been introduced by [13] and it is a variant of the ReLU function whose expression is given by:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases} \tag{2}$$

where a is a small constant number (we have taken $a = 0.3$ in our tests).

(3) The sigmoid activation function [12, 14] transforms the input into an output that lies in the interval $(0, 1)$ as follows:

$$\rho(x) = \frac{1}{1 + \exp(-x)} \tag{3}$$

The Batch Normalization operation is used to normalize and stabilize the distributions of the input layers, considering the study of [15]. The formula is the following ([16]):

$$\bar{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{4}$$

This equation normalizes the input elements x_i (which are the outputs from the previous activation layer) by calculating the mean μ_B and variance σ_B^2 over all the samples. Here, the positive constant ϵ prevents the calculation from being invalid when the variance is very small or equal to zero.

3.2 MLP

The first tested neural network is MLP. Since MLP was first proposed in [17], it has become a commonly used method that consists of sets of fully connected layers. The work of [18] proposed a method to detect arrhythmia in ECG using MLP.
In MLP method, if there are $L \in \mathbb{N}^*$ hidden layers and that, for each $1 \leq l \leq L$, the l-th layer has $k^{(l)} \in \mathbb{N}^*$ units. The input is denoted by $x \in \mathbb{R}^d$ (where $d \in \mathbb{N}^*$ corresponds to the number of features), the first hidden layer output is a set of $k^{(1)}$ values given by:

$$o_i = \phi\Big(\sum_{j=1}^{d} w_{ij} x_j + b_i\Big), \ 1 \leq i \leq k^{(1)}, \tag{5}$$

where $\phi(.)$ is the activation function. Then, the output of the first hidden layer will be the input of the second hidden layer, the output of the second hidden layer will be the input of the third hidden layer, until the last hidden layer (see [19, Chapter 4, section 1.1], [20]).
The architecture of the MLP network is shown in Figure 3 (we refer to [21, Chapter 6], for the definitions of the technical terms that follow). In the first layer which corresponds to the input layer, 64 features are fed into the network. A batch-load [19, Chapter 11, section 5] has been used and each load will have 40 sets of 64 features propagated through the network. The weights for the first hidden layer are initialized with random normal distributed numbers. The bias for the first layer is initialized to zero. The output from the first hidden layer is then rendered to the next 6 sets of fully connected layers. In the 6 sets of fully connected layers, they have 320, 320, 192, 64, 32, and 10 hidden units in the layers and the neurons are activated by utilizing the ReLU activation function (1) in each hidden layer considering the features were rescaled in Section 2. The output from the last hidden layer will be passed to the fully connected output layer of 1 neuron with the sigmoid activation function (3) to provide a prediction for the binary classification.
During the training phase of the network, the predictions from the MLP method will be compared with the actual labels in order to compute the loss for each training. The Binary Cross Entropy has been chosen as the loss function. The weights of hidden layers were computed by using an Adam optimizer [19, Chapter 11, section 7].

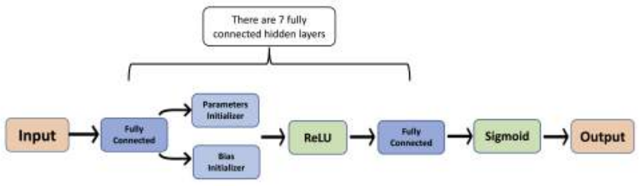

Fig. 3: Architecture of the MLP model

3.3 1D-CNN

When CNN was first proposed in [22], it was named a self-organized neural network. After many improvements and extensions [23], CNN has become a neural network commonly applied to image analysis. While 1D-CNN is very often used for the classification of time series data, 2D-CNN is a leading machine learning approach for image classification. Compared to MLP, the CNN method consists of convolutional layers and works as a feature extractor, so it does not require heavy data pre-processing.

