PAC: Privacy-preserving Arrhythmia Classification with neural networks

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

EURECOM, Sophia Antipolis, France
{Mohamad.Mansouri, Beyza.Bozdemir, Melek.Onen, Orhan.Ermis}@eurecom.fr

Abstract. In this paper, we propose to study privacy concerns raised by the analysis of Electro CardioGram (ECG) data for arrhythmia classification. We propose a solution named PAC that combines the use of Neural Networks (NN) with secure two-party computation in order to enable an efficient NN prediction of arrhythmia without discovering the actual ECG data. To achieve a good trade-off between privacy, accuracy, and efficiency, we first build a dedicated NN model which consists of two fully connected layers and one activation layer as a square function. The solution is implemented with the ABY framework. PAC also supports classifications in batches. Experimental results show an accuracy of 96.34% which outperforms existing solutions.

Keywords: privacy · neural networks · arrhythmia classification.

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 data, easily. This powerful technology helps make better decisions and accurate predictions in many domains including finance, healthcare, etc. In particular, Neural Networks (NN) can support pharmacists and doctors to analyse patients' data and quickly diagnose a particular disease such as heart arrhythmia. 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[2] 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. Therefore, there is an urgent need for tools enabling the protection of such 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].

[1] https://www.apple.com/lae/apple-watch-series-4/health/
[2] https://www.ibm.com/security/data-breach
[3] https://eur-lex.europa.eu/eli/reg/2016/679/oj

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 (data processors) perform classification without discovering the input (the ECG data). 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 (data subjects or data controllers). 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 service providers and the users and combine the use of neural networks with secure two-party computation (2PC). Since 2PC 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.

With this aim, we reduce the input size of neural network by employing Principal Component Analysis (PCA) [6]. 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 [11] 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 one heartbeat in 1 second, approximately. 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.

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 together with experimental results. 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

As defined in [22], 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 an-

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

alyzing 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). ECG signals representing patients' heartbeats can be considered as sensitive information. Thus, we aim at finding a solution where a service provider 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 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 sending queries.

Performing some operations over data while these are kept confidential requires the use of advanced cryptographic tools such as homomorphic encryption [13, 14, 31, 5] or secure multi-party computation [23, 25]. 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. 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. Hence, the size of the neural network has to be optimized when designing privacy-preserving variants. 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.

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 a publicly available database, namely the PhysioBank database, to build a concrete model as an example and evaluate the accuracy and performance of the newly developed model.

3.1 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 (2PC) which enables two parties to jointly compute a function over their private inputs without revealing any extra information about these inputs than what is leaked by the result of the computation. This setting is well motivated and captures many different applications to ensure privacy protection[5]. We propose to use ABY [11], a mixed-protocol framework that efficiently combines the use of Arithmetic shares, Boolean shares, and Yao's garbled circuits.

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. This architecture seems sufficient to achieve a good accuracy level (see next section). The number of intermediate neurons can be optimized based on several simulations evaluating the accuracy of a model for each case. Additionally, to reduce the number of input neurons, we propose to apply Principal Component Analysis (PCA) [6] and filter out the most significant inputs. Furthermore, because of the complexity of the activation functions, we propose to use the square function, only. This operation 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.

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 (i) the **secrecy of the input** supplied by the client which means that the client would like to get the prediction results without leaking any information about the heartbeat signal; (ii) the **secrecy of the model parameters** supplied by the server while assuming that the client knows the architecture of the model, only; (iii) the **secrecy of the prediction results** with respect to the server.

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

3.2 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 [23] whereby the client holds the input vector, and the server has the NN prediction model which consists of the weight matrices and bias vectors. Similarly to [12] and [7], 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. The resulting neural network model is presented with an incremental approach.

