Received November 1, 2019, accepted December 5, 2019, date of publication December 13, 2019,
date of current version December 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2959220

Secure Architectures Implementing Trusted Coalitions for Blockchained Distributed Learning (TCLearn)

SÉBASTIEN LUGAN[1], (Member, IEEE), PAUL DESBORDES[1], ELIOTT BRION[1], LUIS XAVIER RAMOS TORMO[2], AXEL LEGAY[1], AND BENOÎT MACQ[1], (Fellow, IEEE)
[1]ICTEAM, Université Catholique de Louvain, 1348 Louvain-la-Neuve, Belgium
[2]Department of Electrical Engineering and Computer Science (EECS), Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Corresponding author: Sébastien lugan (sebastien.lugan@uclouvain.be)

This work was supported in part by the Wal-e-Cities, a Research Project Co-Funded by the European Regional Development Fund (ERDF) and Wallonia, Belgium, in part by the Bidmed, a Project Funded by Wallonia, Belgium, under Grant RW-DGO6-Biowin-Bidmed, and in part by the Prother-Wal Project funded by Wallonia, Belgium.

ABSTRACT Distributed learning across coalitions is becoming popular for multi-centric implementation of deep learning models. However, the level of trust between the members of a coalition can vary and requires different security architectures. Privacy of the training data has been largely described in distributed learning. In this paper, we present a scalable security architecture providing additional features such as validation on the sources quality, confidentiality on the model within a trusted coalition or confidentiality among untrusted partners inside the coalition. More specifically, we propose solutions that guarantee preservation not only of data privacy but also of data quality, enforce a trustworthy sequence of iterative learning, and that lead to equitable sharing of the learned model among the coalition's members. We give an example of its deployment in the case of the distributed optimization of a deep learning convolutional neural network trained on medical images.

INDEX TERMS Distributed learning, blockchain, convolutional neural network, federated byzantine agreement.

I. INTRODUCTION

Deep learning algorithms, such as the deep Convolutional Neural Network (CNN) [1], constitute outstanding predictive models that help to extract relevant information from large datasets potentially sensitive. In medicine, practitioners routinely use CNN to identify various pathologies [2], contour organs at risk [3], [4], or optimize treatment plans [5]. However, the quality and size of the datasets used for the training phase have a major impact on performances. Still it is often difficult for a single organization such as a single health center to gather enough data on their own and multi-center studies are often hampered to check legal or ethical issues [6].

As a result, distributed learning [7] has been suggested for multiple applications, including in the medical field [8], [9]. This approach facilitates cooperation through coalitions in which each member retains control and responsibility over its own data, including accountability for privacy and consent of the data owners, such as patients. Batches of data are processed iteratively to feed a shared model locally. Parameters generated at each step are then sent to the other organizations to be validated as an acceptable global iteration for adjusting the model parameters. Thus, partners of the coalition will optimize a shared model by dividing the learning set into batches corresponding to blocks of data provided by the coalition members.

The naive use of a CNN in a distributed environment exposes it to a risk of corruption, whether intentional or not, during the training phase. This is because of the lack of monitoring of the training increments and difficulty of checking the quality of the training datasets. One solution could be to have the distributed learning monitored by a centralized certification authority that would oversee the validation of each iteration of the learning process. Alternatively, a blockchain could be used to store auditable records of each transaction on an immutable decentralized ledger. This approach has been

The associate editor coordinating the review of this manuscript and approving it for publication was Diego Oliva.

suggested in [10], where it is advocated that blockchains can be used to store signatures of patient records. In our context, blockchains would provide a more robust and equitable distributed learning process for the stakeholders involved in the learning process since all of them would also be involved in the certification process of each iteration of the model.

Weng *et al.* [11] proposed DeepChain as an algorithm based on blockchain for privacy-preserving deep learning training. It allows massive local training and secure aggregation of intermediate gradients among distrustful owners. Two types of user interact in DeepChain, namely, parties and workers. After signing the trading contract, parties will train the model on their own data to generate intermediate gradients. Those gradients are considered transactions and are collected according to the trading contract. Before being merged with the previous version of the weights, the collected gradients will be validated by the workers according to the processing contract. Parties are rewarded for their contributions to the training process when they provide increments, while workers are rewarded according to their contribution to the validation of these increments. Because DeepChain maintains the payoff maximization of parties and workers, a healthy, win-win environment is created. At the end of the process, Deepchain provides the auditability and confidentiality to train gradients for each participant locally while employing economic incentives to promote honest behavior. Nevertheless, it does not prevent data exposure through a malicious partner. Thus this model does not protect the shared model against degradation or divulgation.

The contribution of this paper is to derive a new model of coalitions with a high degree of reliability that respects data privacy and incentives participation in the coalition without a central authority. In order to implement this model, we propose novel scalable security architectures, called Trusted Coalitions for distributed Learning (TCLearn), based on either public - to be open to a large amount of participants - or permissioned blockchains to provide distributed deep learning with increasing levels of security and privacy preservation.

In our approach, a CNN model is shared among the members of the coalition and optimized in an iterative sequence, with each member of the coalition updating it sequentially with new batches of local data. Each iteration of the shared model is validated by a process involving the members of the coalition and then stored in the blockchain. Each step of the evolution of the model can be retrieved from the immutable ledger provided by the certified blockchain.

While early implementations of blockchains, such as Bitcoin, initially relied on a proof-of-work [12] consensus mechanism, other architectures such as proof-of-stake [13] have been suggested since. Our approach has a different goal: to provide an iterative certification process for each learning step of the shared model, all of them being registered in the underlying ledger. We therefore suggest a new consensus mechanism based on a custom Federated Byzantine Agreement (FBA) integrating performance evaluation into the block validation step.

In this article, the model designates a (deep-)CNN with its associated weights (parameters). The gradients represent the evolution of the model weights after a training step. The supervisor is an entity handling the storage of the model and controlling its access.

II. THREATS AND SECURITY GOALS

In this section, we discuss threat scenarios where the data or model is attacked by disruptive parties. In comparison with DeepChain, we do not have trust issues with the encryption process because it is done by a trusted entity: the supervisor. This role is strictly restricted to ensure secure channels between the members of the coalition. For this reason we focus on threats applied to the data and the model.

A. THREAT 1: KEEP CONTROL OVER THE DATA

The leakage of data is an important threat to address, specially in the case of medical data. New regulations (e.g., the GDPR in EU) are making access to personal data more restrictive in order to respect their owners' privacy and consent of use.

1) SECURITY GOAL 1: PRIVACY OF THE TRAINING DATASET

This threat refers to two different elements: a) disclosure of the private data brought by each partner to perform a training step and b) the possibility of reconstructing part of the training set from the generated gradients, which is known as the long-term memory effect [14].