3.3.1 Pre-processing for 1D-CNN. Compared to the MLP method, the data pre-processing for 1D-CNN is much simpler. In this work, the univariate 1D-CNN and the multivariate 1D-CNN have been tested. The univariate 1D-CNN method takes a single signal as input. We label the individual signals in period $\mathcal{P}_1$ and the signals obtained from the experiments of Diltiazem as NO-K-blocker and those signals are denoted by $s_p(t)$. We label the individual signals obtained from experiments of K-blocker drugs as K-blocker and those signals are denoted by $s_{np}(t)$.
On the other hand, the multivariate 1D-CNN takes a pair of signals as input. One signal from period $\mathcal{P}_1$ is paired with another signal in the same well and beat but recorded by a different electrode (in particular, it also belongs to period $\mathcal{P}_1$). $s_{c1}(t)$ and $s_{c2}(t)$ are the notations of these paired signals and N is the notation of their length. Then, S is defined as the array in $\mathbb{R}^{2\times N}$ given by :

$$S_{ij} = \begin{cases} s_{c1}(t_j), & i = 1 \\ s_{c2}(t_j), & i = 2 \end{cases} \tag{6}$$

where $(t_j)_{1\leq j\leq N}$ corresponds to the set of the time steps. All these pairs of signals will be labeled as NO-K-blocker.
In a comparable way, one signal from period $\mathcal{P}_1$ (that is denoted by $s_c(t)$) is paired with one signal from period $\mathcal{P}_2$ (that is denoted by $s_d(t)$) in the same electrode and well by introducing the array S of size $2 \times N$ given by:

$$S_{ij} = \begin{cases} s_c(t_j), & i = 1 \\ s_d(t_j), & i = 2 \end{cases} \tag{7}$$

For these pairs of signals, the ones coming from experiments of Diltiazem will be labeled as NO-K-blocker whereas the ones coming from experiments of K-blocker drugs will be labeled as K-blocker.

3.3.2 1D-CNN methodology and architecture. The architecture of the implemented 1D-CNN method (which is common to the univariate and multivariate 1D-CNNs) is shown in Figure 4. It is based on the traditional AlexNet [24] proposed in [25]. In the CNN method, the raw signals are used as input. The weights for the first hidden layer are initialized with random normal distributed numbers. Then, the kernel window with a height of 5 and a width of 1 unit slides across the input time series to do a cross-correlation operation [19, Chapter 6, section 2] with 1 stride from top to bottom. The example of a cross-correlation operation can be seen in Figure 5. The results from those cross-correlation operations con-

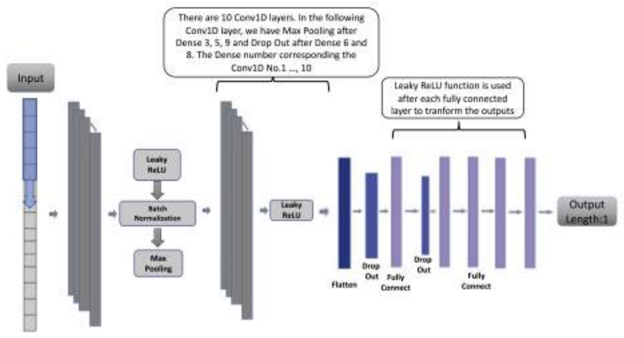
There are 10 Conv1D layers. In the following Conv1D layer, we have Max Pooling after Dense 3, 5, 9 and Drop Out after Dense 6 and 8. The Dense number corresponding the Conv1D No.1 ..., 10
Input
Leaky ReLU
Batch Normalization
Max Pooling
Leaky ReLU
Leaky ReLU function is used after each fully connected layer to tranform the outputs
Flatten
Drop Out
Fully Connect
Drop Out
Fully Connect
Output Length:1

Fig. 4: Architecture of the 1D-CNN model

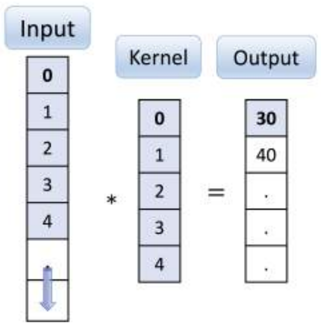

Fig. 5: One-dimensional cross-correlation operation. The shaded portions are the first output element as well as the input and kernel tensor elements used for the output computation: $0 \times 0 + 1 \times 1 + 2 \times 2 + 3 \times 3 + 4 \times 4 = 30$