Table 1. Heartbeats for Arrhythmia classification and frequency in datasets

		Our Dataset		PhysioBank Dataset	
Arrhythmia Class	**Symbol**	#	%	#	%
Normal beat	N	14985	34.02%	59926	66.593%
Left bundle branch block beat	L	6450	14.64%	6450	7.168%
Right bundle branch block beat	R	5794	13.15%	5794	6.439%
Premature ventricular contraction	V	5712	12.97%	5712	6.347%
Paced beat	/	5608	12.73%	5608	6.232%
Atrial premature beat	A	2042	4.64%	2042	2.269%
Rhythm change	+	1005	2.28%	1005	1.117%
Fusion of paced and normal beat	f	786	1.78%	786	0.873%
Fusion of ventricular and normal beat	F	647	1.47%	647	0.719%
Ventricular flutter wave	!	378	0.86%	378	0.42%
Nodal (junctional) escape beat	j	184	0.42%	184	0.204%
Non-conducted P-wave (blocked APB)	x	155	0.35%	155	0.172%
Aberrated atrial premature beat	a	123	0.28%	123	0.137%
Ventricular escape beat	E	85	0.19%	85	0.094%
Nodal (junctional) premature beat	J	68	0.15%	68	0.076%
Atrial escape beat	e	26	0.06%	13	0.014%
Signal quality change	V	NA	NA	508	0.565%
Comment annotation	”	NA	NA	352	0.391%
Isolated QRS-like artifact	—	NA	NA	112	0.124%
Unclassifiable beat	Q	NA	NA	29	0.032%
Start of ventricular flutter/fibrillation	[	NA	NA	5	0.006%
End of ventricular flutter/fibrillation	]	NA	NA	5	0.006%
Premature or ectopic supraventricular beat	S	NA	NA	2	0.002%

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. 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 1 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 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. We have removed some heartbeats from the dataset since the classes of V, ", —, Q do not represent any arrhythmia class, and also [,], S are rare. This dataset is further split such that 80% of the heartbeats are used to train the network and 20% of the heartbeats are used as a test dataset. We propose a model with 2 fully connected (FC) layers, one activation function, and a final softmax function that would provide the resulting arrhythmia class.

We now aim at optimizing the number of neurons in each 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.

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 38 is a good choice as this implies less complexity in the model as well as its corresponding confusion matrix shows better fairness toward less frequent classes (see [26] for more details). The accuracy of our model is 96.51%. We represent the model's performance on the test dataset with the confusion matrix as illustrated in Figure 1.

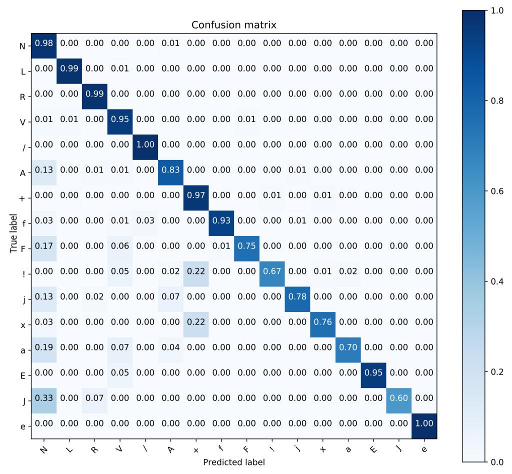

Fig. 1. Confusion matrix of the model for each class in the test dataset

The other 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) [20]. 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 [6]. 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 the square operation as for the activation function. 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 (please see [26] for more details).
The resulting model: To summarize, the developed model, compatible with the use of 2PC, consists of 2-FC layer involving matrix multiplication, one activation layer implementing a square function and one softmax function. The architecture of the proposed neural network model is shown in Figure 2.

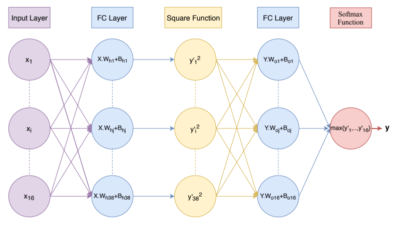

Fig. 2. The proposed NN model

The first layer consists of a fully connected layer and its main operations are: $Y'_h = X^T.W_h + B_h$. In this equation, 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 of Y'_h. The resulting vector Y_h is the final output of the hidden layer. This vector further becomes the input for another FC layer which is defined as: $Y' = Y_h^T.W_o + B_o$. W_o and B_o denote the weight matrix and the bias vector, 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.