The use of distributed learning protects the training data. It makes it possible to improve a shared model with new data without their leaving the local environment.

To avoid any leaks from the gradients, differential privacy has been proposed in the literature. A computation is considered to be differentially private if the probability of producing a given output does not depend very much on whether a particular data point is included in the input dataset [15]. For any of two datasets *D* and *D'* differing by a single item, also called adjacent databases, and any output *O* of function *f*,

$$Pr\{f(D) \in O\} \leq \exp^{\epsilon} Pr\{f(D') \in O\} \tag{1}$$

The parameter ϵ controls the trade-off between the accuracy of the differential privacy f and how much information it leaks.

In our use case, as evoked by Shokri *et al.* [16], participants may reveal some information about the training datasets indirectly via public updates to a fraction of the CNN parameters during training.

Several approaches have been proposed in the literature to mitigate this effect. As an example, the use of Homomorphic Encryption (HE) techniques has been considered in [17]. In their article Shokri*et al.* [16] suggest sharing a small proportion of their gradients randomly and perturbing them by adding some noise. This approach reduces the performances of the model while improving privacy.

B. THREAT 2: KEEP CONTROL OVER THE MODEL

This threat refers to two different elements: a) the model is exposed to a risk of corruption or degradation during the training phase. For instance, a partner could attempt to train the model on corrupted data or a different pathology from the one studied. b) The blockchain keeps track of each and every modification to the model, but does not prevent unauthorized use of the shared model outside the coalition. If the learned model has to be confidential, an extra level of protection must be added to prevent any potential deliberate or accidental leak outside the consortium.

1) SECURITY GOAL 2A: PROTECTION OF THE MODEL AGAINST DEGRADATION BY TRAINING ON INADEQUATE DATA

The evolution of the model must be resilient to malicious or clumsy actions that would decrease the performances of the model. The proposed approach must detect this kind of misuse and reject the resulting increment. Traceability of every operation involving the model must be ensured to deter any malicious event.

2) SECURITY GOAL 2B: CONFIDENTIALITY AND TRACEABILITY OF THE MODEL

The disclosure of a trained model defined as confidential by the consortium leads to a leak of intellectual property. This threat depends on the confidentiality level required by the consortium.

III. A SCALABLE SECURITY ARCHITECTURE FOR TRUSTED COALITIONS

The trade-off between security and cost [18] is a well-known issue. In this section, we will develop three different methods corresponding to three distinct trust levels depending on the shared rules in the coalition:

- **Method TCLearn-A:** The learned model is public but each member of the coalition is accountable for the privacy protection of its own data.
- **Method TCLearn-B:** The learned model is private (shared only within the coalition) and the members of the coalition trust each other.
- **Method TCLearn-C:** The members of the coalition do not trust each other and want to prevent any unfair behavior by any of them, such as unauthorized use or leaking of the model outside the coalition.

These three methods address the previously described security issues at different levels (see Table 1), offering an inherent trade-off between security needs and costs.

A. ARCHITECTURE OF TCLEARN-A

Type of coalition: coalition sharing a public model built using private datasets. The integrity of the increments is ensured through the use of a new federated Byzantine agreement protocol.

Solution to Threat 1 (Privacy of the training dataset) A partner willing to improve a model with its new data has to:

TABLE 1. Summary of the features for the three TCLearn methods.

TCLearn	A	B	C
Data privacy	✓	✓	✓
Protection against degradation	✓	✓	✓
Model privacy for outside threats	×	✓	✓
Model privacy for inside threats	×	×	✓
Model access audit trail	×	✓	✓
Model leakage traceability	×	×	✓

1) fetch the current version of the weights W_i and apply it to the known architecture.
2) trains the model locally with its own dataset, generating new gradients G_{i+1}.
3) uploads the resulting gradients G_{i+1}.

This way, the dataset used for training never leaves the partner's infrastructure, ensuring its privacy and excluding any leakage of the processed data. Because access to the previous datasets is forbidden, this training step builds epochs only from the new dataset provided by the user. After this training, the generated gradients G_{i+1} are uploaded and applied to the previous weights (W_i), leading to new parameters (W_{i+1}) and a candidate model, noted $tmp(M_{i+1})$.

To protect against any leaks coming from the gradients, it is possible to share a small proportion of the gradients and add some noise as suggested by Shokri *et al.* [16] in order to improve differential privacy. The parameter ϵ, controlling the trade-off between the accuracy and privacy, should be determined by the user.

Solution to Threat 2a (Protection of the model against degradation by training on inadequate data)
Our approach relies on a blockchain to carry only unalterable cryptographic hashes of the successive training steps of a model built in a distributed environment that ensures the validation of the successive iterations. The iteratively optimized model is made public. Each block represents an iteration step achieved locally by a specific member of the coalition and validated by the whole coalition. First, the model and the genesis block are initialized, setting the architecture (layers, activation functions, loss function, etc.) and the weights according to a normal distribution. The weights of the model are updated iteratively by the batches of data provided by the members of the coalition.

In our approach, we use a blockchain relying on a federated Byzantine agreement to prevent corrupted increments caused by inadequate training from being added to the model. The candidate increment has to be validated by multiple validators. Since the blockchain and the deep learning model are strongly linked in TCLearn, the FBA has two goals:

- control the quality of the updated CNN model, through a "peer review" system and
- control the integrity of the new block (hash, index, timestamp, etc.) and concatenate it to the chain.

The FBA process starts with the random selection of validators within the consortium. All the members can be

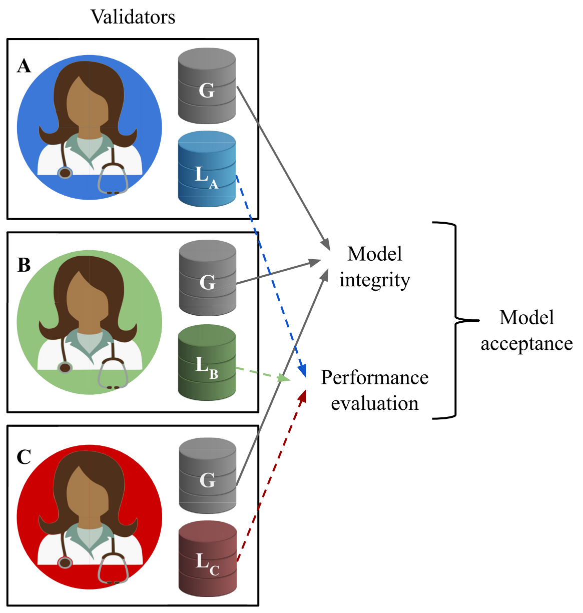


FIGURE 1. Federated Byzantine agreement and candidate model checking process. Two datasets are used: a global one (*G*) similar for all the partners that is used to control the model's integrity and a local one (*L*), different for all the partners that is used for performance evaluation. After a majority vote, the candidate model is accepted or not.

selected but the probability depends on the size of the consortium and the proportion of data brought by each member as follows (Equation 2),

$$S_i = 1/N + D_i/D \tag{2}$$

where S_i is the strength of the partner i, N the total number of partners in the consortium, D the total amount of samples used for the model and D_i the amount of samples supplied by the partner i. Initially, this strength is the same for each of the partners and evolves with each contribution. The main role of the validators is to check the candidate model M_{i+1} incremented from M_i proposed by a partner. The model M_{i+1} must show an improvement over the previous one. The test on the validity of the increment should protect the model against corrupted data in the training set.

