EURECOM
Department of Digital Security
Campus SophiaTech
CS 50193
06904 Sophia Antipolis cedex
FRANCE

Research Report RR-19-341

PAC: Privacy-preserving Arrhythmia Classification with neural networks

November 13th, 2018
Last update August 30th, 2019

Mohamad Mansouri, Beyza Bozdemir, Melek Önen, and Orhan Ermis

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00
Email : {Mohamad.Mansouri, Beyza.Bozdemir, Melek.Onen,
Orhan.Ermis}@eurecom.fr

[1]EURECOM's research is partially supported by its industrial members: BMW Group Research and Technology, IABG, Monaco Telecom, Orange, Principauté de Monaco, SAP, Symantec.

PAC: Privacy-preserving Arrhythmia Classification with neural networks

Mohamad Mansouri, Beyza Bozdemir, Melek Önen, and Orhan Ermis

Abstract

Recent advances in information technology such as the Internet of Things enable businesses and organisations to collect large amounts of data and apply advanced machine learning techniques in order to infer valuable insights and improve predictions. Unfortunately, such benefits come with a high cost in terms of privacy exposures given the high sensitivity of the data that are usually analysed/processed at third party servers. In this study, we aim at protecting health data, more precisely, Electro-Cardiogram (ECG) data while enabling an efficient prediction of arrhythmia using neural networks. We propose a solution named PAC that combines the use of Neural Networks with secure two-party computation. To achieve a good trade-off between privacy, accuracy, and efficiency, we first build a dedicated, efficient neural network model for which the underlying operations can be further performed while data is privacy protected. The resulting model consists of two fully connected layers which perform small size matrix multiplication and one activation layer which executes a square function. The solution is implemented using the ABY framework and makes use of Arithmetic and Boolean circuits. Several approximations are performed over the representation of the data, accordingly. PAC also supports classifications in batches when needed. Experimental results run on the PhysioBank datasets show an accuracy of 96.34% which outperforms existing solutions.

Index Terms

data privacy; privacy-preserving neural networks; secure two-party computation; arrhythmia classification

Contents

1 Introduction **1**

2 Problem Statement **2**
2.1 Arrhythmia Classification with Neural Networks 2
2.2 Arrhythmia Classification with Data Privacy 3
2.3 Solution Idea 4

3 Privacy-preserving Neural Network Arrhythmia Classifier - A case study with PhysioBank **5**
3.1 The optimized neural network model 6
3.2 PAC: Detailed description 13

4 PAC in batches **16**
4.1 Description 16
4.2 Evaluation 19

5 Related Work **20**

6 Conclusion **21**

A The arrhythmia dataset from the PhysioBank database **25**

B Accuracy evaluation with different numbers of hidden neurons **26**

C First truncation method with 2PC using Arithmetic Circuits **26**

D Evaluation of models using the localhost **27**

List of Figures

1 The accuracy of the model with different dimensions of the input vector 7
2 Confusion matrix of the model for each class in the test dataset . . 8
3 The proposed NN model 9
4 PAC Overview 11
5 Arithmetic circuit representation of the model with Truncation v2 14
6 Accuracy with different hidden neurons (# input: 16, # output: 16) 26

1 Introduction

Artificial intelligence and machine learning have gained a renewed popularity thanks to the recent advances in information technology such as the Internet of Things that help collect, share and process large datasets. This powerful technology helps make better decisions and accurate predictions in many domains including heavy industry, transportation, finance and healthcare. In particular, Neural Networks (NN) can support pharmacists and doctors to analyse patients' data and quickly diagnose a particular disease such as heart arrhythmia that can cause sudden death. Nowadays, this disease can be detected at early stages with the help of smart wearable devices such as Apple Watch 4[1] that can record electric heart activities using Electro-Cardiograms (ECG) data.

Nevertheless, we are experiencing severe data breaches and these cause crucial damages. A recent research [1] concludes that in 2018 the global average cost of a data breach is 3.86 million dollars and the healthcare sector is the first sector facing huge costs. ECG data is considered as very sensitive and is even sometimes used for biometrics[2]. Therefore, there is an urgent need for tools enabling the protection of such collected data while still being able to launch predictive analytics and hence improve individuals' lives. These tools will also help stakeholders be compliant with the General Data Protection Regulations (GDPR)[3].

In this work, we aim at addressing privacy concerns raised by the analysis of the ECG data for arrhythmia classification. Our goal is to enable service providers perform classification without discovering the input (the ECG data) to this operation. On the other hand, we also look into the problem from the service providers' point of view as they care about keeping the design of their services confidential from the users. Users using these systems/solutions should not be able to discover the details about the underlying system (such as the Neural Network model). The challenge often manifests as a choice between the privacy of the user and the secrecy of the system parameters. We propose to reconcile both parties, namely the stakeholders and the users and combine the use of neural networks with secure two-party computation (2PC). Since secure two-party computation protocols cannot efficiently support all kinds of operations, we propose to revisit the underlying neural network operations and design a new, customized neural network model that can be executed to classify arrhythmia accurately, and this, without disclosing neither the input ECG data to the service provider nor the neural network parameters to the users.

The first goal is to minimize the overhead incurred by the use of 2PC. Hence, both the neural network architecture and the underlying operations should be customized and simplified. Furthermore, to be compatible with 2PC, the input values and the model's parameters also need to be rounded to integers. For this respect, two different methods have been proposed. To reduce the input size of neural net-

[1] https://www.apple.com/lae/apple-watch-series-4/health/

[2] https://findbiometrics.com/ecg-biometrics-connected-car-507264/

[3] https://eur-lex.europa.eu/eli/reg/2016/679/oj

work, we employ Principal Component Analysis (PCA) [2]. The proposed methodology is illustrated with a case study whereby some arrhythmia dataset from the PhysioBank database[4] is used. With this dataset, we show that the newly designed model only involves 2 layers with 54 hidden neurons. The resulting model is implemented in a realistic environment and uses the ABY framework [3] for 2PC. Experimental results show that the most optimal resulting model reaches an accuracy level of 96.34%. Our solution helps predict the class of a heartbeat in 1 second, approximately. We also evaluate the performance of the system under three different design choices based on the implementation (or not) of PCA (when PCA is performed at the user, when PCA is integrated to 2PC, and when PCA is not used). In order to improve the performance of the solution even further, we propose to make predictions in batches and thus help the analyser (the doctor) receive the prediction of a set of heartbeats for a given period (e.g., 30s). We show that by using the Single Instruction Multiple Data (SIMD) packing method offered by ABY, the computational overhead is significantly reduced.

The rest of the paper is organized as follows. In the next section, we introduce the problem of arrhythmia classification and identify the main challenges to ensure the privacy of the ECG data at the same time. Section 3 focuses on the case study and presents the newly proposed privacy-preserving variant of the neural network that we name PAC. Experimental results on its performance and accuracy are also provided. In section 4, we describe the additional optimisation method which consists of executing predictions in batches. Finally, we review the state of the art in Section 5.

2 Problem Statement

2.1 Arrhythmia Classification with Neural Networks

As defined in [4], cardiac arrhythmias are abnormal heart rhythms, which cause the heart to beat too fast (tachycardia) or too slow (bradycardia) and to pump blood less effectively. These irregularities can be detected and classified by analyzing the Electro-Cardiogram (ECG) signals of a heart. Doctors classify arrhythmia to several types according to such behaviors of the heart.

In this work, we focus on the classification of heartbeats extracted from ECG signals into different classes of arrhythmia using machine learning techniques. In order to design an efficient arrhythmia classifier, we propose to use Neural Networks (NN). A NN, as defined by the inventor of one of the first neurocomputers, Dr. Robert Hecht-Nielsen, is *a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs*. A neural network is typically organized in layers. Layers are made up of a number of interconnected nodes called neurons. Patterns are presented to the network via the input layer which communicates to

[4] `https://www.physionet.org/physiobank/database/mitdb`

one or more hidden layers where the actual processing is done via a system of weighted connections. Each neuron carries a value that represents its activation bias and each connection between two neuron in two successive layers carries a weight. Thus each layer has a bias vector B holding all the bias values and a weight matrix W holding all the connections' weights in the layer. The hidden layers then link to an output layer where the answer is the output of the system.

