EPIC: Efficient Private Image Classification (or: Learning from the Masters)

Eleftheria Makri[1,2], Dragos Rotaru[1,3], Nigel P. Smart[1,3], and Frederik Vercauteren[1]

[1] imec-COSIC, KU Leuven, Belgium.
[2] ABRR, Saxion University of Applied Sciences, The Netherlands.
[3] University of Bristol, UK.

Abstract. Outsourcing an image classification task raises privacy concerns, both from the image provider's perspective, who wishes to keep their images confidential, and from the classification algorithm provider's perspective, who wishes to protect the intellectual property of their classifier. We propose EPIC, an efficient private image classification system based on support vector machine (SVM) learning, secure against malicious adversaries. EPIC builds upon transfer learning techniques known from the Machine Learning (ML) literature and minimizes the load on the privacy-preserving part. Our solution is based on Secure Multiparty Computation (MPC), it is 34 times faster than Gazelle (USENIX 2018) –the state-of-the-art in private image classification– and it improves the communication cost by 50 times, with a 7% higher accuracy on CIFAR-10 dataset. For the same accuracy as Gazelle reaches on CIFAR-10, EPIC is 700 times faster and the communication cost is reduced by 500 times.

1 Introduction

Visual object recognition is an important machine learning application, deployed in numerous real-life settings. Machine Learning as a Service (MLaaS) is becoming increasingly popular in the era of cloud computing, data mining, and knowledge extraction. Object recognition is such a machine learning task that can be provided as a cloud service. However, in most application scenarios, straightforward outsourcing of the object recognition task is not possible due to privacy concerns. Generally, the *image holder* who wishes to perform the image classification process, requires their input images to remain confidential. On the other hand, the *classification algorithm provider* wishes to commercially exploit their algorithm; hence, requires the algorithm parameters to remain confidential.

We consider an approach, which facilitates the outsourcing of the image classification task to an external classification algorithm provider, without requiring the establishment of trust, contractually or otherwise, between the involved parties. We focus on the evaluation task (i.e., labeling a new unclassified image), and not the learning task. Our proposal is based on Secure Multiparty Computation (MPC), and allows for private image classification without revealing anything about the private images of the image holder, nor about the parameters of the classification algorithm. Unlike previous work [4,5,20], we can fully outsource the task at hand, in such a way that the classification algorithm provider does not need to be the same entity as the *cloud computing provider*. Although any of the involved parties (i.e., the classification algorithm provider, and the image holder) can play the role of one of the MPC servers, this is not a requirement for guaranteeing the security of our proposal. MPC allows distribution of trust to two or more parties. As long as the image holder (resp. the classification algorithm provider) trusts at least one of the MPC servers, their input images (resp. their classification algorithm parameters) remain secret.

MPC allows a set of mutually distrusting parties to jointly compute a function on their inputs, without revealing anything about these inputs (other than what can be inferred from the function output itself). Currently, MPC allows one to compute relatively simple functions on private data; arbitrarily complex functions can be supported, but with an often prohibitive computational cost. EPIC, our privacy-preserving image classification solution, combines the techniques of transfer learning feature extraction, support vector machine (SVM) classification, and MPC. In this work we use recently developed techniques for generic image classification (within the ImageNet competition) such as transfer learning to extract powerful generic features. Transfer learning using raw CNN features has been studied extensively by Azizpour et al. [3], and Yosinski et al. [54]. Then, the computation done in the MPC setting is minimized to only evaluate a simple function with secret shared inputs.

We focus on classification via SVM, as opposed to using more sophisticated techniques, such as Neural Networks (NNs), in the privacy-preserving domain to minimize the computational cost. While the field of private image classification is shifting towards NN-based approaches [25,33,42], we show that it is not necessary to use private NNs, as we can achieve classification with better accuracy by using generic NNs to improve the feature extraction techniques used. Although Convolutional NNs (CNNs) are the state-of-the-art for image classification [23], we confirm that SVMs can achieve high accuracy, as long as they are provided with good quality features. Transforming a NN to a privacy-preserving one results in inefficient solutions (e.g., 570 seconds for one image classification by CryptoNets [20]).