Two types of test databases are used to assess the performances of each increment of the model (Fig. 1) and to avoid the introduction of invalid or inadequate training sets:

- A global test database (G), common for every block creation and for all the partners. This database is created by experts to be representative of the pathology.
- A local test database (L), different for each partner in the consortium. To avoid overfitting on the global test dataset (G), a small percentage of the input signals is put aside locally for each contribution. It is used later by each validator as a local test set to evaluate the proposed model individually.

Both datasets are used for the testing phase. First, results obtained using the common, global dataset are compared between validators to ensure that the candidate model is functional and identical. Then, those "global" results are merged with those obtained using the individual, local datasets. To be accepted, the candidate model must have higher performances than the previous model within a specific threshold λ ($\in [0, 1]$) (Equation 3).

$$\text{Block creation IF } \lambda \times perf(M_i) \leq perf(M_{i+1}) \tag{3}$$

Once the model is accepted, a new block can be created. First, the block creation requires that a "general" (or "speaker") be selected from among the validators. The "general" will be the creator of a new block containing the reference to the validated model M_{i+1} and the ID of the partner who proposed it. All the validators then check the integrity of this candidate block.

This block is analyzed in the frame of a delegated Byzantine fault tolerance system also suggested by Damaskinos *et al.* in [19]. Each validator broadcasts its opinion (acceptance or rejection) of this block to the other validators. If at least two-thirds of the validators agree, the FBA process can stop, leading to the acceptance or rejection of the block. If not, the role of "general" is switched to another randomly selected validator and the block creation process can restart. If the block is accepted by the validators, the "general" can append it to the blockchain and broadcasts this update to the whole consortium, requesting synchronization of the blockchains.

The overall scheme of TCLearn-A is represented in Fig. 2.

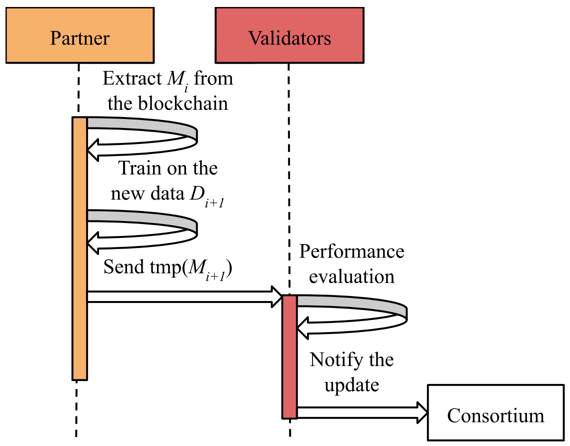


FIGURE 2. Scheme of the TCLearn-A procedure. Partner trains a model (M_i) on its data (D_{i+1}). The candidate model $tmp(M_{i+1})$ is validated by a federated Byzantine agreement.

Each block includes:

- block index
- timestamp
- previous block hash
- hash of the model's parameters
- user ID: identification of the contributor
- users' strength: level of contribution of each member to the FBA
- block hash

The pseudo code for this approach is the following:

Algorithm 1 TCLearn Concept

```
   Input: A blockchain $bc$ and a training set $D_i$
1  modelTrain($bc$, $D_i$):
2      $w_i$ = loadWeights($bc$);
3      $M_i$ = initializeModel($w_i$);
4      $tmp(M_{i+1})$ = training($D_i$, $M_i$);
5      $tmp(bloc)$ = blocCreation($bc$, $tmp(M_{i+1})$, $D_i$);
6  federatedByzantineAgreement($bc$, $tmp(bloc)$):
7      controlPerformances($bc$, $tmp(M_{i+1})$);
8  if $\lambda \times perf(M_i) \leq perf(M_{i+1})$ then
9      if $integrity\ of\ bc == TRUE$ then
10         blockCreation(bc, tmp(bloc));
11     end
12 end
```

Audit & Traceability of the Training Data: The evaluation of performances may not be enough to avoid duplicate input signals during training. For this reason, anonymous IDs can be attached to each input signal and stored in the blockchain. This process also enforces the auditability and the traceability of the input signals. This can be implemented as an additional field stored in each block, such as an anonymized data ID, therefore avoiding training on duplicate input signals.

B. ARCHITECTURE OF TCLEARN-B

Type of coalition: coalition sharing a private model built using private datasets in a restricted consortium of trusted partners. The integrity of the increments is ensured through the use of the FBA protocol. In this situation, the model has to be protected during transfers between partners and for its storage.

Solution to Threat 1 & 2a
As this approach is based on TCLearn-A, the data privacy of the inputs is preserved and the model iterations are validated by members of the coalition.

Solution to Threat 2b (confidentiality and traceability of the model)
With TCLearn-A, the evolutions of the model are certified by the blockchain. However, this scheme does not guarantee the confidentiality of the model during its distribution to partners. This situation is not acceptable in some situations where the privacy of the model must be preserved (for example, if the members of the coalition want to avoid any leakage of the model).

To solve this issue, the storage, transfer, and upgrades through gradient computations of the model have to be protected by encryption, where the private keys are stored by some trusted entities. In this work, we propose to isolate all iterations of the model in an external, off-chain encrypted storage facility (“vault”) and control its access by each partner.

We also suggest the use of secure transport (e.g., TLS or S/MIME) for transferring the model. Moreover, the model could be stored by using an efficient encryption method and implementing access control and auditing mechanisms. Only authorized users should be able to download a given version of the model weights, and each access should be logged in an audit trail.

In order to (1) ensure that only authorized participants are granted access to the models and (2) minimize the risks of leakage, we propose to store the actual models (including the associated weights) in an external and secure “off-chain storage vault”. In this approach, the blockchain provides only “links” to the corresponding version of the model’s weights.