Building the arrhythmia classifier model involves the design of the architecture of the Neural Network, such as the number of the layers, the size of the input, the number of neurons in each layer and the underlying operations that each neuron has to perform. A dataset of ECG heartbeats representing different arrhythmia types should be prepared. The dataset is split into a training dataset (80%) and a test dataset (20%). In order to come up with a model that accurately predicts the actual arrhythmia type, we first train the model on a portion of the training dataset, then we evaluate its performance on the remaining portion. We use the results of the evaluation on the validation dataset to compare performance between different models and to choose the most efficient one. Finally, we test the performance of the chosen model by checking its performance on the test dataset.

2.2 Arrhythmia Classification with Data Privacy

ECG signals representing patients' heartbeats can be considered as sensitive information. Thus, outsourcing the arrhythmia classifier to online servers may put the privacy of the patients at risk. Hence, we aim at finding a solution where a party can execute the classification model without leaking and even discovering information about the input data. On the other hand, the classification model can also be considered as confidential against its users, namely parties who will send their queries for classification. This model itself can also have some business value and therefore be protected. For this respect, we assume that the model should be unknown to the parties querying it.

Performing some operations over data while these are kept confidential requires the use of advanced cryptographic tools such as homomorphic encryption [5–8] or secure multi-party computation [9, 10]. While the integration of such tools offers better privacy guarantees, they unfortunately introduce some non-negligible overhead in terms of computation and communication. Furthermore, these tools may not always be compatible with the complex NN operations. Therefore, we believe that to design the privacy preserving variant of the classification tool, the actual classification model should be revisited and built while taking the privacy requirements into consideration, as well. Hence, we propose to follow a privacy-by-design approach and consider privacy requirements at the design phase of the neural network. We have identified the following three main challenges when building a neural network model customized for the use of privacy enhancing technologies:

- Large size of the NN: The size of the neural network directly depends on the size of the input and output vectors, the number of layers, and the number

of neurons in the model. These parameters have a significant impact on the complexity of the model. In order to reduce the overhead resulting from introducing the privacy-preserving variants of the underlying operations, the number of these operations, hence the size of the neural network has to be optimized. Such an optimization, on the other hand, should not have an impact on the actual accuracy of the model.

- Complex NN operations: A neural network involves various operations executed by each neuron during the classification phase. These include sophisticated operations such as sigmoid or hyperbolic tangent that may not be easily and efficiently supported by existing cryptographic tools. Hence, the underlying operations should be optimized and sometimes even transformed when designing the privacy-preserving variant of the neural network classification model.

- Real numbers instead of integers: Most of the operations in the neural network are executed over real numbers whereas cryptographic tools usually support integers. Therefore, there is a need for either supporting floating point numbers or approximating them to integers. Such an approximation should nevertheless not have a significant impact on the accuracy of the model.

To summarize, when designing a neural network model customized for the use of privacy enhancing technologies, one should address the trade-off between privacy, performance, and accuracy. The dedicated model should involve an optimized number of "simple" operations that advanced cryptographic tools can support while reaching a good accuracy level.

2.3 Solution Idea

In order to address the three main challenges identified in the previous section, we propose to build a neural network model from scratch. This approach is illustrated with a case study where a publicly available arrhythmia dataset is used. The design of the NN model is combined with secure two-party computation. Secure two-party computation (2PC) is a sub-problem of secure multiparty computation (MPC). MPC considers the problem of different parties jointly computing a function over their separate, private inputs without revealing any extra information about these inputs than what is leaked by the result of the computation, only. This setting is well motivated and captures many different applications to ensure privacy protection[5]. We propose to use ABY [3], a mixed-protocol framework that efficiently combines the use of Arithmetic shares, Boolean shares, and Yao's garbled circuits, to implement and evaluate the NN model. ABY supports many operations and provides novel, highly efficient conversions between different shares.

[5]Lectures 1&2: Introduction to Secure Computation, Yao's and GMW Protocols, Secure Computation Course at Berkeley University

As for the design of the appropriate model, we propose to define a small neural network with two fully connected (FC) layers, one activation layer, and one softmax layer. Experimental results (also detailed in the next section) show that this architecture is sufficient to achieve a good accuracy level. The number of intermediate neurons can be optimized based on several simulations evaluating the accuracy of a model for each case. Additionally, in order to reduce the number of input neurons, we propose to apply Principal Component Analysis (PCA) [2] and filter out the most significant inputs. Furthermore, because of the complexity of the activation functions, we propose to use the ReLU or square functions, only. These operations can be supported by 2PC more efficiently. The design of the new model customized for the use of 2PC should not result in a significant decrease on the accuracy of the classification. By definition, the accuracy of class x is the probability that the model predicts the class x knowing that the heartbeat belongs to class x ($P(predict = x|class = x)$). We use a confusion matrix to evaluate the accuracy of the model. For each $y \in Classes$ and $x \in Classes$ the confusion matrix presents the probability that the model predicts the class y knowing that the heartbeat belongs to class x ($P(predict = y|class = x)$).

All operations within the newly designed neural network model will be executed through a client-server system, whereby the client who could be considered as the patient (Data Subject) or the hospital (Data Controller) holds the input vector and the server (Data Processor) holds the model's parameters. The underlying protocol should therefore ensure the following:

- The secrecy of the input supplied by the client. This means that the client would like to get the prediction results without leaking any information about the heartbeat signal.
- The secrecy of the model parameters supplied by the server. The server cares about supplying the service to the client without leaking any information about the model parameters. We assume that the client knows the architecture of the model but not the parameters.
- The secrecy of the prediction results with respect to the server. The results of the prediction should only be accessible to the client and no additional information concerning it should be leaked to the server.

3 Privacy-preserving Neural Network Arrhythmia Classifier - A case study with PhysioBank

In this section, we describe the privacy by design approach in details and use the PhysioBank database to show concrete results on the accuracy and efficiency of the newly developed model.

3.1 The optimized neural network model

In order to ensure data privacy during the arrhythmia classification phase, a dedicated neural network model should be computed. Because the use of cryptographic primitives adds a non-negligible overhead, the complexity of the model should be optimized as much as possible. Hence, the primary goal while building a new prediction model, is to optimize the number of neurons at each layer while keeping an adequate accuracy level. As mentioned in the previous section, the cryptographic tool that we chose to ensure data privacy is 2PC [9] whereby the first party holds the input vector, and the second party has the NN prediction model, namely the weight matrices and bias vectors. Similarly to [11] and [12], non-linear operations such as the activation functions should be replaced with more efficient operations such as low degree polynomials. In this section, we describe our approach with a case study using the MIT-BIH arrhythmia dataset from the PhysioBank database[6]. The resulting neural network model is presented with an incremental approach.

We first extract heartbeats from the Electro-Cardiogram (ECG) signals. Each heartbeat is composed of 180 samples with 90 samples before the R-peak, 1 sample for the R-peak, and, 89 samples after the R-peak.

Table 1: Heartbeats for Arrhythmia classification and their frequency in our dataset

Arrhythmia Class	**Symbol**	**#**	**%**
Normal beat	N	14985	34.02%
Left bundle branch block beat	L	6450	14.64%
Right bundle branch block beat	R	5794	13.15%
Premature ventricular contraction	V	5712	12.97%
Paced beat	/	5608	12.73%
Atrial premature beat	A	2042	4.64%
Rhythm change	+	1005	2.28%
Fusion of paced and normal beat	f	786	1.78%
Fusion of ventricular and normal beat	F	647	1.47%
Ventricular flutter wave	!	378	0.86%
Nodal (junctional) escape beat	j	184	0.42%
Non-conducted P-wave (blocked APB)	x	155	0.35%
Aberrated atrial premature beat	a	123	0.28%
Ventricular escape beat	E	85	0.19%
Nodal (junctional) premature beat	J	68	0.15%
Atrial escape beat	e	26	0.06%

Once heartbeats were extracted, we have performed various filtering operations to create an appropriate dataset to build the neural network model. The PhysioBank database is shown in Table 4 in Appendix A and contains 23 different annotations for the extracted heartbeats. We have decided to only consider 16 out of 23 annotations representing meaningful arrhythmia classes that have significant number of instances in the dataset. Secondly, we realized that normal beats were dominating

[6]MIT-BIH Arrhythmia Database: `https://www.physionet.org/physiobank/database/mitdb/`

the dataset (67.3%) and hence resulting in an unbalanced dataset for model training purposes. We have reduced the number of normal beats in order for the model to predict anomalies more accurately while keeping this number sufficiently large so that it reflects reality. Moreover, we have used the over-sampling method to enforce the learning of low frequent classes such as class "e". Table 1 provides details about the final dataset we are actually using. This dataset is further split such that 80% of the heartbeats are used to train the network and 20% of the heartbeats are used to test the performance of the model. We propose a model with two fully connected layers involving matrix multiplications, one activation function and a final softmax function that would provide the resulting arrhythmia class.