stitute the features extracted from the first convolutional layer (the convolutional layer can be called Conv1D) and the Conv1D has 16 extractors to do this kind of computation named convolution filters. The output from the first Conv1D will be transformed by a leaky ReLU function (2) before the next process. As ReLU (1) could not be used as negative features are transformed to 0. So, to avoid losing information from negative features in the following computation, the leaky ReLU (2) has been chosen to transform the output from all Conv1D layers and fully connected layers.
After the first Conv1D layer and the activation process, the Batch Normalization layer has been used to speed up the training process and reduce the sensitivity of the initialization of the convolutional neural network [16]. Then the max-filter of the max-pooling layer [19, Chapter 6, section 5] extracts the maximum values from the defined region which has a height of 5 and a width of 1, as has been proposed in [25]. There is also a dropout layer with a rate of 50% added between several Conv1D layers. Then the other convolution layers are following the same logic as the first convolution layer. In summary, our model includes 10 Conv1D layers and each layer includes: 16, 16, 26, 26, 32, 52, 52, 72, 72, and 84 filters. The max pooling or dropout layers have been put between several Conv1D layers.
At the end of the Conv1D layers, the flattened tensor reshapes the outputs from the previous layer in a one-dimensional array. After flattening the layer, there is a dropout layer that reduces 50% of the parameters. Then some fully connected layers were added to the dropout layer. Every neuron in a fully connected layer is connected to every neuron in the next fully connected layer. There are 5 fully connected layers that have 872, 328, 128, 64, and 16 neurons respectively. Between the last fully connected layer and the output, the sigmoid activation function (3) has been used to transform the input to the output.
In the training phase of the network, we have used Binary Cross Entropy as the loss function and the weights of hidden layers were computed by using an Adam optimizer for CNN models.

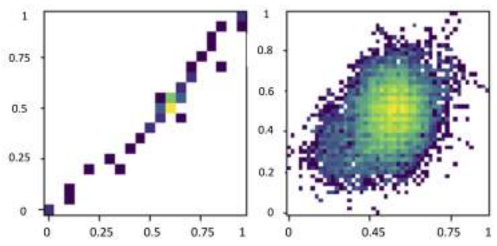

Fig. 6: Example of image obtained by pairing signals both in period $\mathcal{P}_1$ as in (6). Left: depolarization phase, right: repolarization phase

3.4 2D-CNN

3.4.1 Pre-processing for 2D-CNN. Before presenting the 2D-CNN method, let us explain how images have been generated from the signals. The main idea is to associate an image to a pair of signals constructed in the data pre-processing step for the multivariate CNN. The image is a portrait in which the first signal of the pair is a function of the second signal of the pair.
Since the depolarization phase and the repolarization phase are very different in terms of magnitude and duration, we will consider them separately and two images will be associated with each pair of signals after a renormalization step. To be more precise, reusing the notations (6) or (7), we define

$$p = \max_{1 \le i \le 2, 1 \le j \le N} S_{ij} \text{ and } q = \min_{1 \le i \le 2, 1 \le j \le N} S_{ij}$$

and then, we define the following points in the 2D space:

$$\left(\frac{S_{1j} - q}{p - q}, \frac{S_{2j} - q}{p - q} \right)$$

for $1 \le j \le N$.
The images have been generated from the data of these points considering the following process. Every image is divided into a certain number of squares. For the depolarization and repolarization phase, the graph is divided into 30×30 and 60×60 squares respectively. The images have been generated by counting how many curve points lay in each square. The number of points included in each square also represents the density of the pixels and the density is presented by the colour of the squares in the image. Examples of images are given in Figure 6 and 7. When a drug causes an alteration of the signals, we expect to observe that the pixels diverge along the diagonals.

3.4.2 2D-CNN methodology and architecture. These generated images are the inputs of the 2D-CNN method. The size of the image corresponds to height$\times$ width $\times$ colour channels. In contrast to 1D-CNN method, the kernel for 2D-CNN slides crosses the input along with two dimensional directions which are the height and width of the image like in Figure 8. The sizes for kernel and max pooling window correspond to height $\times$ width $\times$ colour channel.

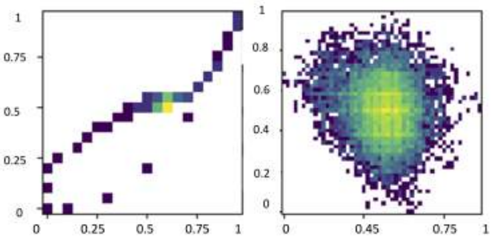

Fig. 7: Example of image obtained by pairing a signal coming from period $\mathcal{P}_1$ and a signal coming from period $\mathcal{P}_2$ as in (7). Left: depolarization phase, right: repolarization phase

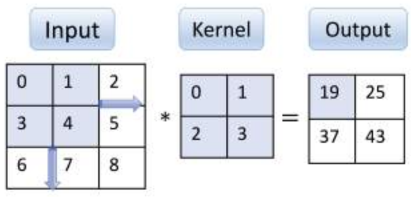

Fig. 8: Two-dimensional cross-correlation operation. The shaded portions correspond to the first output element as well as the input and kernel tensor elements used for the computation of this output: $0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$, figure from [19, Chapter 6, section 4.1].