A schematic representation of the application scenario treated by EPIC is given in Fig. 1. Using additive secret sharing techniques both the *classification algorithm provider*, and the *image holder* share their inputs to the $n \geq 2$ *MPC servers*. Note that no information about the actual secret inputs can be gained by the individual shares alone. Thus, each MPC server learns nothing about the inputs of the two parties. The cluster of the MPC servers comprise the *cloud computing provider*, which together execute the MPC protocol to produce the final classification result. The MPC servers communicate via authenticated channels to accomplish what the protocol prescribes. The protocol completes its execution by having all MPC servers sending their share of the final classification result to the designated party, who can then reconstruct the result by combining the received shares. This party can be the *image holder*, or an external *analyst*,

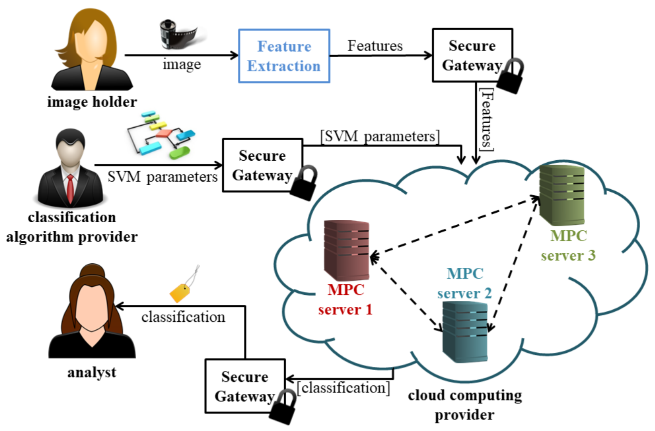

Fig. 1. A schematic representation of the private image classification scenario.

assigned to examine the classification results, without getting access to the underlying private images. Note that the involved parties, namely the image holder, the classification algorithm provider, and potentially the analyst, may play the role of the MPC servers themselves, avoiding completely the outsourcing to the cloud provider(s).

A key aspect of our work is how the data is processed before the MPC engine is used to perform the classification. The SVM classification is performed on so-called feature vectors, and not directly on the images. The way one determines these feature vectors not only affects accuracy, but it also has an impact on security. As shown in Fig. 1 the image holder performs the feature extraction on the input image before it is passed to the secure gateway. Thus this feature extation must not be specific to the algorithm classification provider; otherwise the extracted features could reveal information about exactly what is being classified. We apply a generic feature extraction method, which is independent of the underlying classification task.

In particular, we employ TensorFlow [1], to extract features based on the activation of a deep CNN (specifically the Inception-v3 [46] CNN) trained on a set of object recognition tasks, *different* from the target task. This method is known as CNN-off-the-shelf in the ML literature, and it has been successfully applied in various image recognition tasks [15,41]. Since the CNN is generic, it can be released in the clear, and hence become part of the image holder's preprocessing. This not only gives us a security benefit, but it also significantly improves the accuracy of our method. There are many public CNNs available online for generic feature extraction in Caffe's Model Zoo, which can be used with our EPIC solution to add a privacy dimension to a typical ML problem [24]. In our paper we selected Inception-v3 as the public CNN to extract features, because it suits many generic image recognition tasks, and allows us to benchmark EPIC against previous solutions on traditional datasets such as CIFAR-10.

We also present a second variant of EPIC, which aims at allowing a tradeoff between the accuracy of the classifier's predictions, and its performance. It does so by deploying a kernel approximation method, on top of Inception-v3 features for dimensionality reduction.

We implemented our solution using SPDZ [7], which was introduced by Damgård et al. [12,13], and is based on additive secret sharing. We assume the reader to be familiar with MPC, but we discuss preliminaries on the techniques used by EPIC for completeness in Appendix A. EPIC outperforms the state-of-the-art in secure neural network inference [25], both in terms of efficiency, and in terms of prediction accuracy. Our implementation shows that privacy-preserving image classification has become practical. As shown in Table 1, we are the only provably secure work in the active security model. A system like EPIC could find application in numerous real-life cases, such as in purchase scenarios where visual insepction is performed, or when targeted surveillance is required without compromising non-targets' privacy.

Our contributions are thus four fold: i) We enable full outsourcing of privacy-preserving image classification to a third independent party using a simple technique yet much faster and accurate than others, which require complicated machinery. ii) Our solution does not leak *any* information about the private images, nor the classifier, while being the first to provide active security. iii) We show how to deploy a data-independent feature extraction method to alleviate the privacy-preserving computations, while increasing accuracy and efficiency. iv) We demonstrate the practicality of our approach, both in terms of efficiency, and in terms of accuracy, by conducting experiments on realistic datasets.