Output layer: The number of neurons in the output layer corresponds to the number of arrhythmia classes. As shown in Table 1, we decide to take the first 16 out of the 23 arrhythmia classes in the studied dataset. Hence, the number of neurons in the output layer is set to 16.

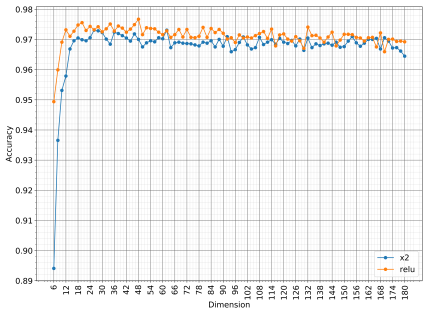

Figure 1: The accuracy of the model with different dimensions of the input vector

Hidden layers: In order to choose the appropriate number of neurons within the hidden FC layer, we have evaluated the accuracy of models on the validation dataset whereby the number of neurons varies from 2 to 100. We not only evaluate the overall accuracy but compute the confusion matrix that indicates the accuracy with respect to each arrhythmia class. We observe that although a model with more than 38 neurons in the hidden layer may show a slightly better accuracy (see Figure 6 in Appendix B), 38 is a better choice as this implies less complexity in the model as well as its corresponding confusion matrix shows better fairness toward less frequent classes. Hence, from Figure 6, we observe that the accuracy of our model is 96.51% on the test data. We represent the model's performance on the test dataset with the confusion matrix as illustrated in Figure 2. Moreover, the model presents a good precision value with 96.5%.

Furthermore, in addition to the optimization of the number of hidden neurons, we have to select the most appropriate activation function that cryptographic tools (in this case 2-party computation) can support. Although Figure 1 shows better accuracy results when ReLU is used, we opt for the use of the square function

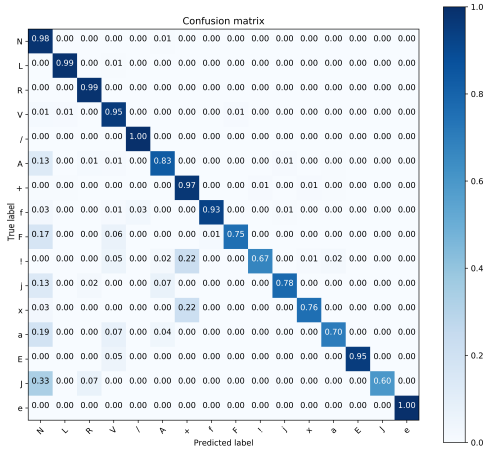

Figure 2: Confusion matrix of the model for each class in the test dataset

mainly for performance reasons. Indeed, the ReLU function involves comparison operations that can incur higher overhead compared to the square function, the resulting degradation is not very significant (0.34%). Finally, we replace the softmax function with a simple argmax operation since the exponentiation cannot be easily computed with 2PC. Note that this approximation does not incur any accuracy loss.
Input layer: The second parameter that affects the complexity of the NN model is the size of the input vector. This inherently reduces the dimension of the first matrix used for the FC layer. The main tool to adequately reduce the number of neurons of the input layer is the principal component analysis technique (PCA) [13]. PCA uses orthogonal linear transformations to find a projection of all input data (ECG heartbeats) into k dimensions which are defined as the principal components. Hence, PCA outputs the k features with the largest variance. The first k eigenvectors of the covariance matrix (of the dataset) are the target dimensions. The efficiency of using PCA for the ECG analysis domain has also been proved in [2]. It also helps reduce the noise in the ECG signals and hence improve the accuracy of the model. This is due to the fact that dimensions with low variance noise are automatically discarded.

To identify the appropriate number of eigenvalues we run a simulation with 100 hidden neurons and change the value of the input size n starting from $n = 180$. The same simulation is executed using the ReLU and the square operations as for the activation functions. The choice of these two functions instead of more sophisticated functions such as the widely used sigmoid function is due to their simplicity and hence their easy integration. The results of the simulation - in terms of the accuracy measured on the validation data - are as shown in Figure 1. We observe that reducing the dimension of the input data can sometimes increase the accuracy of the prediction model. This is mainly due to the existence of low variance noise in the ECG heartbeats. From this analysis, we choose to set the input size

to 16, mainly because the resulting prediction model provides good accuracy with acceptable complexity. Hence, the number of neurons of the input layer is now set to 16.

The resulting model: To summarize, the developed model, compatible with the use of 2PC, consists of 2 fully connected layers, one activation layer implementing a square function and one softmax function. The architecture of the proposed neural network model is as shown in Figure 3. The first layer consists of a fully

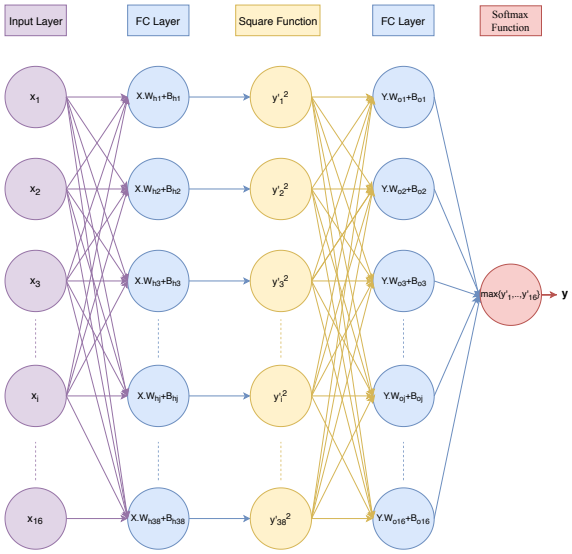

Figure 3: The proposed NN model

connected layer and its main operations are given in Equation (1):

$$Y'_h = X^T.W_h + B_h \tag{1}$$

where X represents the input vector (PCA transform of a heartbeat). This input vector X of size 16 is multiplied with the weight matrix of the hidden layer, W_h, of size 16×38. This intermediate result is further added to the bias vector of the hidden layer, B_h, of size 38.

The resulting vector Y'_h becomes the input of the activation layer which consists of computing the square of each element y'_{h_i} of Y'_h described in (2).

$$Y_h = \begin{bmatrix} {y'_{h1}}^2 \\ {y'_{h2}}^2 \\ \vdots \\ {y'_{h38}}^2 \end{bmatrix} \tag{2}$$

The resulting vector Y_h is the final output of the hidden layer. This vector further becomes the input for another FC layer as shown in (3):

$$Y' = Y_h^T.W_o + B_o \tag{3}$$

W_o, B_o Y' denote the weight matrix, the bias vector and the result of the output layer, respectively. The output is the vector Y' of size 16. Finally, a softmax function is executed over the components y'_j of Y'. The aim of this function is to mainly identify the actual predicted class (the one that shows the greatest probability). The result y is the index of one of the 16 arrhythmia classes as given in Equation (4).

$$y = \max_j \frac{e^{y'_j}}{\sum_{i=1}^{16} e^{y'_i}} \text{ for } j = 1, 2 \dots 16 \tag{4}$$

In total, the prediction phase consists of: $16 \times 38 + 38 \times 16 + 38 = 1254$ multiplications, $15 \times 38 + 38 + 37 \times 16 + 16 = 1216$ additions, 16 exponentiations, 16 divisions and 1 argmax operation.

Discussion on Principle Component Analysis:

As previously mentioned, the NN model is revised and designed from scratch in order to be compatible with 2PC and remain as efficient as possible. To improve the performance of the classification phase, the size of the input is reduced using the PCA method. Principle component analysis (PCA) is a statistical method which identifies patterns, highlights similarity between elements within the dataset and finally reduces the dimension of the feature space. More formally, let S be a dataset and x_i an element of it with dimension d. The first step of PCA consists of computing the mean μ of all the elements x_i. Then the covariance matrix A of S is computed. Evidently, A will have the dimension $d \times d$. The eigenvectors and corresponding eigenvalues of matrix A are further evaluated and the first k eigenvectors with the largest eigenvalues are selected. Thus, the final output of this method is a $d \times k$ matrix of the most relevant eigenvalues.