Table 4. : Details for 2D CNN model: number of layers and their type, number of neurons, kernel size, whether it has Batch Normalization, max pooling (and, if so, the size) and dropout (and, if so, the rate)

Number of Hidden Layers	Number of Filters	Number of Neurons	Kernel Size	Batch Normalization	Max Pooling	Dropout
Conv2D Layer 1	16	None	3	Yes	Yes, pool size(2, 2)	Yes, 20%
Conv2D Layer 2	16	None	3	Yes	No	No
Conv2D Layer 3	32	None	3	No	Yes, pool size(2, 2)	No
Conv2D Layer 4	32	None	3	No	No	Yes, 20%
Conv2D Layer 5	54	None	3	No	Yes, pool size(2, 2)	No
Conv2D Layer 6	54	None	3	No	No	Yes, 20%
Conv2D Layer 7	62	None	3	No	No	Yes, 20%
Conv2D Layer 8	62	None	3	No	Yes, pool size(2, 2)	Yes, 20%
Flatten Layer 9	None	None	None	No	No	Yes, 20%
Dense Layer 10	None	872	None	No	No	Yes, 20%
Dense Layer 11	None	328	None	No	No	Yes, 20%
Dense Layer 12	None	164	None	No	No	No
Dense Layer 13	None	64	None	No	No	No
Dense Layer 14	None	16	None	No	No	No

In the 2D-CNN model, the kernel size is $3 \times 3 \times 3$, the window size is $2 \times 2 \times 3$ for max-pooling and a 20% dropout rate. The architecture of proposed 2D-CNN model is shown in Figure 9. There are 8 hidden two dimensional convolution layers (Conv2D) and the input of image data will be passed to those sets of Conv2D layers. The results from the last Conv2D layer will be flattened by a Flatten layer. Then, the flattened parameters will be pass to a set of fully connected layers. The results from each hidden layer will be transformed by a ReLU activation function (1). The outputs from the last hidden layer will be passed to the fully connected output layer of 1 neuron with the sigmoid activation function (3). Binary Cross Entropy was used as a loss function and the weights of hidden layers were computed by using an Adam optimizer.

Table 5. : Criteria to evaluate the classification methods

Criterion	Meaning
False negative (FN)	The classification result where positive training data are evaluated as negative
Percentage of FN (FN%)	$FN\% = \frac{FN}{TP+FN}$
False positive (FP)	The classification result where negative training data are evaluated as positive
True negative (TN)	The classification result where negative training data are evaluated as negative
True positive (TP)	The classification result where positive training data are evaluated as positive
Accuracy	The percentage of signals correctly classified, $\frac{TP+TN}{TP+TN+FP+FN}$
Precision	The percentage of predicted controls cases that were correctly classified, $\frac{TP}{TP+FP}$
Recall	The percentage of actual control cases that were correctly classified, $\frac{TP}{TP+FN}$
AUC	Area under the curve of a receiver operating characteristic (ROC) curve

More details of the architecture of the 2D-CNN model are given in Table 4 which includes the name of hidden layers, the number of filters and the kernel size for each Conv2D layer and the number of neurons in the fully connected layers. Table 4 also lists the information regarding whether there is a Batch Normalization layer, Max Pooling layer or Dropout process after each Conv2D or fully connected layer. The techniques used in Table 4 refer to [19, Chapter 6], [21, Chapter 9].

4. CLASSIFICATION SETUP

This section will present the details of the design of the classification tasks for testing four ANN methods presented above (MLP, Univariate 1D-CNN, Multivariate 1D-CNN and 2D-CNN). The tests have performed for two different scenarios:

—*Scenario 1*: The training set is composed of random signals taken from the recordings of all the 11 K blockers of our dataset. The test set is composed of random signals taken from the same drugs.

—*Scenario 2*: The training set is composed of random signals taken from the recordings of only 7 K blockers among the 11. The test set is composed of random signals from the 4 remaining K blockers.