This introduces a single point of failure (the secure, off-chain storage and its associated access control infrastructure), but offers three major advantages. First, it greatly reduces the size of the blockchain while increasing its scalability, so that each participant needs to synchronize fewer data. Second, this approach makes it possible to implement active access control over all of the stored information (e.g., who is allowed to access which kind of data, when and how). Finally, each of the requests to access the secure, off-chain storage vault could be recorded in a journal, offering the ability to audit all accesses to any record. This audit could be used to gather statistics about the actual accesses of each individual user to models. This in turn could be used to restrain future accesses (e.g., allowing users to access a certain number of models depending on their level of contribution to the models). Therefore, a moderated level of traceability is offered: if a given version of a model is leaked, it will be possible to audit all of the requests related to this specific version predating the leak in order to establish the list of partners that actually downloaded this version and potentially leaked it.

The blockchain/offline storage approach requires that an entity (the “supervisor”) manages the access control and the secure storage of the model. An overview of this approach is presented in Fig. 3.

If we look again at the pseudocode (cf. Algorithm 12), we can add a line of encryption between line 5 and 9, and one of decryption between lines 6 and 7.

Secure Authentication and Transport of the Model: The secure authentication and transport of the model could be performed in two ways: either online (requiring direct, interactive communication between the partner and the supervisor) or offline (allowing for delayed communication, e.g., using periodic batches of file transfers or e-mail).

In the former case, we strongly recommend using an industry-standard protocol such as TLS (Transport Layer Security) v.1.3 (RFC 8446), which is used, for instance, in HTTPS. Unlike network-level encryption (such as IPsec VPNs), TLS offers end-to-end encryption that guarantees the confidentiality of the model and gradients, including between the secure gateway (e.g., VPN concentrator) and the machine performing the machine learning. The server (the supervisor) must be authenticated by validating its X.509 certification chain. The identity of the client could also optionally be requested (using a X.509 certificate chain) in order to provide a stronger authentication mechanism. Once the

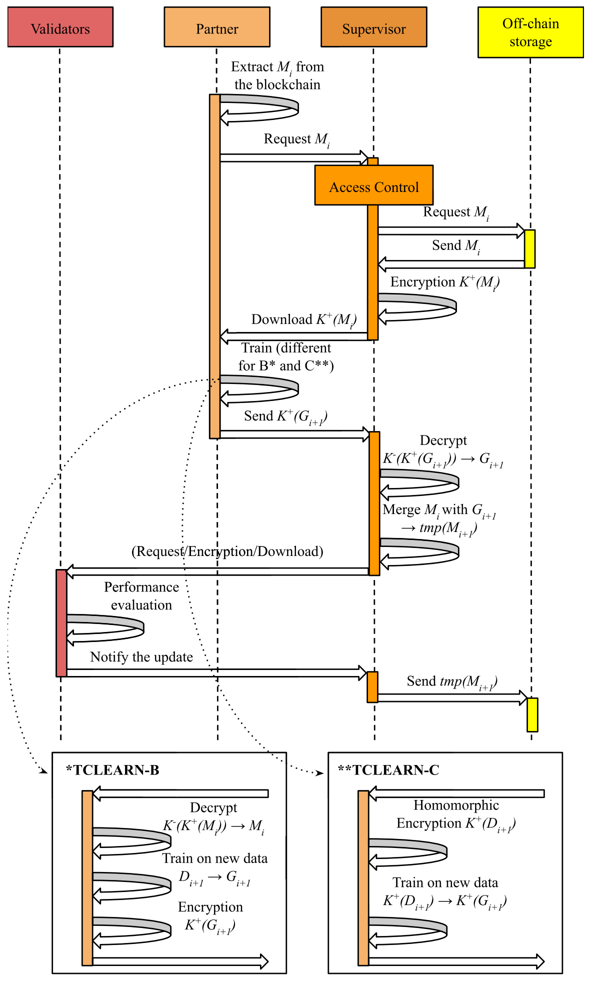

Validators
Partner
Supervisor
Off-chain storage
Extract M_i from the blockchain
Request M_i
Access Control
Request M_i
Send M_i
Encryption $K^+(M_i)$
Download $K^+(M_i)$
Train (different for B* and C**)
Send $K^+(G_{i+1})$
Decrypt $K^-(K^+(G_{i+1})) \rightarrow G_{i+1}$
Merge M_i with $G_{i+1} \rightarrow tmp(M_{i+1})$
(Request/Encryption/Download)
Performance evaluation
Notify the update
Send $tmp(M_{i+1})$
*TCLEARN-B
Decrypt $K^-(K^+(M_i)) \rightarrow M_i$
Train on new data $D_{i+1} \rightarrow G_{i+1}$
Encryption $K^+(G_{i+1})$
**TCLEARN-C
Homomorphic Encryption $K^+(D_{i+1})$
Train on new data $K^+(D_{i+1}) \rightarrow K^+(G_{i+1})$

FIGURE 3. Scheme of the TCLearn-B & C procedure. The partner trains a model (M_i) on their data (D_{i+1}) leading to new gradients (G_{i+1}). The model is successively encrypted and decrypted, e.g., using a public (K^+) and a private key (K^-), respectively. The candidate model ($tmp(M_{i+1})$) is validated by federated Byzantine agreement. The two methods are principally differentiated by the training process in or out of the encryption domain.

TLS handshake and authentication successfully performed, the model and its gradients can be exchanged safely using strong encryption (such as AES-256).

In the latter case, we suggest using an industry-standard protocol such as S/MIME (RFC 5751). Both the supervisor and the partner need to possess X.509 certificates, which could be used to both authenticate (digital signatures of the partner's request and the supervisor's reply) and encrypt the data (using the partner's public key so that only this partner can decrypt it). In this scenario, the partner could send a request, signing it digitally using its X.509 private key; the supervisor ensures the authenticity of the request by verifying the provided digital signature (decrypts it using the partner's public key). Once the partner (and the corresponding request) are authenticated, the supervisor will be able to send the encrypted model and gradients. This operation could be performed by the supervisor, for example, by first generating a random session key that could be used as a symmetric key with a symmetric encryption system such as AES. This random session key is then encrypted using the partner's public key (so that only the partner could decrypt it using its private key) and sent along with the encrypted model and gradients to the partner. The partner then simply needs to decrypt the symmetric session key using its private key and then use this (decrypted) session key to decrypt the message. This scheme (illustrated in Fig. 3) requires each party to possess an X.509 certificate and the associated private key in order to decrypt information provided by other parties.

C. ARCHITECTURE OF TCLEARN-C

Type of coalition: coalition sharing a private model built using private datasets in a restricted consortium of untrusting partners. The integrity of the increments is ensured through the use of an FBA protocol. To protect the model in this situation, it is necessary to secure the exchanges and the storage even at the partners' facilities.