We propose to make use of the PCA method to decrease the size of the input vector. Thus, the client would transmit less data to the server when sending the input to the model. At this step, the server will first compute the mean μ of his dataset. Then it will compute the covariance matrix from the training dataset (for this case study, 44048 heartbeats consisting of 180 samples) and obtain the 180×16 matrix of the most relevant eigenvalues. This matrix along with the vector μ is sent to the client who reduces the dimension of its input to 16 (instead of 180) by first normalizing his signal by subtracting it with the mean vector and further multiplying the result with the received matrix. Consequently, this operation has already reduced the complexity of the proposed NN.

Nevertheless, the use of the PCA transformation at the client side can result in some information leakage. We propose to analyse which information and how much information is leaked, and introduce two design approaches to avoid towards the information leakage. The leakage resulting from the use of PCA is represented by two components: the mean of the dataset and the 180×16 covariance matrix. The mean of all the signals in the training dataset does not carry any valuable information since the labels of the training signals are not included in the computation

of the mean. On the other hand, the matrix of 16 eigenvectors does not correspond to the entire matrix of eigenvalues. In addition to that, without the knowledge of the eigenvalues there exist an infinite number of inverse transformations back to the original covariance matrix. Therefore, one cannot discover the training dataset and hence the model from this reduced and transformed matrix. If we choose not to leak this information while designing the privacy-preserving NN classification, then either we do not use PCA (high bandwidth and computational cost) or include the PCA steps to the 2PC solution (additional overhead but less costly). Accordingly, in this work, we propose the following three design approaches for PAC (as shown in Figure 4) and evaluate the performance for each of them:

- Model 1: PCA is not integrated to 2PC (original and most efficient solution implies some leakage),
- Model 2: PCA is integrated to 2PC (less efficient but no leakage),
- Model 3: PCA is not used (worse performance but no leakage).

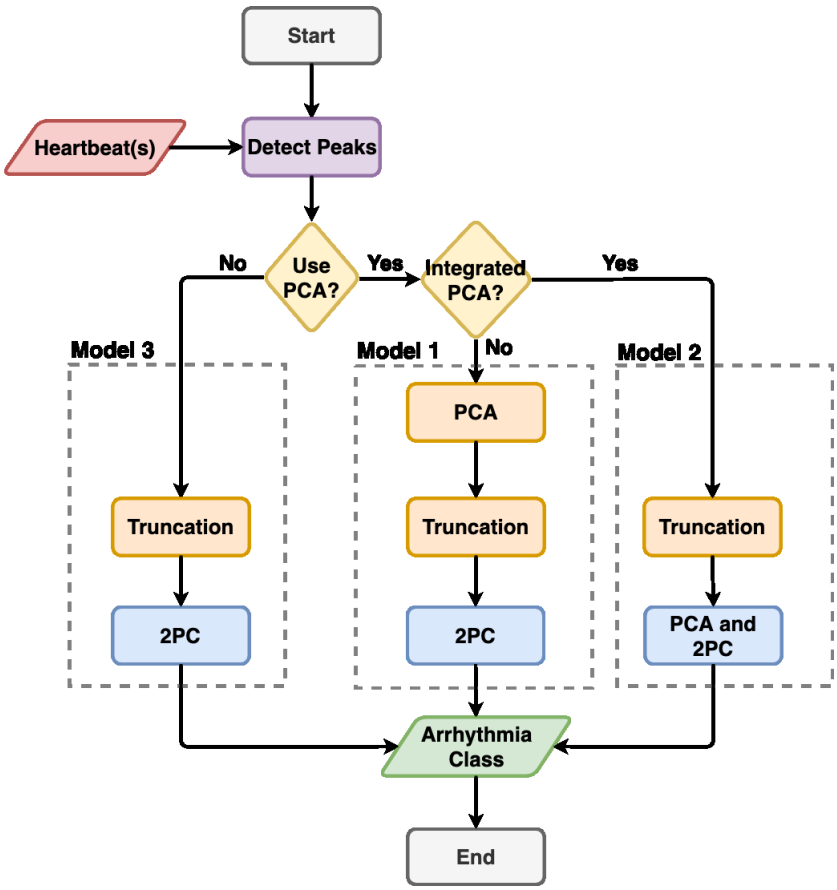

Figure 4: PAC Overview

Model 1 and Model 2 are similar: the only difference among them is on the integration of PCA into the 2PC (no information leakage if Model 2 is used). Model 3 is customized to support 2PC without disclosing any information. When implementing Model 3, one can follow the similar idea of having less complex NN operations and less NN complexity that we make use of when building models 1 and 2. The resulting model has the same architecture as models 1 and 2 in terms of NN layers and the size of the output layer but the size of the input (180-sample instead of 16) and the number of the hidden neurons (40 instead of 38 neurons) are much higher. In the sequel of this section, we only describe the implementation of Model 1. Nevertheless, the three models for PAC are implemented and performance results are provided in the next section.

SIMD circuits: In addition to reducing the size of the neural network and decreasing the cost of the underlying operations, we also take advantage of Single Instruction Multiple Data (SIMD) circuits which allow the packing of multiple data elements and the execution of operations in parallel. We use this technique to perform the matrix multiplications and additions more efficiently. In more details, since the number of hidden neurons is 38, the client creates the SIMD version of its input X (of size 16) repeated 38 times (i.e. the size of the share is $38 * 16 = 608$). Similarly, the server creates a SIMD version of the weight matrices W_h (of size 16x38) and W_o (of size 38x16) by flattening them to two vectors of 608 elements. Once these versions obtained, one single SIMD multiplication gate can be used to perform element-wise multiplication. Next, to finalize the matrix multiplication, some elements of the resulting vector should also be added. The server also creates a SIMD version of the bias vectors and adds them to the vector resulting from the previous SIMD matrix multiplication. The square activation function can also be computed using one SIMD multiplication gate. To implement the argmax function, we transform the SIMD share of the previous layer to a non-SIMD share (i.e., the SIMD share is composed of 1 wire holding all the 16 values of Y' while the non-SIMD share is composed of 16 wires each wire will hold one value of Y'). Due to the inability of a comparison gates to compare between negative and positive values, we use a comparison gate to check the sign of the value by comparing it with the smallest negative number, namely -1, and then we replace negative values with a zero using a multiplexer gate. Then, we loop on all the values (wires), and compare each of them with the max value using a comparison gate. Eventually, in each iteration two multiplexer gates are used to store the max value and max index relying on the result of the comparison gate. Hence, Equation (1) involves 1 SIMD multiplication and 16 SIMD additions; the activation function consists of 1 SIMD multiplication; further, Equation (3) is computed using 1 SIMD multiplication and 38 SIMD addition gates; finally, to evaluate the argmax, 46 multiplexer gates and 31 comparison gates are used.

Moreover, we propose a secure computation of PCA in Model 2. As described in Section 3.1, the computation of the PCA can eventually introduce a limited leakage of the training dataset. Therefore, a solution might be deployed by introducing the computation of the PCA vector to the 2PC model. For this to happen, the server

shares the mean of the dataset as well as the 16 eigenvectors using ABY (the same way it shares the weights and biases). On the other hand, the client shares its sampled heartbeat signal. The two parties will first collaboratively compute PCA of the signal while the rest part of the circuit remains unchanged. This PCA computation layer adds 181 SIMD addition gates and 1 SIMD multiplication gate.

3.2 PAC: Detailed description

As previously mentioned, we propose to use 2PC to obtain the privacy-preserving variant of the arrhythmia prediction model, i.e., PAC. Since the underlying model involves several different operations (such as additions, multiplications and comparisons), we propose to use the ABY framework which supports all basic operations in a flexible manner using Arithmetic, Boolean or Yao's circuits. ABY supports Single Instruction Multiple Data (SIMD) gates. Furthermore, the current ABY implementation[7] also supports floating point representation if Boolean circuits are used. Hence, we first implement the privacy-preserving model using Boolean circuits.

PAC with Boolean shares: The first solution translates the NN model regrouping the four previously described equations into Boolean circuits. Both the input vector and the model are represented with matrices and vectors with floating points which values are represented as doubles (64 bits variables). Each Boolean share consists of 64 wires. When working with floating point numbers, ABY builds a specific circuit for each operational gate: For example, one floating point multiplication gate consists of 3034 XOR gates, 11065 AND gates and 3 MUX gates. Consequently, the total number of gates in the resulting circuit becomes 553,925 and the depth of the circuit is evaluated as 4,513.