For each scenario, the four ANN methods have been tested to predict the label of the signals in the test set. The obtained results make it possible to compare the ANN methods in terms of performances, data processing costs and network training costs. The performances of the networks are evaluated using the classical criteria [25] which are listed in Table 5. The ANN methods have implemented using the Sequential model from $TensorFlow^{TM}$ [26].

4.1 Training and test sets

For *scenario 1*, different methods have been implemented in the following way:

(1) The MLP method takes the rescaled features array of size 64 as input. There are 29116 signals have been selected.
(2) The Univariate 1D-CNN method takes single signals as inputs. 27831 signals have been randomly chosen from NO-K-blocker experiments and 30000 signals from K-blocker experiments (with a third of the signals in plate 1, a third of the signals in plate 2 and a third of the signals in plate 3, each plate containing signals from 4 tested drugs).
(3) The Multivariate 1D-CNN method takes pairs of signals as inputs. The same signals have been used as inputs for the Univariate CNN method.
(4) The 2D-CNN takes images as inputs. The images were generated from the same set of signals as the Multivariate 1D-CNN. By loading the images to the 2D-CNN, each image corresponds to a matrix of size 97 $\times$ 181 $\times$ 3 (corresponding to height, width and colour channels).

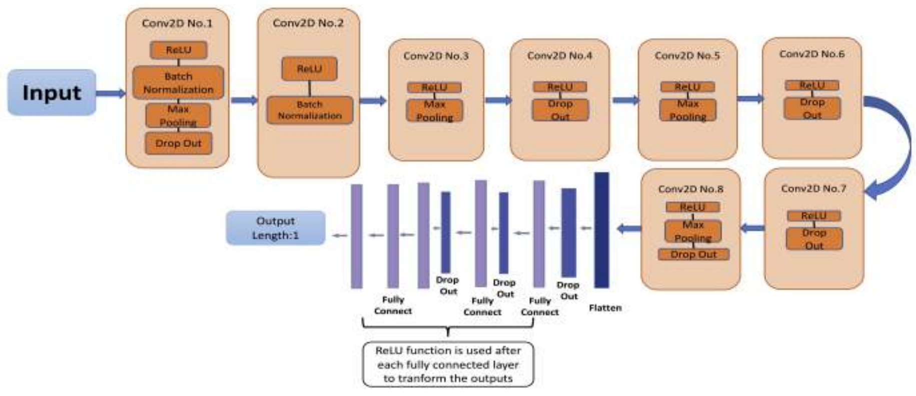

Fig. 9: Architecture of the 2D-CNN model

During the model training phase, there are 20% of the signals randomly distributed in the test set and 80% of the signals in the training set. In the training set, 20% have been randomly chosen as a validation set.
For *scenario 2*, the same signals have been used as in *scenario 1* for the four types of ANNs. However, the signals have been divided in a different way since the training is made by using signals coming only from a part of the drugs. More precisely, the network is trained by using the signals coming from only 7 K blockers (Loratadine, Ibutilide, Mexiletine, Droperidol, Chlorpromazine, Clozapine, Dofetilide). In the training part, 20% of signals have been randomly chosen as the validation set and the rest of the signals are in the training set. Then, the signals coming from the 4 remaining K blockers (Clarithromycine, Cisapride, Bepridil, Azimilide) are used to test the networks.

5. CLASSIFICATION RESULTS AND METHODS EVALUATIONS

In this subsection, the classification results obtained for each scenario and each ANN method are presented and the evaluations of each method will be listed.

5.1 Classification results

The classification results are listed in Table 6 for *scenario 1* and Table 7 for *scenario 2*. In general, the performances of MLP and multivariate 1D-CNN methods are superior with 96.82% and 99.26% of accuracy for *scenario 1* and 98.33% and 99.13% of accuracy for *scenario 2*, respectively. A special attention has to be paid to the error rate of K-blocker wrongly classified as NO-K-blocker, because in practice this may have more critical outcomes. Again, the MLP and multivariate 1D-CNN methods give satisfactory results with respectively 0.2% and 0.14% being false negative (FN). A study on the signals which have been wrongly classified allows to notice that a large part of them corresponds to drugs whose channel effect is complex either because they correspond to the mixed blockers, like Chlorpromazine and Clozapine, or because it has been classified as a non-K blocker (Diltiazem) whereas this drug partly blocks the K channel at high concentration. So, the classification error is partly related to the fact that classifying the drug as K-blocker or non-K blocker may sometimes be reductive due to their complex effect on the ionic channel activity. There will be more discussion about this point in the Section 6.