Solution to Threat 1 & 2a

As this approach is based on TCLearn-A, the data privacy of the inputs is preserved and the model iterations are validated by members of the coalition.

Solution to Threat 2b (confidentiality and traceability of the model)

In this scenario, our objective is to identify the partner responsible leaking a model. Such identification could be performed by adding some unique, hidden information to the model provided to each partner, e.g., by altering the weights by introducing a moderate noise following a specific, hidden pattern (constituting a watermark) every time the model is requested by a partner. The date of the access, identity of the partner, and associated hidden pattern (the watermark) could then all be stored in an audit trail allowing to uniquely identify the partner associated with a leaked model. Unfortunately, CNNs are quite robust to a slight alterations of the weights, which means that an adverse party might try to alter those (watermarked) weights, jeopardizing the recovery of the watermark while keeping the model usable. This adverse party could even perform a subsequent training of the model on new datasets, further compromising the watermark's recoverability.

Another option could be to send to each partner a model encrypted with a specific key, allowing them to manipulate (use and even train) it without being able to decrypt it. Some crypto algorithms (such as the ElGamal scheme [20]) allow operations to be performed directly on the encrypted data without knowing the associated private key. For instance, the ElGamal scheme offers an encrypted product operator allowing one to compute the product $E(m_1 \cdot m_2) = HM(E(m_1), E(m_2))$ of two encrypted messages $E(m_1)$ and $E(m_2)$. An operator offering such a property constitutes a homomorphic operator. An encryption algorithm offering both homomorphic product and homomorphic addition, such as the Brakerski-Gentry-Vaikuntanathan scheme [21], constitutes a fully homomorphic encryption scheme.

Homomorphic encryption brings a new level of protection to the model (since actually only the supervisor is able to decrypt it), but also a considerable drawback: in order to perform prediction, the encrypted result must be sent by the partner to the supervisor to be decrypted and the final result sent back to the partner. This reduces the partners' autonomy (they depend on the supervisor to perform any prediction using the model) and potentially introduces a single point of failure.

Similarly to TCLearn-B, the confidentiality and the distribution control over the models is ensured by off-chain storage of the models and gradients. However, since the members of the consortium do not trust each other, no member should have access to unencrypted models or gradients. This is ensured by the use of the aforementioned homomorphic encryption technique. In order to use the model (prediction), the input signals must first be encrypted using the same technique and the same homomorphic public key (which must be sent together with the encrypted models and gradients to each partner). This public key can be used only to encrypt (and not decrypt) data. Once the prediction operation (using encrypted model and input signals) is performed, the result must be decrypted. This must be done by the supervisor (which holds both the homomorphic public and private keys corresponding to the models sent to each partner). The supervisor uses its homomorphic private key to decrypt the result, which is then sent to the requesting partner. The supervisor must inspect the ''results'' to decrypt very carefully in order to reject any attempt to decrypt models or gradients.

Considerations Regarding the Use of Homomorphism in CNNs: In TCLearn-C we suggest the use of homomorphic encryption to ensure traceability and confidentiality of the model. In this scenario, the whole manipulation of the CNN takes place in the homomorphic domain, which introduces some challenges that are described hereafter.

First, the neural network algorithm requires that both addition and multiplication operations be used and combined. Consequently, the ''somewhat homomorphic encryption'' scheme (see ElGamal [20], BGV [21] or Paillier [22]), which allows the use of one sole type of homomorphic arithmetic operation, cannot be used. We thus have to use FHE, which requires considerable memory and computational resources.

The use of a CNN in the homomorphic domain also introduces implementation issues. Indeed, while most FHE implementations support only homomorphic addition and multiplication, implementating CNN activation functions in the homomorphic domain requires complex operations such as trigonometric operations (tanH), exponentials (sigmoid), and tests (ReLU). Zhang *et al.* [23] proposed the use of a Taylor development to replace the sigmoid function in order to compensate for the lack of exponential functions in FHE schemes. Although this method allows us to use the sigmoid function in the homomorphic domain, it remains approximate. Moreover, it still increases the number of operations to perform.

Solving those problems has been the subject of some recent work. As an example, in [24], Bourse et al. presented a framework for homomorphic evaluation of neural networks using a highly optimized FHE algorithm. This scheme, dubbed TFHE, offers several orders of magnitude performance improvements over previous FHE architectures. In this article, the authors used a ''discretized'' neural network to allow the creation of a model capable of fitting data from the MNIST database. Several tips are proposed to reduce the time required for learning and prediction (bootstrapping, look-up table, and noise management).

One could thus use Bourse's solution. If this approach is not sufficient to mitigate the resource issues associated with TCLearn-C, the training process could alternatively be performed using tamper-proof black boxes deployed in each of the partner's facilities. Such black boxes could be used to decrypt the CNN model, retrain it, and then re-encrypt it without anyone being able to interfere. However, this solution requires all of the partners to request the installation and maintenance of this black box by a trusted third-party service provider.

If we look again at the pseudocode (cf. Algorithm 12), we can add a line of encryption between lines 5 and 6.

D. ADDITIONAL FEATURES

In addition to the TCLearn approaches, we propose two optional extra features.

1) REVERTING TO A PREVIOUS STATE OF THE MODEL

Each and every increment of the model recorded on the blockchain and validated using the FBA are accessible by the partners. In the case of late detection of corruption, it could be necessary to restore the model to one of its previous states. For this reason, we add the possibility of performing prediction or even continuing the training from old weights stored in the off-chain site. Only authorized partners are able to perform this operation, in which case a new block is generated, prevailing over previous learning steps.

2) UPDATES TO THE MODEL HYPERPARAMETERS

The model architecture (kernel size, number of layers, etc.) is initialized in the genesis block of the blockchain but some hyperparameters (epoch number, batch size) can be modified during the learning step. The learning rate of the optimizer for the gradient descent is a critical parameter. It could be high at the beginning of the blockchain because the model is still naive, but the more precise the model is, the lower the learning rate should be (without becoming too low). A method that adapts the learning rate according to the performances of the model automatically could be integrated into our concept [25].