PAC with Arithmetic shares: Multiplication and addition of Boo-lean shares are much more time and bandwidth consuming compared to multiplication and addition of arithmetic shares. We therefore consider the use of arithmetic circuits only, and represent real numbers with fixed-point numbers. As the multiplication of two fixed-point numbers can yield numbers with a number of bits higher than the two initial numbers, hence to an overflow, these numbers need to be truncated and/or rounded in order to ensure that all intermediate values can be represented in 64 bits. We mainly propose two truncation methods: The first method consists of applying truncation at intermediate stages in the circuit and hence try to keep a good accuracy level whereas the second method truncates the inputs before the prediction process starts, only. In this section, we only present the second truncation method since it shows better performance gains. The description of the first truncation method is given in Appendix C. The problem with the first approach is that it implies the modification of the actual circuit, specifically adding shifters between layers. Although the shifters themselves do not add any overhead on the resulting model, but since shifters can only process Boolean shares this introduces

[7] `https://github.com/encryptogroup/ABY`

an arithmetic to Boolean conversion gate before every shifter and vice versa after it. The conversion gates add significant delay in the computational time as well as it increases the amount of data transferred between the client and the server. Still, its good to mention that in general the first approach is more scalable as it does not imply any restrictions on the number of layers of the computed model. But since we have chosen our model, which happens to be a 2-layered model, we consider a more simple truncation approach for which only the inputs to the circuits are truncated and this before the actual execution of the circuit. Thanks to this new approach, the actual circuit only consists of arithmetic gates except at the last stage where an argmax operation needs to be executed. In order to avoid overflows, we multiply X, W_o and W_h by 10^3, B_h by 10^6 and B_o by 10^{15} and truncate the fractional part afterwards. We observe that this method is as safe as the maximum number a signed 64-bit integer variable can take is $9.223372037 \times 10^{18}$ and the upper bound for the values of Y' is $9,223$ and the lower bound is $-9,223$. We observe that the risk of overflow is very low.

We have tested the accuracy of the new model using the test dataset and we have achieved an accuracy of 96.34% which is very close to the accuracy of the original model (96.51%). The confusion matrix of the new model shows the same accuracies as presented in the original model (Figure 2).

Regarding the complexity of the circuit, thanks to the use of arithmetic gates only, the number of gates is reduced from 553,925 to 34,329 and the depth is reduced from 4,513 to 146. Our final circuit is composed of an arithmetic circuit represented in the diagram as shown in Figure 5 followed by a Boolean circuit representing the argmax layer. On the other hand, with the first truncation approach, the number of gates is reduced from 553,925 to 35,477 whereas the total depth is reduced from 4,513 to 160. The decrease in the number of gates and the depth of the circuit is due to the use of arithmetic gates instead of Boolean gates for multiplications and additions. Nevertheless, this decrease remain greater in the second version because of the use of conversion gates in the first version to switch between Arithmetic and Boolean shares in order to perform the shifting operations.

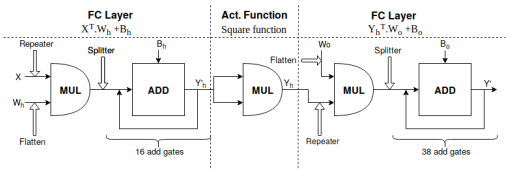

Figure 5: Arithmetic circuit representation of the model with Truncation v2

To evaluate the computational and communication overhead of the model, experiments were carried out by a computer which has four 3.60GHz Intel Core i7-7700 processors, 32GB of RAM acting as the server and a laptop which has two 1.70GHz Intel Core i5-4210U processors, 4 GB of RAM acting as the client. On the other hand, the client and the server communicate through a local area network (LAN). The client is connected to the LAN through a wireless access point. A

Table 2: Performance results for each model

	Boolean Circuits	Truncation v1		Truncation v2			
Proposed NN models	**Model 1**	**Model 1**	**Model 2**	**Model 1 without ARGMAX**	**Model 1**	**Model 2**	**Model 3**
Circuit							
Gates	553925	35477	36418	128	34329	34696	34660
Depth	4513	160	168	5	146	147	146
Time (ms)							
Total	117571.82	1218.613	2776.862	735.357	1082.804	2641.846	4723.203
Init	0.046	0.076	0.071	0.056	0.062	0.037	0.033
CircuitGen	0.046	0.074	0.062	0.067	0.078	0.055	0.047
Network	272.867	268.39	94.142	248.92	51.391	89.46	34.221
BaseOTs	288.047	309.288	310.06	311.387	291.705	294.698	298.294
Setup	107481.557	851.397	2373.818	714.511	817.807	2354.391	4409.689
OTExtension	106645.796	847.424	2369.377	714.278	816.069	2351.584	4407.521
Garbling	812.573	2.502	3.268	0.002	1.405	1.851	1.252
Online	10090.26	367.21	403.042	20.844	264.995	287.453	313.512
Data Transfer (Sent/Rcv, in KB)							
Total	319269 / 309573	2629 / 2252	7113 / 6651	1910 / 1900	2171 / 2095	6560 / 6461	12266 / 12139
BaseOTs	48 / 48	48 / 48	48 / 48	48 / 48	48 / 48	48 / 48	48 / 48
Setup	305915 / 304815	2240 / 2227	6591 / 6579	1881 / 1881	2086 / 2071	6406 / 6391	12025 / 12010
OTExtension	301095 / 304815	2057 / 2227	6377 / 6579	1881 / 1881	2053 / 2071	6373 / 6391	11992 / 12010
Garbling	4819 / 0	183 / 0	214 / 0	0 / 0	33 / 0	33 / 0	33 / 0
Online	13354 / 4757	389 / 25	522 / 72	29 / 19	85 / 24	154 / 70	240 / 129

simulation of the bandwidth and the latency of the connection between the client and the server showed the values of 39Mbit/sec for the bandwidth and 3.36 ms for the latency. Furthermore, we run the client and the server on two separate processes communicating through the localhost of the same computer, specifically the one with the four 3.60GHz Intel Cores to evaluate the performance of the model without considering the limitation of the bandwidth. In ABY, we set the security parameter to 128 bits.

Table 2 shows the performance results in terms of prediction time and bandwidth consumption for the original Boolean circuits as well as for both truncation approaches (namely Truncation v1 and v2) with PCA integrated and not integrated into the 2PC. We further evaluate the model implemented by making use of the Model 2 and not use of the PCA method, Model 3. Thus, we implement Model 3 to compare with Model 2. Moreover, we have repeated all evaluations on the local set-up, i.e., the localhost on one machine, to give an insight about the overhead incured by the low bandwidth (the results are given in the table 5 in Appendix D). The Network time in the table of performance results represents the time for the connection to be established. The Total time corresponds to the Setup time and the Online time; moreover, the Setup time corresponds to the OTExtension time and the Garbling time. Thus, the prediction time of one heartbeat is the Total time added to the baseOT time.

Furthermore, we have evaluated the performance of the prediction model without using any privacy enhancing technologies and making use of Tensorflow[8]. It takes 7.29ms to predict a heartbeat in cleartext while this value becomes 117,859 ms with the privacy-preserving protocol, PAC (without any truncation). Never-

[8] `https://www.tensorflow.org/`

theless, from Table 2, we observe some significant performance gain in terms of computational and communication cost by employing the two truncation methods. Compared to the model built with Boolean circuits, the Total time with the second truncation method is reduced by a factor of 108. As expected, the second truncation method shows some improvement over the first truncation method in terms of time and bandwidth consumption. This consumption resulted from the conversion of the Boolean and Arithmetic gates causes overhead to the model with Truncation v1.

From Table 2, Model 2 still provides a better results than Model 3, i.e., building the NN model without utilising the PCA method. In details, the time and bandwidth consumption of Model 3 is larger with a factor of 1.8 than Model 2 and as previously mentioned, the PCA method reduces the noise in the ECG signal and results in gaining the accuracy. We have also implemented Model 2 without the argmax layer (the output is a vector of 16-value where its argmax can be computed locally by the client) to show the size and performance of the arithmetic circuit without introducing any Boolean gates. Finally, the time performance presented in the table is highly affected by the low bandwidth of the communication channel between the client and the server. The time performance difference can be easily seen by comparing the results in Table 2 with the one in Table 5 in Appendix D which represent the result of when the client and the server resides on the same machine and so no bandwidth limitation can affect the result. We observe that the time consumption of the model evaluated locally on the same machine can reach 39.785 ms which is 27 times less than the remotely evaluated model. This limitation comes from the core of the 2PC protocol which suffers from high bandwidth consumption. We believe this problem can be easily solved by a decent connection between the client and the server.