2 Related Work

Privacy-preserving machine learning can focus either on providing a secure training phase, a secure classification phase, or both secure training and classification phases. The first research works in the field aimed at designing a privacy-preserving training phase. Recently, due to the advent of cloud computing, and Machine Learning as a Service, more and more works focus on the design of a privacy-preserving classification phase. Fewer works have attempted to address both the training, and the classification phases in a privacy-preserving manner.

To facilitate an easy comparison of the related work, we summarize the main features of each proposal in Table 1.

- The first column of Table 1 is the reference to the corresponding paper.
- The second column indicates whether the work considers secure training (T), secure training and classification (T+C), or only secure classification (C).
- The third column indicates the security model, under which the proposed protocols are secure, where P stands for passive security, and A stands for active security. N/A (not applicable) refers to differential privacy techniques, which are designed to protect against inference about the inputs from the outputs, and thus are orthogonal to the issue of securing the computation which we deal with.

- The fourth column denotes the method used to preserve privacy. DP stands for differential privacy; SP stands for selective privacy, and refers to the unique characteristic of the work of Shokri and Shmatikov [45] allowing the users to decide how much private information about their learned models they wish to reveal. SHE stands for Somewhat Homomorphic Encryption, 2-PC for 2-Party Computation, and MPC stands for Multiparty Computation (which could include 2-PC).
- The fifth column lists the training method(s) used. N-L SVM stands for non-linear SVM, NN for Neural Networks, LM for Linear Means, FLD for Fisher's Linear Discriminant, HD for hyperplane decision, LIR for linear regression, LOR for logistic regression, and DT for decision trees.
- The sixth column lists the information that is revealed by the protocol execution. C stands for information about the classifier, and TD for information about the training data. We note with boldface letters the information that is intentionally revealed by the protocol execution, and we mark with an asterisk the information that is protected by means of differential privacy techniques. Information that can potentially, and unintentionally be leaked is noted with normal, non-boldface letters.
- The last column indicates whether the work provides an implementation.

Training a SVM in a privacy-friendly way, has been previously considered based on techniques of differential privacy [30,31]. Despite the little overhead that these techniques incur, which makes them competitive from an efficiency perspective, they do not consider the security of the actual *computation* during the training or classification phases. Shokri and Shmatikov [45] achieve such privacy-preserving collaborative deep learning with multiple participants, while refraining from using cryptographic techniques. Their work focuses on learning the neural network, but they also consider protecting the privacy of each individual's NN, allowing the participants to decide how much information to share about their models.

A lot of research has been devoted to provable privacy-preserving techniques for training a classifier. Privacy-preserving data mining has been an active research area since the seminal work of Lindell and Pinkas [32]. More recently, Vaidya et al. [50] showed how to train a SVM classifier, in a privacy-preserving manner, based on vertically, horizontally, and arbitrarily partitioned training data. In follow-up work, Teo et al. [47] improved upon the efficiency of the solution of Vaidya et al. [50], and showed that their approach scales well to address the challenges of data mining on big data. Chase et al. [9] combine MPC techniques with differential privacy techniques to achieve private neural network learning. Their work provides provable security guarantees for the learning phase (in the passive security model), and adds noise to the resulting network to protect its privacy during classification.

A parallel research line aiming to address the same challenge, namely privacy-preserving data mining, is based on homomorphic encryption (instead of MPC). The notion of homomorphic encryption dates back to the work of Rivest et al. [43], but only recently fully homomorphic encryption was devised [19]. This

	Func.	**Sec. model**	**Privacy mthd**	**Train mthd**	**Info leak**	**Impl.**
[30]	T	N/A	DP	N-L SVM	**C**; TD*	✓
[31]	T	N/A	DP	N-L SVM	**C**; TD*	✓
[45]	T	P	SP	NN	**C**	✓
[50]	T	P	MPC	N-L SVM	**C**	✓
[47]	T	P	MPC; DP	N-L SVM	**C**	✓
[9]	T	P	MPC; DP	NN	**C***	✓
[21]	T+C	P	SHE	LM; FLD	no	✓
[2]	T+C	P	SHE	Bayes; random forests	no	✓
[29]	T+C	P	2-PC	N-L SVM	no	×
[10]	T+C	P	2-PC	NN	TD	×
[34]	T+C	P	2-PC	NN; LIR; LOR	no	✓
[20]	C	P	SHE	NN	no	✓
[4]	C	P	SHE	N-L SVM	C	✓
[8]	C	P	SHE	NN	no	×
[6]	C	P	SHE; 2-PC	HD; Bayes; DT	no	✓
[40]	C	P	2-PC	N-L SVM	no	✓
[5]	C	P	2-PC	NN	**C**	×
[35]	C	P	2-PC	NN	no	✓
[44]	C	P	2-PC	NN	no	✓
[33]	C	P	2-PC	NN	**filter size**	✓
[42]	C	P	2-PC	NN; SVM	no	✓
[25]	C	P	2-PC	NN	no	✓
EPIC	C	A	MPC	SVM	no	✓