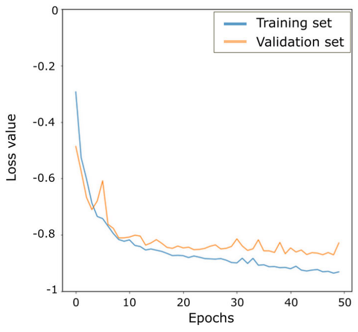
IV. SECURITY ANALYSIS

A. SOLUTION TO THREAT 1: KEEP CONTROL OVER THE DATA

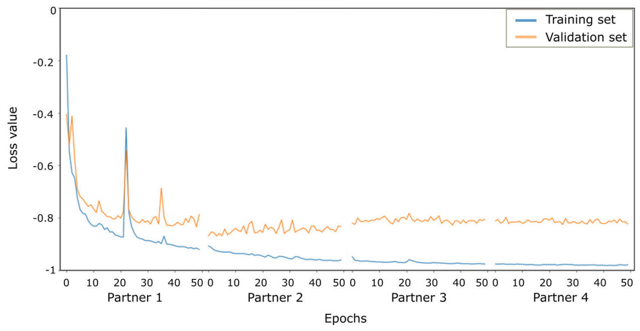
1) DISTRIBUTED LEARNING FOR A MEDICAL APPLICATION: AN EXAMPLE

To illustrate the use of distributed learning, we propose to apply it to a medical challenge that has already been solved using CNNs, namely, bladder contouring on computed tomography scans [3], [4]. Léger et al. proposed using U-Net [26] to segment the bladders of 339 patients with prostate cancer. Their semi-automatic approach takes two channels as input (a 3D volume of the bladder with one of its slices labeled manually by an expert) and outputs a prediction for the target bladder segmentation tile.

We reproduced their results using the parameters and database used by Léger *et al.* [3] and Brion *et al.* [4]. First, we performed a centralized training using all the training samples in 50 epochs (see Fig 4a). Then, we randomly split the initial database into several subsets to simulate several partners and perform a distributed learning using the smaller datasets successively. Thus, each partner realizes their own training process (with 50 epochs) one after the other (see Fig 4b).



(a)



(b)

FIGURE 4. Loss value (- Dice score) during training (a) in a centralized and (b) in a distributed way [3], [4].

Once the training was over, the accuracy achieved with a distributed database was not significantly different from that resulting from centralized training (88.4% and 88.7%, respectively). This result means that the model is able to catch relevant information from the split dataset as well as from the centralized one. This result is the same even if we switch the order of the partners. In both cases each partner has a balanced dataset. Otherwise, this could lead to performance impairment of the model; TCLEARN would therefore reject the proposed model. Moreover, to avoid the long term memory effect, the training step included batch inputs. Thus, each gradient represents the average information provided by these batches and not just by a single patient. Therefore, we cannot directly infer the features of a given patient from the gradients.

B. SOLUTION TO THREAT 2: KEEP CONTROL OVER THE MODEL

1) SOLUTION TO THREAT 2A: PROTECTION OF THE MODEL AGAINST DEGRADATION BY TRAINING ON INADEQUATE DATA

To protect the model against malicious events that might reduce its accuracy we proposed using of federated Byzantine agreement. This system ensures the integrity and performances achieved by each proposed candidate model.

2) SOLUTION TO THREAT 2B: CONFIDENTIALITY AND TRACEABILITY OF THE MODEL

This threat depends on the confidentiality level required by the consortium. In TCLearn-A, the model is open and does not require any protection. In TCLearn-B, the model is built using private datasets and shared within a restricted consortium of trusted partners. In this case, we propose using secure transport for exchanging the model between the partners and the supervisor. For TCLearn-C, the model is built using private datasets and shared within a restricted consortium of untrusting partners. In this case, we suggest the use of homomorphic encryption allowing to use the model in the encryption domain.

In both cases, the encryption is managed by a trusted supervisor (either at the transport level such as in TCLearn-B or at the application level such as in TCLearn-C). Moreover, we suggest storing the model in an external off-chain storage vault. The blockchain will contain links to the model, thereby minimizing its size while allowing the implementation of access control and auditing mechanisms. Only authorized users should be able to download a given version of the model weights, and each access should be logged into an immutable audit trail.

It might be possible to use this audit to gather statistics about the actual accesses of each individual user to models. This in turn could be used to restrain future accesses (e.g., allowing users to access a certain number of models depending on their level of contribution to the models). Therefore, a moderated level of traceability is offered even in the case of TCLearn-B: if a given version of a model is leaked, it is possible to audit all of the requests related to this specific version predating the leak in order to establish the list of partners that actually downloaded this version and may have leaked it. In the case of TCLearn-C, the public homomorphic key associated with the model could be directly matched to a single access (member, date and time, and accessed model).

V. IMPLEMENTATION AND EVALUATION

In this section, we present an implementation prototype for TCLearn-A & B, that demonstrates the feasibility of our approach. This prototype is available at https://github.com/slugan/TCLearn .

The goal of this prototype (which is not intended for deployment in a production environment) is to illustrate and demonstrate an example of implementation of the presented architectures.

The blockchain architecture is based on the proof-of-concept implementation proposed by Gerald Nash in this article: https://medium.com/crypto-currently/lets-build-the-tiniest-blockchain-e70965a248b.

Our prototype relies on Python version 3.6.8 and Numpy version 1.16.4. The server-side scripts (including the supervisor) are implemented using Flask Microframework version 1.1.1 while the client-side scripts rely on requests version 2.22.0 (a Python HTTP library). The sample certificates are generated using OpenSSL version 1.0.2k. The (purely illustrative) server-side TLS encryption is handled by the uWSGI application container server version 2.0.17.1. The deep learning part is based on Tensorflow version 1.5.0 and Keras version 2.2.4.

In this prototype, each virtual site is associated with a Flask application. A separate application simulates the supervisor (in the case of TCLearn-B and TCLearn-C). Python scripts are used to send commands to these applications using HTTPS calls (e.g., to simulate submissions by a given partner). Please consult the documentation on the GitHub repository for more information.

This proof-of-concept illustrates the principles of the TCLearn architectures on the MNIST dataset (32×32 pixels handwritten digits, 60000 training images and 10000 test images). The test set is split in two groups: the first half is used as a global test set and the second half, divided equally between 7 partners, constitutes the virtual consortium. Each partner is able to test a candidate model using both the global test set and their own local test set. The training set is split into batches of increasing size, each associated with a distinct partner of the consortium and constituting the input used by this partner to submit a new candidate model (iteration). The CNN architecture used is: $Input \rightarrow Conv \rightarrow Maxpool \rightarrow Fully\ Connected \rightarrow Output$. The optimizer is Adadelta with a learning rate of 1.0, a mini batch size of 64 and number of epochs equal to 20.

In the first example (Fig. 5), the training set is randomly and arbitrarily split in (small) batches of 204, 1080, 1080, 3000, 4800, 4800 and 9600 images (40% of the complete set). The accuracy slowly increases (due to the small size of the initial batches) quite steadily among the various test sets with each iteration (corresponding to the submission of a new candidate model by a partner). The global test database (7 times larger than each of the local test set of each partner) is associated to the highest accuracy values as expected.