4 PAC in batches

In this section, we propose to perform arrhythmia predictions in batches, namely, with several heartbeat inputs. This can be justified as the classification of a single heartbeat may not be sufficient to diagnose the disease for a patient and the doctor may need to receive the classification of the n consecutive heartbeats. In the sequel of this section, we describe this new solution that uses the SIMD technique once again and show that the performance can be improved even more.

4.1 Description

Experiment results show that the use of Arithmetic circuits as an optimization together with the second truncation method (Truncation v2) significantly improves the performance. We also realize that, with this optimization, the online time remains short relatively and that 21.2% (82.2% in case of evaluation on the same machine) of the Total time corresponds to the BaseOT phase. This phase is only

processed once the two parties initiate the protocol. Hence the overall time may again be decreased by performing predictions in batches (i.e. performing prediction of many ECG signals at once) using the SIMD technique, once again.

Indeed, the client may first record N signals, prepare the inputs and further store them in a matrix $S(N)$. Let $x_{i,j}$ be the value of the j^{th} PCA component of the signal belonging to the individual i in the dataset.

$$S(N) = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,16} \\ x_{2,1} & x_{2,2} & \dots & x_{2,16} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \dots & x_{N,16} \end{bmatrix}$$

The client further creates the following vector $X(N)$ from $S(N)$.

$$X(N) = \underbrace{\Big[\underbrace{s_1 \quad \dots \quad s_1}_{38} \quad \dots \quad \underbrace{s_N \quad \dots \quad s_N}_{38}\Big]}_{16 \times 38 \times N}$$

On the other hand, the server creates the weight matrix vector $W_h(N)$ from the original weight matrix W_h of the hidden layer.

$$W_h(N) = \left.\begin{bmatrix} w_{h1} \\ \vdots \\ w_{h38} \\ \vdots \\ w_{h1} \\ \vdots \\ w_{h38} \end{bmatrix}\right\} 16 \times 38 \times N \text{ where } W_{hi} = \begin{bmatrix} w_{h1,i} \\ w_{h2,i} \\ \vdots \\ w_{h16,i} \end{bmatrix}$$

Similarly, the output layer's SIMD weight vector $W_o(N)$ transforms the weight matrix of the output layer W_o as follows:

$$W_o(N) = \left.\begin{bmatrix} w_{o1} \\ \vdots \\ w_{o16} \\ \vdots \\ w_{o1} \\ \vdots \\ w_{o16} \end{bmatrix}\right\} 38 \times 16 \times N \text{ where } W_{oi} = \begin{bmatrix} w_{o1,i} \\ w_{o2,i} \\ \vdots \\ w_{o38,i} \end{bmatrix}$$

The server also creates the vectors $B_h(N)$ and $B_o(N)$ from the two bias vectors B_h and B_o, respectively.

$$B_h(N) = \left.\begin{bmatrix} b_{h1} \\ \vdots \\ b_{h38} \\ \vdots \\ b_{h1} \\ \vdots \\ b_{h38} \end{bmatrix}\right\} 38 \times N \; B_o(N) = \left.\begin{bmatrix} b_{o1} \\ \vdots \\ b_{o16} \\ \vdots \\ b_{o1} \\ \vdots \\ b_{o16} \end{bmatrix}\right\} 16 \times N$$

The Arithmetic circuit of the NN model is implemented as illustrated in Figure 5 whereby only the structure of the inputs and output differ (ie. SIMD vectors result in larger size). The number of SIMD multiplications does not change since all values are regrouped in one SIMD share and the multiplication is further performed. The number of SIMD additions also remains the same. For the sake of clarity, we describe the following matrix whereby each column represents one of the 16 SIMD shares ($1 \leq i \leq N$ and $1 \leq j \leq 38$):

$$\left.\begin{bmatrix} x_{1,1}.w_{h1,1} & x_{1,1}.w_{h2,1} & \dots & x_{1,16}.w_{h16,1} \\ x_{1,1}.w_{h1,2} & x_{1,2}.w_{h2,2} & \dots & x_{1,16}.w_{h16,2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,1}.w_{h1,38} & x_{1,2}.w_{h2,38} & \dots & x_{1,16}.w_{h16,38} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i,1}.w_{h1,j} & x_{i,2}.w_{h2,j} & \dots & x_{i,16}.w_{h16,j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1}.w_{h1,1} & x_{N,2}.w_{h2,1} & \dots & x_{N,16}.w_{h16,1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1}.w_{h1,38} & x_{N,2}.w_{h2,38} & \dots & x_{N,16}.w_{h16,38} \end{bmatrix}\right\} 38 \times N$$

After adding the 16 SIMD shares, we add the bias to complete the evaluation of (1). The square function is computed by one simple SIMD multiplication as described before. The result of equation (2) will thus be:

$$Y_h(N) = \begin{bmatrix} y_{h1,1} \\ \vdots \\ y_{h1,38} \\ \vdots \\ y_{hN,1} \\ \vdots \\ y_{hN,38} \end{bmatrix} = \left.\begin{bmatrix} y_{h1} \\ y_{h2} \\ \vdots \\ y_{hN} \end{bmatrix}\right\} 38 \times N$$

A new vector $Y_h^{new}(N)$ is further created in order to be able to evaluate (3) with the same number of SIMD multiplication and SIMD addition gates.

$$Y_h^{new}(N) = \left.\begin{bmatrix} y_{h1} \\ \vdots \\ y_{h1} \\ \vdots \\ y_{hN} \\ \vdots \\ y_{hN} \end{bmatrix}\right\} 16 \times 38 \times N$$

For the next layer, the same procedure performed in the hidden layer is repeated: Namely, starting from the vector $Y_h^{new}(N)$, at the output we get the vector $Y(N)$ of length $16 \times N$ which holds the outputs of the prediction of each signal.

Finally, the Boolean circuit which represents the argmax layer is also performed with SIMD gates. Values of each class in each individual output in the vector $Y(N)$ are grouped in seperate SIMD vectors. The the same comparison and multiplexers gates described before will be used but this time the inputs are SIMD shares. The output of the Boolean circuit will be the vector $y(N)$ where each value represents the index of the class of the corresponding heart beat.

Table 3: Performance results for the multi-signal model

	# Input signals				
	1	**10**	**100**	**200**	**400**
Circuit					
Gates	33741	39552	40918	42422	45426
Depth	148	148	148	148	148
Time (ms)					
Total	1084.713	8002.287	77867.26	160114.6	314311
Init	0.061	0.09	0.062	0.059	0.058
CircuitGen	0.041	0.043	0.052	0.053	0.066
Network	7.115	7.681	7.513	5.34	4.307
BaseOTs	290.672	294.094	293.867	300.302	285.169
Setup	814.036	7575.32	75821.76	155921.4	306985
OTExtension	808.49	7509.797	75149.46	154673.2	304616
Garbling	5.056	62.455	650.642	1214.046	2310
Online	270.673	426.961	2045.492	4193.194	7325.77
Bandwidth (Rcv/Sent in KB)					
Total	2095 / 2167	21010 / 21652	209912 / 216247	419805 / 432465	839588 / 864898
BaseOTs	48 / 48	48 / 48	48 / 48	48 / 48	48 / 48
Setup	2071 / 2084	20782 / 20936	207647 / 209107	415276 / 418186	830533 / 836342
OTExtension	2071 / 2053	20782 / 20612	207647 / 205947	415276 / 411876	830532 / 823732
Garbling	0 / 31	0 / 315	0 / 3159	0 / 6309	0 / 12609
Online	24 / 83	227 / 716	2264 / 7139	4528 / 14278	9055 / 28555

4.2 Evaluation

We have run experiments with different batch sizes using the local and remote setups. The results are given in Table 3 for the remote setup and Table 6 in Appendix D for the local setup. We can observe that the number of gates y increases

slightly with respect to the number of signals, which is much better than performing prediction on signals individually which will cost $y = 34329N$ gates. Also, the depth is constant in this model regardless of the number of input signals the model predicts.

The time performance for different number of signals is as shown in Table 3. We observe that the Total time t still increases linearly with the number of signals but with a much better rate. More specifically, the batches model can decrease the time consumption with a percentage of 27% (70% in case of local evaluation) compared to performing prediction on signals one by one which takes $t = 1082.8N$ ms ($t = 39.7N$ in case of local evaluation). Finally, the BaseOT time is approximately 290ms and remains constant regardless to the number of input signals the model predicts. This is, again, much better than performing prediction on signals, one by one, where the BaseOT time bt costs $bt = 290$ ms. Table 3 also shows that the data grows perfectly linear with the number of signals.