Table 1. Comparison of the related work.

type of homomorphic encryption allows the computation of any polynomial function on the encrypted data, and unlike MPC, it does not require communication, as the task can be outsourced to one single party. Since the seminal work of Gentry [19], a lot of somewhat homomorphic encryption schemes have been proposed, allowing computations of polynomial functions of a limited degree. Graepel et al. [21] consider both machine learning training, and classification based on encrypted data, with their solutions being secure in the passive model. Due to the selected homomorphic encryption scheme, Graepel et al. [21] cannot treat comparisons efficiently, which excludes SVM-based solutions. Addressing both learning, and classification based on extremely random forests, and naïve Bayes networks, Aslett et al. [2], also work on homomorphically encrypted data.

One of the first private SVM classifiers was proposed by Laur et al. [29], which addresses both the training and the classification in a privacy-preserving manner. Their work combines the techniques of homomorphic encryption, secret sharing, and circuit evaluation, into a passively secure 2-PC solution. Concurrently, and independently Dahl [10] is working on using the same MPC framework as in our work, to realize both the training, and the classification of CNN based privacy-preserving algorithms. While Dahl [10] is deploying CNNs instead of SVM, he needs to apply them in a non-black-box fashion. The protocol of Dahl [10] allows some leakage of information during the training phase, which is not the case with our approach. SecureML [34] also considers both training and classification in the 2-PC setting, and the passive security model. These approaches [29,10,34] can only treat the two-party setting, and cannot be trivially extended to allow the classifier provider to be a different entity than the cloud provider.

Other works focus particularly on the private image classification problem, instead of the training of the model. Gilad-Bachrach et al. [20] propose a solution applicable to the image classification problem, based on homomorphically encrypted data. The resulting *CryptoNets* [20] provide an accuracy of 99% for the MNIST dataset, and can make on average 51739 predictions per hour. However, this is only the case when the predictions are to be made simultaneously; for a single prediction the task takes 570 seconds to complete.

Recent work by Barnett et al. [4] demonstrated the potential of polynomial-kernel SVM to be used for classification in a privacy-preserving manner. Specifically, Barnett et al. apply SVM techniques for the classification –as in our work– but on encrypted data. Although they mention the potential of an MPC approach to be more efficient in this setting, they do not consider it, because direct translation of the protocols to MPC would require interaction between the client and the classification algorithm provider during the computations. We overcome this limitation by extending the application scenario in a way that allows the classification task to be fully outsourced to a cluster of independent third parties. We implement their approach using SPDZ in a more secure way by keeping the PCA components private (they choose to make them public). This implementation is more expensive than EPIC, due to the non-linearity of the polynomial SVM, and it is also less accurate. Albeit inefficient and inaccurate, it provides an initial benchmark, and it shows the gap between an FHE and an MPC approach (see details in Section 4). Chabanne et al. [8] attempted to approximate commonly used functions used in NN-based classification in a SHE-friendly manner. Despite the high prediction accuracy that their work achieves, Chabanne et al. do not provide any performance evaluation results.

In the 2-PC setting, Bost et al. [6], and Rahulamathavan et al. [40] focus on the problem of private classification, where both the classifier parameters, and the client's input to be classified need to remain private. The latter approach does not consider linear SVM, while both approaches only offer passive security. Barni et al. [5] propose private NN-based data classification, also in the 2-PC setting and passive security model. They suggest three protocols, which offer different privacy guarantees for the classifier owner, while always protecting fully the client's input. Follow up work by Orlandi et al. [35] extends the work of Barni et al. in terms of privacy. DeepSecure [44] is another work in the 2-PC setting, and the passive security model, using Garbled-Circuit techniques. A direct performance comparison of DeepSecure versus CryptoNets [20] confirmed a significant efficiency improvement achieved by DeepSecure.