In the second example (Fig. 6), the training set is now randomly split in batches of 2400, 4800, 4800, 6000, 10800,

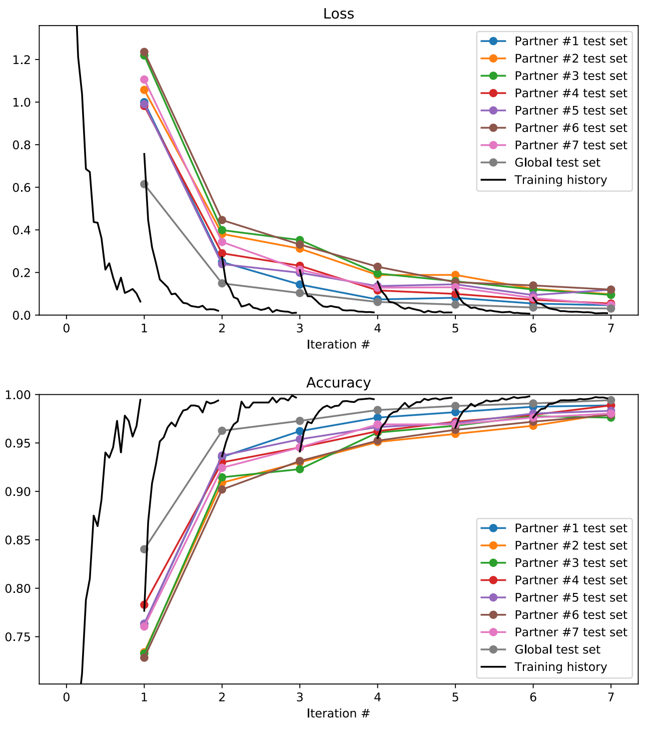


FIGURE 5. Loss and accuracy with small batches (204, 1080, 1080, 3000, 4800, 4800 and 9600 images). Training history represents the performances obtained during the training process on the training dataset.

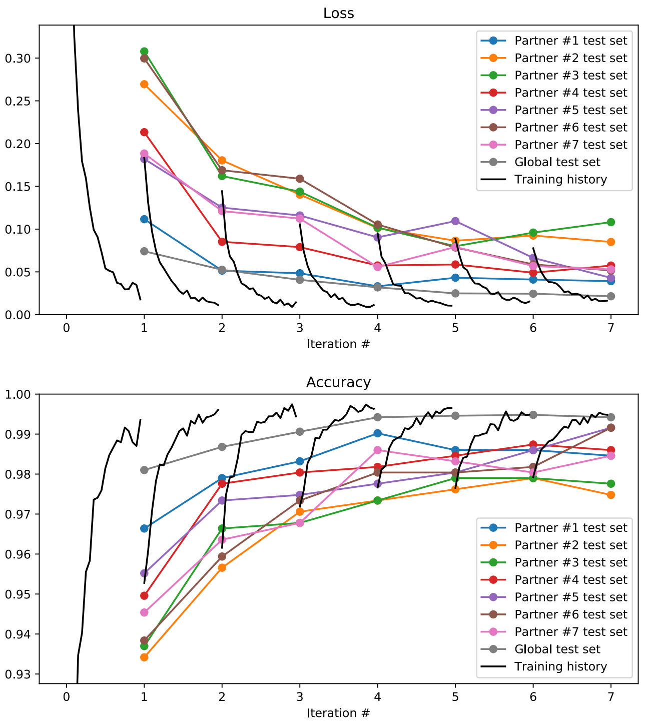


FIGURE 6. Loss and accuracy with larger batches (2400, 4800, 4800, 6000, 10800, 10800 and 20400 images). Training history represents the performances obtained during the training process on the training dataset.

10800 and 20400 images (100% of the complete set). Given the larger size of the initial batches, the accuracy reaches a high level from the firsts iterations, and globally continues to increase. However, we notice a few occurrences of decreasing accuracy from one iteration to the next one when tested against some local test sets. This illustrates the need for the joint use of the global test set by each partner in addition to their own local test set when evaluating a candidate model. This also shows the importance of the λ threshold tolerating a moderate decrease of the performances when evaluating a new submitted model.

VI. CONCLUSION

In this article, we propose a new architecture for distributed learning by a model based on federated Byzantine agreement mechanism. The performance of the model is ensured through a shared evaluation of individual contributions leading to acceptance or rejection based on an objective criterion.

This approach makes it possible to constitute trusted coalitions in which the actions for updating the model by the members are registered in a public ledger implemented as a blockchain. We have explored three kinds of coalitions according to the access control required for distributing the model. Each approach corresponds to distinct trust levels that depend on the shared rules in the coalition. We have proposed solutions that rely on efficient cryptographic tools, including homomorphic encryption. We have given a example of our proposed architectures with the case of the distributed learning by a CNN model applied to distributed medical images databases. The proposed architectures safeguards data privacy thanks to a system of encryption and off-chain storage to avoid the dissemination of sensitive medical data or metadata.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[2] P. Lambin, R. G. P. M. van Stiphout, M. H. W. Starmans, E. Rios-Velazquez, G. Nalbantov, H. J. W. L. Aerts, E. Roelofs, W. van Elmpt, P. C. Boutros, P. Granone, V. Valentini, A. C. Begg, D. De Ruysscher, and A. Dekker, "Predicting outcomes in radiation oncology—Multifactorial decision support systems," *Nature Rev. Clin. Oncol.*, vol. 10, no. 1, pp. 27–40, 2013.

[3] J. Léger, E. Brion, U. Javaid, J. Lee, C. De Vleeschouwer, and B. Macq, "Contour propagation in CT scans with convolutional neural networks," in *Advanced Concepts for Intelligent Vision Systems* (Lecture Notes in Computer Science), vol. 11182. Cham, Switzerland: Springer, Sep. 2018, pp. 380–391.

[4] E. Brion, J. Léger,, U. Javaid, J. Lee, C. De Vleeschouwer, and B. Macq, "Using planning CTs to enhance CNN-based bladder segmentation on cone beam CT," in *Proc. Med. Imag., Image-Guided Procedures, Robotic Intervent., Modeling*, 2019.