Table 6. : Classification results of the tested ANNs with *scenario 1*

Metrics	**Classifier**			
	MLP	Univariate 1D-CNN	Multivariate 1D-CNN	2D-CNN
Accuracy	**96.82%**	86.58%	**99.26%**	88.50%
Precision	**92.79%**	86.98%	**98.70%**	89.92%
Recall	**99.74%**	85.98%	**99.86%**	87.52%
AUC	**99.63%**	94.21%	**99.90%**	95.65%
FN%	**0.20%**	13.80%	**0.14%**	12.95%

Table 7. : Classification results of the tested ANNs with *scenario 2*

Metrics	**Classifier**			
	MLP	Univariate 1D-CNN	Multivariate 1D-CNN	2D-CNN
Accuracy	**98.33%**	64.63%	**99.13%**	84.42%
Precision	**99.90%**	65.62%	**98.76%**	88.43%
Recall	**96.00%**	61.47%	**99.67%**	82.42%
AUC	**98.80%**	70.85%	**99.63%**	92.27%
FN%	**2.72%**	36.24%	**0.41%**	19.77%

It is interesting to notice that, contrary to the other methods, the performances of MLP and multivariate 1D-CNN methods with *scenario 2* are similar to the ones with *scenario 1*. This suggests that these methods will have good predictive capabilities for testing new drugs whose action on ionic channels is still unknown.

Table 8. : Data Processing and Training Costs of the tested ANNs

Classifier	Data Processing Cost	Training Time for Each Epoch	Number of Epochs needed	Training Time
MLP	**Very High**	1 minutes	20 epochs	**20 minutes**
Univariate 1D-CNN	Very Low	3.5 minutes	100 epochs	5.8 hours
Multivariate 1D-CNN	**Very Low**	3.67 minutes	20 epochs	**1.2 hours**
2D-CNN	Low	6.67 minutes	60 epochs	6.7 hours

5.2 Methods evaluations

Considering the showed classification results, MLP and multivariate 1D-CNN methods can provide the most promising analysis of experimental data set. Besides the evaluation of the performance of the ANN methods, additional factors have to be considered such as the data processing costs and the model training costs to have a more comprehensive evaluation of each method. In terms of industrial implementation, it is important to consider which ANN methods would be easier to deploy. The data processing costs and model training costs are listed in Table 8. Depending on the size of the data set and computational power, the time demand would be different. The data processing costs are ranked from very low to very high. In terms of training costs, the training time for each epoch and the total training time are considered to get a well-trained model. In summary, MLP needs the shortest training time but has the largest data processing costs. Multivariate 1D-CNN has very low data processing costs and needs 1.2 hours of training time. In other words, compared to MLP, Multivariate 1D-CNN has a lower requirement for data processing but higher computation requirements for training the network.

6. CONCLUSION AND DISCUSSION

In this paper, MLP, univariate 1D-CNN, multivariate 1D-CNN and 2D-CNN neural network methods have been tested to detect whether a drug acts as a K channel blocker. After a detailed presentation of the architectures of the different tested NN methods, their effectiveness has been assessed by presenting their performance, data pre-processing costs and training costs. Considering these different criteria, our conclusion is that, among the four neural network methods, the MLP and multivariate 1D-CNN methods offer the most promising results and are well ranked candidates to develop a more comprehensive framework for high-throughput screening.
Several developments of this study could be explored in the future. As already mentioned in the previous section, it may be too restrictive to assign a class (K-blocker or NO-K-blocker) to a drug because this binary assignment does not reflect the complex behavior of drugs on ionic channels. In particular, once a drug has been identified as a K-blocker, a natural refinement could be to classify it as a pure K-blocker or a multi-channel blocker with the different possible combinations K+Na, K+Ca or K+Na+Ca. It may also be important to estimate the severity of a drug by classifying it as a strong, medium, or weak blocker, which may be associated with their risk to induce pro-arrhythmia. At last, this classification could consider each concentration separately since they may have a different impact on the ionic channels. More generally, the encouraging results of this study lead us to believe that the use of Neural Networks may allow to achieve a fine classification of drugs and may be of great help to assess cardiac safety of drugs in a semi-automatic and high throughput manner.