The recently proposed MiniONN [33] is one of the latest NN-based data classification approaches in the 2-PC setting. MiniONN demonstrates significant performance increase compared to CryptoNets, without loss of accuracy; as well as better accuracy compared to SecureML [34], combined with increased performance. However, it still operates in the 2-PC setting, which is more restricted than the MPC setting we consider, and it only offers passive security. Under a comparable configuration as MiniONN, and still in the passive security model, Chameleon [42] achieves a 4.2 times performance improvement. Chameleon operates in the 2-PC setting, under the assumption that a Semi-Honest Third Party (STP) is engaged in the offline phase to generate correlated randomness. Despite the strong STP assumption, Chameleon does not need the third party for the online phase, while it gets a significant performance increase from this STP.

Gazelle [25], the latest work on secure NN classification, outperforms, in terms of efficiency, the best previous solutions in the literature [20,33,42], by carefully selecting which parts of the CNN to carry out using a packed additively homomorphic encryption, and which using garbled circuits. EPIC performs better than Gazelle, while also being secure in the active security model. This is because EPIC only treats linear computations in the privacy-preserving domain.

To the best of our knowledge, we are the first to provide a privacy-preserving image classification tool combining SVM classification with transfer learning feature extraction, offering active security. EPIC is more efficient than previous work and achieves prediction accuracy higher than that of the related work on the same datasets, although it does not deploy sophisticated NN-based classification on the private inputs. Interestingly, EPIC is not limited to the 2-PC setting, allowing a broad range of application scenarios to be treated by our solution.

3 EPIC

The proposed private image classification solution, EPIC, is based on transfer learning techniques [48] for feature extraction. The EPIC algorithm for image classification runs in two phases. In the first phase the image is passed through a generic feature extraction method. Being generic, i.e., not task specific, this method can be published in-the-clear and hence can be applied by the image holder before passing the output securely to the MPC engine. In the second phase the actual classification, via an SVM, is applied. This SVM is specific to the task at hand, and hence needs to be securely passed to the MPC engine. We thus have two problems to solve Feature Extraction and SVM classification.

Feature Extraction. High quality features are key to the accuracy of a trained classifier. We ensure high quality feature extraction by deploying the techniques of transfer learning. Specifically, we perform feature extraction based on Inception-v3 [46], which is a *public* CNN classifier trained on a set of non-privacy-sensitive object recognition tasks. Commonly, the training for such a CNN classifier is performed on large datasets, which enhances the prediction accuracy of the classifier. In our context, the trained classifier extracts features based on the activation of a deep convolutional network. Our work shows that powerful feature extraction is essential to the quality of the final classification accuracy. In fact, we demonstrate that the high-complexity (CNN) tasks can be learned on non-private datasets, and still use their power for feature extraction of unrelated tasks. Eventually, this allows us to deploy only linear functions for the actual classification, which enables accurate, and efficient privacy-preserving solutions.

SVM Classification. Despite the increasing popularity and high effectiveness of CNN classification techniques, the direct deployment of CNN techniques requires large training datasets [14] that are potentially difficult to obtain when the underlying data is privacy sensitive. In addition, black-box transformation of CNN-based methods to their privacy-preserving equivalents will result in classifiers that are computationally prohibitive to use. Thus using a light-weight classification method such as SVMs can be beneficial in privacy sensitive environments, and their evaluation can be done (as we show) in a secure manner. With the CNN features, an SVM can learn quickly from very few positive examples, which shows that they are useful to perform one-shot learning [15]. Thus, we opted for the design of a private SVM classifier, while using the techniques of CNN-based *transfer learning* in the context of feature extraction, which does not raise privacy concerns.

To classify a new unlabeled input with our classifier trained with a linear SVM, we need to securely evaluate the following equation:

$$\mathsf{class}(\mathbf{h}) = \arg\max_i(\mathbf{x_i} \cdot \mathbf{h} + b_i), \quad (1)$$

where:

- $\mathbf{h}$ is the vector representing the client's image, and has been provided to the MPC servers in shared form;
- b_i is the model intercept (aka bias), calculated by the classification algorithm provider during the learning phase and secret shared to the MPC servers;
- $\mathbf{x_i}$ are the n support vectors.

The support vectors $\mathbf{x_i}$, and the model intercepts b_i are assumed to need protection, as they represent the intellectual property of the learned model.