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 Principle component analysis (PCA) method. 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 (executed by the server) consists of computing the mean μ of all the elements x_i. Then, the covariance matrix A of S is computed. The eigenvectors and corresponding eigenvalues of matrix A are further evaluated and the first k eigenvectors with the largest eigenvalues are selected. In this particular case, where the dataset consists of 44048 heartbeats of 180 samples, the server obtains he 180×16 matrix of the most relevant eigenvectors. This matrix along with the vector μ is sent to the client who reduces the dimension of its input to 16 by first by subtracting his signal with the mean vector and further multiplying the result with the received matrix.

The use of the PCA transformation at the client side can result in some information leakage. The leakage resulting from the use of PCA mainly are the mean of the dataset and the 180×16 covariance matrix. We argue that 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 eigenvectors. Additionally, 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.

On the other hand, we can also choose not to leak such information while designing the privacy-preserving NN classification. In this case, we either do not use PCA (which causes high bandwidth and computational cost) or include the PCA steps to the 2PC solution (additional overhead but less costly at the first FC layer). Accordingly, in this work, we propose the following three design approaches for PAC (as shown in Figure 3) 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).

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

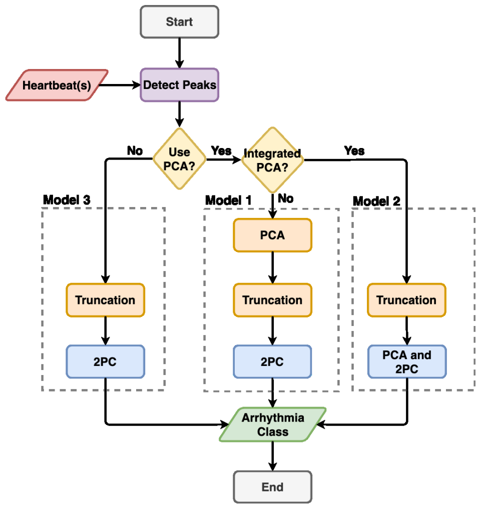

Fig. 3. PAC Overview

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. 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').

Moreover, we propose a secure computation of PCA in Model 2. As previously described, the computation of the PCA can eventually introduce a limited leakage of the training dataset. Therefore, in Model 2, we propose to introduce the computation of the PCA vector into the 2PC. The two parties will first collaboratively compute PCA of the signal (using the confidential mean and the 16 eigenvectors) and further perform classification similarly to Model 1. This PCA computation layer adds 181 SIMD addition gates and 1 SIMD multiplication gate.

3.3 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[6] also supports floating point representation if Boolean circuits are used. Hence, we first implement the privacy-preserving model using Boolean circuits. Nevertheless, floating point representation and the use of Boolean circuits appear to be inefficient. We therefore propose some improvements using fixed point representations that may result in some truncations of the inputs or intermediate values in the circuit. Additionally, since operations executed over Boolean shares are much more expensive than those executed over arithmetic shares, we propose to replace Boolean gates with arithmetic gates as much as possible, hence improve the system. Due to the space limitation, we have not included the model using Boolean circuits and refer to [26] for more details.

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 continuously 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 the full version of the paper. In this truncation approach, only the inputs to the circuits are truncated and this before the actual execution of the circuit. 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.

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. 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 (see Figure 1).

To evaluate the computational and communication overhead of the model in a real setting, 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 simulation of the bandwidth and the latency of the connection between the client and the server showed the values of 39Mbit/sec

[6] https://github.com/encryptogroup/ABY

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 the 3 models implementing arithmetic circuits, mainly. Moreover, we have repeated all evaluations on a local set-up, i.e., the localhost on one machine, to give an insight about the overhead incurred by the low bandwidth (please see the full version for details). The prediction time of one heartbeat is measured as the total time adding to the BaseOT time.

We have also evaluated the performance of the prediction model without using any privacy enhancing technologies and making use of Tensorflow[7]. It takes 7.29ms to predict one heartbeat in cleartext while this value becomes 117,859ms with PAC (without any truncation). Nevertheless, from Table 2, we observe some significant performance gain in terms of computational and communication cost by employing the truncation method. Compared to the model built with Boolean circuits, the Total time with the second truncation method is reduced by a factor of 108.