To summarize, prediction in batches does improve performance in terms of computational cost but the size of the batch should be bounded according to bandwidth limitations.

5 Related Work

Existing privacy-preserving neural network classification techniques mostly focus on Convolutional Neural Networks (CNN) with the goal of classifying images and achieve data privacy using homomorphic encryption [11, 12, 14–20] or secure multi-party computation [21–29] or both techniques [30–32] or the trusted hardware [33–35]. Similarly to our work, these methods aim at reducing the complexity of the underlying neural networks. However, the application scenarios (image classification with CNNs) significantly differ from ours (arrhythmia classification). Thus, the models in previous work are not easily comparable to our solution since authors are dealing with more complex NNs that use convolutions. Furthermore, datasets containing images (hence the use of CNNs) are different.

Differently to our work, Jun et al. [36] propose a CNN based on ECG arrhythmia classification. ECG signals are taken from the MIT-BIH arrhythmia database and converted into the images that are made use of inputs of the deep CNN. Although this work is very interesting, authors did not use any privacy enhancing technologies to ensure the data privacy.

The closest study to our work is [37] which specifically focuses on privacy-preserving arrhythmia classification with neural networks. Authors in [37] use 2PC combined with a partially homomorphic encryption [7]. Their protocol is executed between the client who protects the input vector and the server who stores the model. Similarly to our model, their neural network is also composed of two layers: one hidden layer with SATLIN (a symmetric saturating linear) as an activation function and the output layer implementing the argmax function to decide on the arrhythmia class. Although this work uses the same dataset from the PhysioBank

datasets as we do, authors achieve an accuracy of 86.3% whereas PAC reaches 96.43% for the classification of each heartbeat. Furthermore, whilw our model seems slightly more complex than the one in [37] (38+16=54 neurons instead of 6+6=12 neurons), it shows better accuracy and performance results: The communication channel used in [37] is set to 1 Gbit/sec which is much larger than our bandwidth estimation (39 Mbit/sec). Authors evaluated the timing complexity to be about 7 seconds whereas our solution predicts one heartbeat in 1 second within a more realistic scenario (less performance in the client-side and lower bandwidth). This may be considered acceptable in applications wherein heartbeats are classified at the same pace at which they are produced. Additionally, the accuracy of the predicted heartbeat is higher and further, the number of output neurons is set to 16, PAC detects more arrhythmia classes. Moreover, the solution in [37] combines the use of homomorphic encryption with garbled circuits. The use of both techniques renders the prediction protocol more time consuming compared to PAC whereby mostly arithmetic circuits are used. Finally, while both solutions use packing at the encryption stage and thus allow for prediction in batches, our solution additionally parallelizes each operation in the model (using the SIMD packing method, once again) and hence optimizes the timing complexity.

6 Conclusion

In this paper, we have presented PAC, a new Privacy-preserving Arrhythmia Classification that keeps users' ECG data confidential against service providers and the neural network model confidential against users. As a case study, we have designed a new model based on the PhysioBank dataset. The proposed model involves a customized two-layer neural network with 54 neurons. This model was built from scratch in order to be compatible with 2PC. The solution is implemented with the ABY framework which required the truncation of input values and model parameters. The second truncation method combined with Arithmetic circuits consists of multiplying the input values with 10^3 and shows significant improvement in terms of performance and accuracy. PAC achieves an accuracy of 96.34% and experimental results show that the prediction of one heartbeat takes 1 second in real world scenarios. We show that more savings can be achieved with the use of SIMD for performing predictions in batches. Looking forward, we plan to release the source code of PAC.

References

[1] Ponemon Institute, "Cost of a data breach study: Global overview," 2018. [Online]. Available: https://www.ibm.com/security/data-breach

[2] F. Castells, P. Laguna, L. Sörnmo, A. Bollmann, and J. M. Roig, "Principal Component Analysis in ECG Signal Processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 74580, 2007.

[3] D. Demmler, T. Schneider, and M. Zohner, "ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, 2015.

[4] R. E. Kass and C. E. Clancy, "Basis and Treatment of Cardiac Arrhythmias," 2006.

[5] C. Gentry, " Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, (STOC)*, 2009, pp. 169–178.

[6] ——, "A Fully Homomorphic Encryption Scheme," 2009.

[7] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology - EUROCRYPT*, 1999, pp. 223–238.

[8] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*. ACM, 2012, pp. 309–325.

[9] Y. Lindell, "Secure Multiparty Computation for Privacy-Preserving Data Mining," 2008. [Online]. Available: https://eprint.iacr.org/2008/197.pdf

[10] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay - Secure Two-Party Computation System," in *USENIX Security Symposium*, vol. 4, 2004, p. 9.

[11] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-Preserving Classification on Deep Neural Networks," 2017. [Online]. Available: http://eprint.iacr.org/2017/035

[12] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16, 2016, pp. 201–210.

[13] I. T. Jolliffe, "Principal Component Analysis," 2002.

[14] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *CoRR*, vol. abs/1711.05189, 2017. [Online]. Available: http://arxiv.org/abs/1711.05189

[15] A. Ibarrondo and M. Önen, "FHE-compatible Batch Normalization for Privacy Preserving Deep Learning," in *Proceedings of the 13th Data Privacy Management Workshop (DPM)*, 2018.

[16] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*. Springer, 2018, pp. 483–512.

[17] A. Sanyal, M. J. Kusner, A. Gascón, and V. Kanade, "TAPAS: tricks to accelerate (encrypted) prediction as a service," *CoRR*, vol. abs/1806.03461, 2018. [Online]. Available: http://arxiv.org/abs/1806.03461

[18] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy-preserving machine learning as a service," *PoPETs*, vol. 2018, no. 3, pp. 123–142, 2018.

[19] X. Jiang, M. Kim, K. E. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. ACM, 2018, pp. 1209–1222.

[20] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-Fei, "Faster cryptonets: Leveraging sparsity for real-world encrypted inference," *CoRR*, vol. abs/1811.09953, 2018. [Online]. Available: http://arxiv.org/abs/1811.09953

[21] P. Mohassel and Y. Zhang, "SecureML: A System for Scalable Privacy-Preserving Machine Learning," in *IEEE Symposium on Security and Privacy*, 2017, pp. 19–38.

[22] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious Neural Network Predictions via MiniONN Transformations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 619–631.

[23] P. Mohassel and P. Rindal, "Aby^3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, 2018, pp. 35–52.

[24] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: scalable provably-secure deep learning," in *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*, 2018, pp. 2:1–2:6.

[25] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*. ACM, 2018, pp. 707–721.

[26] M. Dahl, J. Mancuso, Y. Dupis, B. Decoste, M. Giraud, I. Livingstone, J. Patriquin, and G. Uhma, "Private machine learning in tensorflow using secure computation," *CoRR*, vol. abs/1810.08130, 2018. [Online]. Available: http://arxiv.org/abs/1810.08130

[27] M. Ball, B. Carmer, T. Malkin, M. Rosulek, and N. Schimanski, "Garbled neural networks are practical," Cryptology ePrint Archive, Report 2019/338, 2019, https://eprint.iacr.org/2019/338.

[28] N. Chandran, D. Gupta, A. Rastogi, R. Sharma, and S. Tripathi, "Ezpc: Programmable, efficient, and scalable secure two-party computation for machine learning," in *IEEE European Symposium on Security and Privacy*. (IEEE EuroS&P 2019), 2019.

[29] S. Wagh, D. Gupta, and N. Chandran, "Securenn: Efficient and private neural network training," in *Privacy Enhancing Technologies Symposium*. (PETS 2019), February 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/securenn-efficient-and-private-neural-network-training/

[30] M. Barni, C. Orlandi, and A. Piva, "A privacy-preserving protocol for neural-network-based computation," in *Proceedings of the 8th workshop on Multimedia & Security, MM&Sec 2006, Geneva, Switzerland, September 26-27, 2006*. ACM, 2006, pp. 146–151.

[31] C. Orlandi, A. Piva, and M. Barni, "Oblivious neural network computing via homomorphic encryption," *EURASIP J. Information Security*, vol. 2007, pp. 1–11, 2007.

[32] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1651–1669.