Feature Reduction. To achieve efficient training of kernel machines (such as SVM) aimed at non-linear problems, several approximation methods (e.g., the method of Rahimi and Recht [38]) have been proposed. Such approaches have the goal to alleviate the (cleartext) computational, and storage cost of the training, incurred by the high dimensionality of the data, especially when the training datasets are large. The approximation generally is implemented by mapping the input data to a low-dimensional feature space, such that the inner products of the mapped data are approximately equal to the features of a more complex (e.g., Gaussian) kernel. This is known as the kernel trick. These features are then combined with linear techniques (e.g., linear SVM), yielding an efficient training, but also an efficient classification, which we are able to implement in a privacy-preserving way.

One of the first successful approaches for kernel approximations, achieving high accuracy, was proposed by Rahimi and Recht [38], and is based on random features, which are independent of the training data. To the contrary, Nyström based kernel approximations [52,16], are data dependent. Although Nyström approximations outperform randomly extracted features [53] in terms of accuracy, being data dependent makes them unfit for our purposes, as they require applying non-linear functions on the private inputs. From a computational, and storage efficiency perspective, data independent approximations are favored.

We discovered that a variant of the method proposed by Rahimi and Recht [39] is presented in the scikit-learn package [36]. This implements an RBF (Radial Basis Function) sampler, which allows to transform the features without using the training data. This dimensionality reduction (like the feature extraction) is deployed both for the training, and for the data classification. Since the feature selection is random (i.e., data independent), it can be performed on the cleartext data, both by the classification algorithm provider, and by the client, without raising privacy concerns.

Our second variant of EPIC (see below) supports dimensionality reduction for free, by placing all the computational load on the cleartext. This variant makes use of an algorithm implementing the RBF sampler listed in Fig. 2. In our application scenario, the algorithm provider broadcasts the RBF sampler parameters, namely the γ parameter and the feature size. The γ parameter does not reveal any information about the dataset. Note that γ is a floating point number, which is varied to match a cross-validation score on the training data. The shape variable is the feature size of a point (set to 2048), which is the output of Inception-v3.

```
Scikit-learn variant of Random Kitchen Sinks [36].

Init: Set $\gamma, \mathsf{shape}, \mathsf{state}_{\mathsf{random}}, \mathsf{nc}$.
Fit:
    1. Select $\mathsf{weights}_{\mathsf{random}} = \sqrt{2 * \gamma} \cdot \mathsf{state}_{\mathsf{random}}.\mathcal{N}(0,1)$ of size $\mathsf{nc} \times \mathsf{shape}$, with
       mean 0 and standard deviation 1.
    2. Assign $\mathsf{offset}_{\mathsf{random}} = \mathsf{state}_{\mathsf{random}}.\mathcal{U}(0, 2 \cdot \pi)$ of size $\mathsf{nc}$.
Transform($x$):
    1. $\mathsf{projection} = x \cdot \mathsf{weights}_{\mathsf{random}} + \mathsf{offset}_{\mathsf{random}}$.
    2. $\mathsf{projection} = \cos(\mathsf{projection})$.
    3. $\mathsf{projection} = \mathsf{projection} \cdot (\sqrt{2}/\sqrt{\mathsf{nc}})$
```

Fig. 2. RBF Sampler.

EPIC – Simple Variant: The classification algorithm provider has already trained their SVM classifier. The parameters for the SVM classification are shared to the MPC servers by the classification algorithm provider and are never revealed to the image holder (nor the analyst). The image holder applies the Inception-v3 feature extraction to their image, and takes the next to last layer, which has a feature size of 2048, as their output. The resulting features are then shared (via the secure gateway) to the MPC servers by the image holder, and thus are kept secret from the classification algorithm provider. We indicate secret shared data in square brackets (Fig. 1). The MPC engine then evaluates the SVM securely on the features and outputs the result to the analyst (or image holder).

Note that although EPIC does not allow any information leakage about the private SVM parameters, recent work by Tramer et al. [49] showed that only black-box access to the classifiers can still serve to recover an (near-)equivalent model. We consider this problem to be out of this work's scope, as it can easily be tackled by restricting the number of queries an external party is allowed to perform on the MPC Engine. This type of attacks has not been averted by any of the secure computation solutions in the related work.

EPIC – Complex Variant: The second variant of EPIC protocol is summarized in Fig. 3. This EPIC variant trades a small percentage of the classification accuracy to increase efficiency. It achieves this tradeoff by deploying the *kernel approximation* dimensionality reduction explained above, and in particular the kernel approximation sub-step is also considered to be part of the feature extraction phase. Here the algorithm provider needs to publish the feature size of a point (in our case 2048) and the parameter γ from above. At first sight it might seem that γ reveals information about the training data, but we noticed that for our datasets one can fix $\gamma = 2^{-13}$ to a small value and modify the regularization parameter C of the SVM. This parameter C will always remain private to the algorithm provider, hence there is no information leakage. We stress again that for both cases, the CNN feature extraction is input independent, so privacy is maintained for the image holder and algorithm provider.