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[8]	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[8]	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[8]	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

Model 2 still provides better results than Model 3 of which is built without the use of the PCA method: The time and bandwidth consumption of Model 3 is larger with a factor of 1.8 than Model 2. We have also implemented Model 2 without the argmax layer (the output is a vector of 16-value where its argmax

[7] https://www.tensorflow.org/

[8] The Total time corresponds to the Setup time and the Online time. The Network time represents the time for the connection to be established. The Setup time corresponds to the OTExtension time and the Garbling time.

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 the full paper 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.

We also realize that, the online time becomes relatively short when the proposed design is used 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 $S(N)$ (see the full paper [26]). This matrix is further flattened into a vector of $16.N$ elements. Similarly, the server transforms the weight matrix vectors W_h and W_o and obtains vectors of $16.38.N$ elements. The server also creates the bias vectors $B_h(N)$ and $B_o(N)$ of size $38.N$ and $16.N$, respectively (see their representations in the original paper) by expanding the two original bias vectors B_h and B_o, respectively.

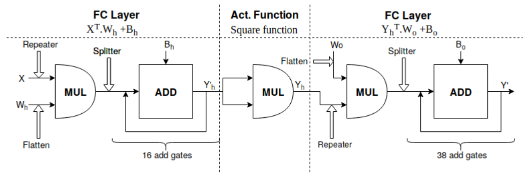

Fig. 4. Arithmetic circuit representation of the model with Truncation v2

The Arithmetic circuit of the NN model is implemented as described in Figure 4 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.

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 separate SIMD vectors. The same comparison and multiplexers gates described before are used but this time the inputs are SIMD shares. The output of the Boolean circuit is the vector $y(N)$ whereby each value represents the index of the class of the corresponding individual.

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

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 (see Table 6 in the full paper [26] for the local setup). We can observe that the number of gates y slightly increases with respect to the number of heartbeats, which is much better than performing prediction on signals individually which will cost $y = 34329N$ gates. Also, the depth is constant regardless of the number of input heartbeats the model predicts.

We observe that the Total time 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 bandwidth grows 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 [12, 7, 15, 18, 4, 34, 16, 19, 9], secure multi-party computation [27, 24, 28, 33, 32, 10, 1, 8, 36], both techniques [3, 30, 21], or, trusted hardware [29, 17, 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) and their models are more complex and not easily comparable to our solutions.

The closest study to our work is [2] which specifically focuses on privacy-preserving arrhythmia classification with neural networks. Authors in [2] use 2PC combined with a partially homomorphic encryption [31]. 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, while our model seems slightly more complex than the one in [2] (38+16=54 neurons instead of 6+6=12 neurons), it shows better accuracy and performance results: The communication channel used in [2] 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 [2] 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 methodology for designing 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 approximately 1 second in real world scenarios. We show that more savings can be achieved with the use of SIMD for performing predictions in batches.

7 Acknowledgments

This work was partly supported by the PAPAYA project funded by the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement no. 786767.

References

1. Ball, M., Carmer, B., Malkin, T., Rosulek, M., Schimanski, N.: Garbled neural networks are practical. Cryptology ePrint Archive, Report 2019/338 (2019), https://eprint.iacr.org/2019/338
2. Barni, M., Failla, P., Lazzeretti, R., Sadeghi, A.R., Schneider, T.: Privacy-preserving ecg classification with branching programs and neural networks. IEEE (TIFS) (2011)
3. Barni, M., Orlandi, C., Piva, A.: A privacy-preserving protocol for neural-network-based computation. In: MM&Sec. ACM (2006)
4. Bourse, F., Minelli, M., Minihold, M., Paillier, P.: Fast homomorphic evaluation of deep discretized neural networks. In: CRYPTO 2018 (2018)
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS (2012)
6. Castells, F., Laguna, P., Sörnmo, L., Bollmann, A., Roig, J.M.: Principal Component Analysis in ECG Signal Processing. EURASIP Journal on Advances in Signal Processing (2007)
7. Chabanne, H., de Wargny, A., Milgram, J., Morel, C., Prouff, E.: Privacy-Preserving Classification on Deep Neural Networks (2017)
8. Chandran, N., Gupta, D., Rastogi, A., Sharma, R., Tripathi, S.: Ezpc: Programmable, efficient, and scalable secure two-party computation for machine learning. In: IEEE EuroS&P (2019)
9. Chou, E., Beal, J., Levy, D., Yeung, S., Haque, A., Fei-Fei, L.: Faster cryptonets: Leveraging sparsity for real-world encrypted inference. http://arxiv.org/abs/1811.09953 (2018)
10. Dahl, M., Mancuso, J., Dupis, Y., Decoste, B., Giraud, M., Livingstone, I., Patriquin, J., Uhma, G.: Private machine learning in tensorflow using secure computation. http://arxiv.org/abs/1810.08130 (2018)
11. Demmler, D., Schneider, T., Zohner, M.: ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In: NDSS (2015)
12. Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: ICML (2016)

13. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: ACM STOC (2009)
14. Gentry, C.: A Fully Homomorphic Encryption Scheme (2009)
15. Hesamifard, E., Takabi, H., Ghasemi, M.: CryptoDL: Deep Neural Networks over Encrypted Data. http://arxiv.org/abs/1711.05189 (2017)
16. Hesamifard, E., Takabi, H., Ghasemi, M., Wright, R.N.: Privacy-preserving machine learning as a service. PoPETs (2018)
17. Hunt, T., Song, C., Shokri, R., Shmatikov, V., Witchel, E.: Chiron: Privacy-preserving machine learning as a service. http://arxiv.org/abs/1803.05961 (2018)
18. Ibarrondo, A., Önen, M.: FHE-compatible Batch Normalization for Privacy Preserving Deep Learning. In: DPM (2018)
19. Jiang, X., Kim, M., Lauter, K.E., Song, Y.: Secure outsourced matrix computation and application to neural networks. In: ACM CCS (2018)
20. Jolliffe, I.T.: Principal Component Analysis (2002)
21. Juvekar, C., Vaikuntanathan, V., Chandrakasan, A.: GAZELLE: A low latency framework for secure neural network inference. In: (USENIX Security) (2018)
22. Kass, R.E., Clancy, C.E.: Basis and Treatment of Cardiac Arrhythmias (2006)
23. Lindell, Y.: Secure Multiparty Computation for Privacy-Preserving Data Mining (2008)
24. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious Neural Network Predictions via MiniONN Transformations. In: ACM CCS (2017)
25. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - Secure Two-Party Computation System. In: USENIX Security (2004)
26. Mansouri, M., Bozdemir, B., Önen, M., Ermis, O.: Pac: Privacy-preserving arrhythmia classification with neural networks. http://www.eurecom.fr/fr/publication/5998/download/sec-publi-5998.pdf (2018)
27. Mohassel, P., Zhang, Y.: SecureML: A System for Scalable Privacy-Preserving Machine Learning. In: IEEE S & P (2017)
28. Mohassel, P., Rindal, P.: Aby^3: A mixed protocol framework for machine learning. In: ACM CCS (2018)
29. Ohrimenko, O., Schuster, F., Fournet, C., Mehta, A., Nowozin, S., Vaswani, K., Costa, M.: Oblivious multi-party machine learning on trusted processors. In: USENIX Security (2016)
30. Orlandi, C., Piva, A., Barni, M.: Oblivious neural network computing via homomorphic encryption. EURASIP (2007)
31. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: EUROCRYPT (1999)
32. Riazi, M.S., Weinert, C., Tkachenko, O., Songhori, E.M., Schneider, T., Koushanfar, F.: Chameleon: A hybrid secure computation framework for machine learning applications. In: AsiaCCS (2018)
33. Rouhani, B.D., Riazi, M.S., Koushanfar, F.: Deepsecure: scalable provably-secure deep learning. In: DAC (2018)
34. Sanyal, A., Kusner, M.J., Gascón, A., Kanade, V.: TAPAS: tricks to accelerate (encrypted) prediction as a service. http://arxiv.org/abs/1806.03461 (2018)
35. Tramèr, F., Boneh, D.: Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In: ICLR (2019)
36. Wagh, S., Gupta, D., Chandran, N.: Securenn: Efficient and private neural network training. In: PETS (2019)