[33] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 619–636.

[34] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," *CoRR*, vol. abs/1803.05961, 2018. [Online]. Available: http://arxiv.org/abs/1803.05961

[35] F. Tramèr and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. [Online]. Available: https://openreview.net/forum?id=rJVorjCcKQ

[36] T. J. Jun, H. M. Nguyen, D. Kang, D. Kim, D. Kim, and Y. Kim, "ECG arrhythmia classification using a 2-d convolutional neural network," *CoRR*, vol. abs/1804.06812, 2018. [Online]. Available: http://arxiv.org/abs/1804.06812

[37] M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Privacy-preserving ecg classification with branching programs and neural networks," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 6, no. 2, pp. 452–468, June 2011.

A The arrhythmia dataset from the PhysioBank database

Table 4: Heartbeats for Arrhythmia classification and their frequency from the PhysioBank database

Arrhythmia Class	**Symbol**	**#**	**%**
Normal beat	N	59926	66.593%
Left bundle branch block beat	L	6450	7.168%
Right bundle branch block beat	R	5794	6.439%
Premature ventricular contraction	V	5712	6.347%
Paced beat	/	5608	6.232%
Atrial premature contraction	A	2042	2.269%
Rhythm change	+	1005	1.117%
Fusion of paced and normal beat	f	786	0.873%
Fusion of ventricular and normal beat	F	647	0.719%
Signal quality change	∼	508	0.565%
Ventricular flutter wave	!	378	0.42%
Comment annotation	”	352	0.391%
Nodal (junctional) escape beat	j	184	0.204%
Non-conducted P-wave (blocked APB)	x	155	0.172%
Aberrated atrial premature beat	a	123	0.137%
Isolated QRS-like artifact	—	112	0.124%
Ventricular escape beat	E	85	0.094%
Nodal (junctional) premature beat	J	68	0.076%
Unclassifiable beat	Q	29	0.032%
Atrial escape beat	e	13	0.014%
Start of ventricular flutter/fibrillation	[	5	0.006%
End of ventricular flutter/fibrillation	]	5	0.006%
Premature or ectopic supraventricular beat	S	2	0.002%

B Accuracy evaluation with different numbers of hidden neurons

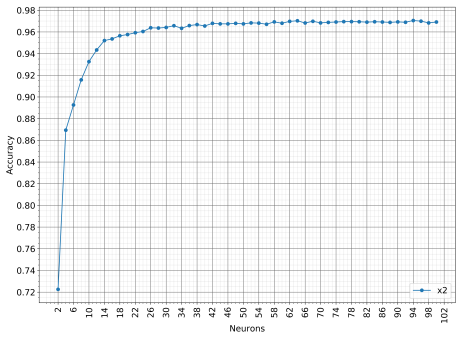

Figure 6: Accuracy with different hidden neurons (# input: 16, # output: 16)

C First truncation method with 2PC using Arithmetic Circuits

We discuss how we implement the first version of the truncation model. The precision of the inputs of the arithmetic circuits is in the order of the 6^{th} floating point: thus weight matrix's and input vector's values are multiplied by 2^{24} and the biases by 2^{48}. These are further converted to integers without having an impact on the precision of the value. Once (1) is executed using arithmetic gates, the intermediate shares of (Y'_h) are amplified by a factor of 2^{48}. Thanks to the initial truncation, this equation does not incur any overflow. However, moving to the square function, the values need to be truncated once more. To make these truncations efficient, we propose to use a simple shift operation. Hence, shares of Y'_h are first transformed to Boolean shares and their bits are shifted by 24: This is equivalent to dividing the values by 2^{24}. To implement the shift function, the wires of position 2,3,...,40 (wire 1 represent the most significant bit (MSB) and wire 64 represent the least significant bit (LSB)) are moved to the position of the 40 most right wires (we do not move the first wire since it represent the sign bit). Then the wires of position 2,3,...,24 are set to the same value of the wire of position 1. This ensures correct binary representation of the values and it is compatible with negative numbers since it respects the 2's complement representation. Note that this method is also implemented in SIMD form as we perform the shift of the bits of all the values in the vector with a single operation. Once the truncation is applied, the Boolean share is re-converted to an arithmetic share and the arithmetic circuit corresponding to the activation function can be applied. Because of the multiplication operation, the

resulting shares of Y_h will again be amplified by a factor of 2^{48}. The same truncation method will thus be repeated. A truncation is not needed at the fourth stage as the argmax operation only outputs the index and not the actual value. Finally, we convert Y' again to a Boolean SIMD share since comparison operations cannot be efficiently computed with arithmetic gates and implement the argmax function the same way it was implemented before.

This first truncation method is evaluated on the test data and it showed good accuracy. The accuracy of the new model with the first truncation method is 96.34% which is similar to what we get from the model with the second truncation method. Also, the confusion matrix of the results is the same as the one corresponding the original model. The evaluation of the model in terms of time and bandwidth consumption was shown in table 2 and discussed in section 3.2

D Evaluation of models using the localhost

Table 5: Performance results for each model (Local evaluation)

	Boolean Circuits	**Truncation v1**		**Truncation v2**			
Proposed NN models	**Model 1**	**Model 1**	**Model 2**	**Model 1 without ARGMAX**	**Model 1**	**Model 2**	**Model 3**
Circuit							
Gates	553925	35477	36418	128	34329	34696	34660
Depth	4513	160	168	5	146	147	146
Time (ms)							
Total	2307.122	44.148	92.071	25.24	39.785	79.934	116.689
Init	0.031	0.027	0.033	0.035	0.032	0.034	0.031
CircuitGen	0.032	0.028	0.031	0.043	0.033	0.031	0.032
Network	0.253	3.458	10.799	10.98	9.78	10.903	8.668
BaseOTs	180.536	180.885	182.411	184.171	183.823	188.162	181.041
Setup	2053.517	32.513	72.795	22.594	29.63	63.531	94.145
OTExtension	2010.62	30.547	70.059	22.462	28.488	62.172	92.311
Garbling	28.847	1.71	2.185	0.001	0.945	0.948	1.23
Online	253.604	11.634	19.274	2.645	10.154	16.402	22.543
Data Transfer (Sent/Rcv, in KB)							
Total	309573 / 319269	2252 / 2629	6651 / 7113	1900 / 1910	2095 / 2171	6461 / 6560	12139 / 12266
BaseOTs	48 / 48	48 / 48	48 / 48	48 / 48	48 / 48	48 / 48	48 / 48
Setup	304815 / 305925	2227 / 2240	6579 / 6591	1881 / 1881	2071 / 2086	6391 / 6406	12010 / 12025
OTExtension	304815 / 301095	2227 / 2057	6579 / 6377	1881 / 1881	2071 / 2053	6391 / 6373	12010 / 11992
Garbling	0 / 4829	0 / 183	0 / 214	0 / 0	0 / 33	0 / 33	0 / 33
Online	4757 / 13344	25 / 389	72 / 522	19 / 29	24 / 85	70 / 154	129 / 240

Table 6: Performance results for the multi-signal model (Local evaluation)

	# Input signals				
	1	**10**	**100**	**200**	**400**
Circuit					
Gates	33741	39552	40918	42422	45426
Depth	148	148	148	148	148
Time (ms)					
Total	37.062	156.838	1124.62	2336.923	4205.39
Init	0.024	0.028	0.025	0.026	0.023
CircuitGen	0.048	0.04	0.054	0.051	0.053
Network	0.323	0.306	0.329	0.327	0.35
BaseOTs	169.302	171.583	173.831	175.802	175.938
Setup	27.101	142.906	1077.62	2245.578	3999.312
OTExtension	26.169	139.448	1049.156	2190.003	3897.242
Garbling	0.747	2.47	19.689	38.629	69.013
Online	9.96	13.931	46.998	91.344	206.077
Bandwidth (Rcv/Sent in KB)					
Total	2095 / 2167	21010 / 21652	209912 / 216247	419805 / 432465	839588 / 864898
BaseOTs	48 / 48	48 / 48	48 / 48	48 / 48	48 / 48
Setup	2071 / 2084	20782 / 20927	207647 / 209107	415276 / 418186	830532 / 836342
OTExtension	2071 / 2053	20782 / 20612	207647 / 205947	415276 / 411876	830532 / 823732
Garbling	0 / 31	0 / 315	0 / 3150	0 / 6300	0 / 12600
Online	24 / 83	227 / 725	2264 / 7139	4528 / 14278	9055 / 28555