Specifically, the protocol starts with the Setup phase, where the algorithm provider (AP) performs the kernel approximation (from Fig. 2) on its own dataset, and broadcasts the type of CNN used, and the Init parameters necessary for the feature reduction at the image holder (IH) side. Then, the algorithm provider secret shares the SVM parameters to the MPC Engine (Eng). Secret shared values are denoted in double square brackets. In the evaluation phase, the image holder performs the feature extraction locally (given the previously obtained parameters), and secret shares the new point to be classified by Eng. Then, the MPC protocol computes Eq. 1.

4 Experiments

Experimental Setup. Our experiments are conducted on two MPC servers, which yields the most efficient solution, but we also show how the proposed system scales with more than two MPC servers. We assume a protocol-independent, input-independent preprocessing phase that takes place prior to the protocol execution between the MPC servers. The inputting parties do not need to be aware, nor contribute to this phase. The preprocessing creates the randomness needed to

EPIC Protocol with kernel approximation as feature reduction

Setup:

1. Algorithm Provider (AP) broadcasts the type of CNN used for feature extraction.
2. AP computes γ from Fig.2 on its own training data. Then AP broadcasts the Init variables from Fig. 2 and secret shares the support vectors x_i, b_i to the MPC engine (Eng). These are stored on Eng as $[\![x_i]\!], [\![b_i]\!]$.

Evaluate:

1. Image Holder (IH) uses public CNN to extract features h' from its image. Then IH maps $h' \mapsto h$ locally using the RBF sampler initialized with the γ broadcasted by AP to obtain a smaller number of features. The new point h is further secret shared to the Eng and stored as $[\![h]\!]$.
2. Eng uses $[\![x_i]\!], [\![b_i]\!], [\![h]\!]$ to compute Eq. 1 with a shared result: $[\![\mathsf{class}(\mathbf{h})]\!]$.

Fig. 3. Protocol for SVM classifcation with RBF sampler.

boost the efficiency of the online phase, and allows the inputting parties (image holder and classification algorithm provider) to securely share their inputs.

The online phase begins with the image holder, and the algorithm provider sharing their inputs (reduced CNN features, and SVM parameters, resp.) to the MPC servers. This is performed by executing an interactive protocol between each inputting party and the two MPC servers, as Damgård et al. [11] proposed. Then, the actual private image classification task is executed only between the two MPC servers, as in the **Evaluate** phase of Fig. 3. In the end, each MPC server sends their resulting share to the image holder, or the analyst, who can combine the shares and reconstruct the cleartext result, which is the desired class label.

From Fixed Point Arithmetic to Integers. For the secure integer comparison sub-protocols that EPIC deploys, we selected the statistical security parameter to be $\kappa = 40$ bits. We stress that everywhere the computational security parameter is set to $\lambda = 128$. We observed experimentally after running the scikit-learn's RBF (see Fig. 2) on top of Inception-v3 that each feature is bounded by $\mathsf{abs}(x_i) \leq 15$ where $\mathsf{len}(x) \leq 2048$.

To avoid the costly fixed point arithmetic, we scale each feature x_i by a factor f, and then perform arithmetic on integers. Particularly, we compute $x_i \cdot f$ and then round it to the nearest lower integer. We varied f, and evaluated the SVM's accuracy. We experimentally concluded that setting $f = 2^8$ gives sufficient accuracy, as if working on floating point numbers, while lowering the scale factor f decreased the accuracy by more than 1%. If $f = 2^8$ then to compute a class score from Eq. 1 becomes: $s = \sum_{j=1}^{2048}(2^8 \cdot x_{ij} \cdot 2^8 \cdot h_j) + 2^{16} \cdot b_i$ since we need to scale both support vectors x_i as well features h. Using the fact that each component is bounded by 15 then clearly $s \leq 2^{35}$.