7. ACKNOWLEDGEMENTS

This project has been supported by INSPIRE and receives funding from the EU Horizon 2020 Research and Innovation programme, under the Marie Sklodowska-Curie GA 858070.

8. REFERENCES

[1] Nicola Ferri et al. "Drug attrition during pre-clinical and clinical development: Understanding and managing drug-induced cardiotoxicity". In: *Pharmacology and Therapeutics* 138.3 (2013), pp. 470–484. ISSN: 0163-7258. DOI: https://doi.org/10.1016/j.pharmthera.2013.03.005.

[2] Bernard Fermini and Anthony A Fossa. "The impact of drug-induced QT interval prolongation on drug discovery and development". In: *Nature reviews Drug discovery* 2.6 (2003), pp. 439–447.

[3] Antje D Ebert, Ping Liang, and Joseph C Wu. "Induced pluripotent stem cells as a disease modeling and drug screening platform". In: *Journal of cardiovascular pharmacology* 60.4 (2012), p. 408.

[4] Fabien Raphel et al. "A greedy classifier optimization strategy to assess ion channel blocking activity and pro-arrhythmia in hiPSC-cardiomyocytes". In: *PLoS computational biology* 16.9 (2020), e1008203.

[5] Nils Strodthoff et al. "Deep learning for ECG analysis: Benchmarks and insights from PTB-XL". In: *IEEE Journal of Biomedical and Health Informatics* 25.5 (2020), pp. 1519–1528.

[6] Amin Ullah et al. "Classification of arrhythmia by using deep learning with 2-D ECG spectral image representation". In: *Remote Sensing* 12.10 (2020), p. 1685.

[7] Alexander Craik, Yongtian He, and Jose L Contreras-Vidal. "Deep learning for electroencephalogram (EEG) classification tasks: a review". In: *Journal of neural engineering* 16.3 (2019), p. 031001.

[8] U Rajendra Acharya et al. "Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals". In: *Computers in biology and medicine* 100 (2018), pp. 270–278.

[9] Sophie Kussauer, Robert David, and Heiko Lemcke. "hiPSCs derived cardiac cells for drug and toxicity screening and disease modeling: what micro-electrode-array analyses can tell us". In: *Cells* 8.11 (2019), p. 1331.

[10] Guang Deng and LW Cahill. "An adaptive Gaussian filter for noise reduction and edge detection". In: *1993 IEEE conference record nuclear science symposium and medical imaging conference*. IEEE. 1993, pp. 1615–1619.

[11] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Icml*. 2010.

[12] Yingying Wang et al. "The influence of the activation function in a convolution neural network model of facial expression recognition". In: *Applied Sciences* 10.5 (2020), p. 1897.

[13] Bing Xu et al. "Empirical evaluation of rectified activations in convolutional network". In: *arXiv preprint arXiv:1505.00853* (2015).

[14] Jun Han and Claudio Moraga. "The influence of the sigmoid function parameters on the speed of backpropagation learning". In: *International workshop on artificial neural networks*. Springer. 1995, pp. 195–201.

[15] Shibani Santurkar et al. "How does batch normalization help optimization". In: *Advances in neural information processing systems* 31 (2018).

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

[17] Marius-Constantin Popescu et al. "Multilayer perceptron and neural networks". In: *WSEAS Transactions on Circuits and Systems* 8.7 (2009), pp. 579–588.

[18] Gaurav Kumar, Urja Pawar, and Ruairi O'Reilly. "Arrhythmia Detection in ECG Signals Using a Multilayer Perceptron Network." In: *AICS*. 2019, pp. 353–364.

[19] Aston Zhang et al. "Dive into deep learning". In: *arXiv preprint arXiv:2106.11342* (2021).

[20] Weili Guo et al. "Theoretical and numerical analysis of learning dynamics near singularity in multilayer perceptrons". In: *Neurocomputing* 151 (2015), pp. 390–400.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[22] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[23] Yann LeCun et al. "Object recognition with gradient-based learning". In: *Shape, contour and grouping in computer vision*. Springer, 1999, pp. 319–345.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[25] Yunan Wu et al. "A comparison of 1-D and 2-D deep convolutional neural networks in ECG classification". In: *arXiv preprint arXiv:1810.07088* (2018).

[26] Ekaba Bisong. *Building machine learning and deep learning models on Google cloud platform*. Springer, 2019.