[5] A. Barragan-Montero, D. Nguyen, W. Lu, MH. Lin, X. Geets, E. Sterpin, and S. Jiang, "Three-dimensional dose prediction for lung IMRT patients with deep neural networks: Robust learning from heterogeneous beam configurations," 2018, *arXiv:1812.06934*. [Online]. Available: https://arxiv.org/abs/1812.06934

[6] P. Doshi, T. Jefferson, and C. Del Mar, "The imperative to share clinical study reports: Recommendations from the Tamiflu experience," *PLoS Med.*, vol. 9, no. 4, Apr. 2012, Art. no. e1001201.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[8] T. M. Deist, A. Jochems, J. van Soest, G. Nalbantov, C. Oberije, S. Walsh, M. Eble, P. Bulens, P. Coucke, w. Dries, A. Dekker, and P. Lambin, "Infrastructure and distributed learning methodology for privacy-preserving multi-centric rapid learning health care: EuroCAT," *Clin. Transl. Radiat. Oncol.*, vol. 4, pp. 24–31, Jun. 2017.

[9] A. Jochems, T. M. Deist, J. van Soest, M. Eble, P. Bulens, P. Coucke, W. Dries, P. Lambin, and A. Dekker, "Distributed learning: Developing a predictive model based on data from multiple hospitals without data leaving the hospital—A real life proof of concept," *Radiother. Oncol.*, vol. 121, no. 3, pp. 459–467, Dec. 2016.

[10] N. Kshetri, "Blockchain and electronic healthcare records [cybertrust]," *Computer*, vol. 51, no. 12, pp. 59–63, 2018.

[11] J.-S. Weng, J. Weng, L. Ming, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 679, Aug. 2018.

[12] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 1992, pp. 139–147.

[13] S. King and S. Nadal, "PPcoin: Peer-to-peer crypto-currency with proof-of-stake," Tech. Rep., 2012.

[14] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2017.

[15] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy" *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.

[16] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2015.

[17] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.

[18] G. Elahi and E. Yu, "Modeling and analysis of security trade-offs—A goal oriented approach," *Data Knowl. Eng.*, vol. 68, no. 7, pp. 579–598, 2009.

[19] G. Damaskinos, E. M. El Mhamdi, R. Guerraoui, A. Guirguis, and S. Rouault, "AGGREGATHOR: Byzantine machine learning via robust gradient aggregation," in *Proc. 1st SysML Conf.*, Palo Alto, CA, USA, 2019.

[20] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 10–18, Jul. 1985.

[21] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. 3rd Innov. Theor. Comput. Sci. Conf. (ITCS)*, 2012.

[22] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT*, vol. 1592. Berlin, Germany: Springer, 1999, pp. 223–238.

[23] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1351–1362, May 2016.

[24] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Proc. Annu. Int. Cryptol. Conf.*, Cham, Switzerland, 2018.

[25] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI* (Lecture Notes in Computer Science), vol. 9351. Cham, Switzerland: Springer, 2015, pp. 234–241.

SÉBASTIEN LUGAN received the master's degree with École Supérieure d'Ingénieurs en électronique et électrotechnique (ESIEE), France, the DEA degree from the Gaspard-Monge Institute of Electronics and Computer Science, Université Paris-Est Marne-la-Vallée UPEM, France, and the Ph.D. degree in engineering sciences from the École Polytechnique de Louvain, Université Catholique de Louvain, Belgium. Since 2003, he has been with the Pixels and Interactions Lab (PiLAB), Institute of Information and Communication Technologies, Electronics and Applied Mathematics, ICTEAM, Université Catholique de Louvain, UCLouvain, where he is currently a Senior Researcher. His research activities include medical imaging, image compression and mega-image navigation, and LIDAR detection of wake turbulences generated by airliners and secure communications for embedded devices and sensors (e.g., the IoT for smart cities).

PAUL DESBORDES received the master's degree in biomedical engineering and the Ph.D. degree in computer science from the University of Rouen Normandy, France, in 2017. Since, he has been a Senior Researcher with the Université Catholique de Louvain (UCLouvain), Belgium, under the supervision of Prof. B. Macq and funded by the Wallonia region. During his thesis, he became an Expert in Radiomics. This concept is based on advanced medical images processing for the prediction of patients' survival and treatment response in oncology. It leads to a personalized medicine for each patient and an early prediction for medical outcome. His current research aims to design innovative architecture helping practitioners in their daily routine, such as tumour segmentation using artificial intelligence, outcome prediction or consensus algorithm for treatment planning.

ELIOTT BRION received the master's degree in applied mathematics jointly from UCLouvain and CentraleSupélec, in 2016. He is currently pursuing the Ph.D. degree with UCLouvain, with a focus on deep learning for radiotherapy, supervised by Prof. B. Macq and Prof. J. Lee. He currently focuses on organs at risk and tumoral volumes segmentation for radiotherapy. This will allow adaptive treatments that better target the tumour while sparing healthy organs. His Ph.D. research was supported by the Walloon Region, its grant is RW-DGO6-Biowin-Bidmed.

LUIS XAVIER RAMOS TORMO is currently pursuing the bachelor's degree in computer science and molecular biology with the Massachusetts Institute of Technology. His previous work includes investigating the use of neural networks to automatically impute epigenetic marker locations based on other markers, and designing pedagogical simulations to add a computer science component to STEM High School courses. Shortly after graduation, he will be joining Google as a Software Engineering Resident.

AXEL LEGAY received the Ph.D. degree in computer science from the University of Liége, Belgium. He is currently a Professor with UCLouvain. He is also a Founder and Major Contributor of statistical model checking (a statistical variant of model checking effectively used in industry). He contributed to several new techniques for advance static malware analysis. He has long term collaborations with key industry players such as Cisco or Thalés. He is also a Referee for top journals and conferences in formal verification and cyber security. His main research interests are in formal verification and cyber security.

BENOÎT MACQ received the Ph.D. degree from UCLouvain–Belgium, in 1989.

He has been a Researcher with the Philips Research Laboratory, Belgium, in 1990 and 1991, where he developing wavelet-based compression algorithms and contributing to the JPEG-2000 standard. He has been a Professor with the Polytechnic School of UCLouvain, since 1993. He has also been a Visiting Professor with the Ecole Polytechnique Fédérale de Lausanne, Massachusetts Institute of Technology, Boston. He is currently a Visiting Professor with McGill University, Montreal. His main research interests are image compression, image watermarking, image analysis for medical and immersive communications. He is also a member of the Royal Academy of Science of Belgium. He was the General Chair of the IEEE ICIP2011, Brussels. He has been the Chair of several European projects, including a Network of Excellence on Multimodal Interactions (SIMILAR Project) and an Integrated Proposal, EDcine which paved the way of the technology of digital cinema in the European Union. He was an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE TRANSACTIONS ON IMAGE PROCESSING, the Guest Editor of the Proceedings of the IEEE. He is also a Senior Editor Associate of the IEEE *Transactions on Image Processing*. He is also the Founder of the *Journal of Multimodal Interfaces* (Springer Verlag). He is the co-founder of 11 spin-off companies.