To improve the underlying MPC performance we wanted to aim for using a 64-bit prime modulus for the underlying linear secret sharing scheme. Unfortunately, if our inputs are of 35 bit size then there is no room left to perform the secure comparisons in $\arg\max$ with 40 bits statistical security, as $35 + 40 > 64$. Surprisingly, for all the datasets we experimented with, s was less than 20 bits long, because some of the x_{ij}'s are negative. Hence, we could run everything on 64-bit primes with 40-bit statistical security, while ensuring there is no information leak from the comparisons. We can achieve an even tighter bounding by normalizing the features using the L2-Norm, after the RBF-Sampler invocation. In our setting this is not necessary, since the expected bound on s is already low (20 bits). We also experimented (see later) with higher statistical security of 100 bits by using 128-bit prime fields.

For the feature reduction we considered whether to use RBF or PCA, and concluded that RBF is more suitable for our purposes. Despite the accuracy loss that RBF incurs compared to PCA, it is justified to use RBF for reasons of computational and communication efficiency. For a more detailed comparison between RBF and PCA feature reduction in our setting, we refer the reader to Appendix B.

Datasets. We selected three image datasets: CIFAR-10, MIT-67, and Caltech-101, to show how EPIC scales in terms of performance, when increasing the number of classes, and to illustrate its classification accuracy.

- **CIFAR-10** [28]: This is a dataset of 60000 32x32 color images, out of which 50000 are training images and 10000 are test images. CIFAR-10 features 10 classes of objects, with 6000 images per class. The accuracy metric is the quotient between correctly classified samples and total number of samples.
- **MIT-67** [37]: MIT-67 has 15620 indoor images from 67 scene categories. We used 80 images per class for training, and the rest of the pictures for testing. The accuracy metric used here is the mAP (mean Accuracy Precision), which consists of calculating the average over the accuracies of each class.
- **Caltech-101** [17]: This dataset contains pictures of objects of 102 categories. Each class has at least 31 images and we chose to use 30 images from each class for the training. The accuracy metric is mAP, just as in MIT-67.

Training. We trained the SVM on the cleartext versions of the aforementioned datasets. Feature extraction was done after resizing each image to 256x256. We trained Linear SVMs based on the one-versus-all strategy (OvA) [51], because it is more efficient to evaluate n classifiers in MPC instead of $n(n-1)/2$. Note that we chose to avoid the data augmentation trick, and adopted the training method presented in DeCAF [15] using the original datasets, and raw features from Inception-v3 [15]. To find parameters that yield high classification accuracy, we have done a grid search for the γ required in the RBF, and the parameter C, which denotes the size-margin hyperplane for the SVM decision function.

We stress that EPIC achieves a sufficient classification accuracy. Given that EPIC workings have been purposely kept simple to allow for efficient secure computations, we consider the classification accuracy of EPIC comparable to that achieved by the state-of-the-art (non-privacy-preserving) works in the ML community. The best classification accuracy in-the-clear on the CIFAR-10 dataset is 97.14% [18], while EPIC achieves 88.8%. On the MIT-67 dataset, EPIC achieves 72.2% accuracy, while the state-of-the-art in-the-clear solution [27] reports an accuracy of 83.1%. More interestingly, on Caltech-101, the state-of-the-art accuracy in-the-clear is still 93.42% [22], while EPIC achieves 91.4%.

Classification Accuracy and Performance Evaluation. We executed our experiments, simulating the two MPC servers on two identical desktop computers equipped with Intel i7-4790 processor, at 3.60 GHz over a 1Gbps LAN with an average round-trip ping of 0.3ms.

Our algorithm hand matches the one listed in Fig. 3, where the **Evaluate** step from Fig. 3 was implemented using the SPDZ software [7]. The preprocessing phase for this step was estimated using the LowGear protocol by Keller et al. [26], which is the fastest known protocol to produce triples for multiple parties with active security. We do not report on the timings for the feature extraction and reduction, since they can be done in the clear, locally by the external parties, which provide inputs to the MPC engine, and they are not privacy-sensitive.

EPIC – Simple Variant: We evaluated the computational performance, data sent over the network, and classification accuracy of EPIC on the default 2048 length feature from the output of Inception-v3. We report these experiment results in Table 2. Increasing the number of classes n (from 10, to 67, to 102) has a worsening effect on the performance, as the amount of data sent over the network scales linearly with n. The runtime of the online phase is affected less as n increases. Going from 10 classes (CIFAR-10) with 0.005 seconds runtime, to 102 classes (Caltech) with 0.03 seconds, is an increase factor of six, whereas for all other metrics it is roughly ten (i.e., linear in the number of classes).

In Table 3 we show that EPIC improves over Gazelle [25] in terms of every relevant metric on CIFAR-10: accuracy with 7%, total communication by 50x, and total runtime by 34x. This is because we start with secret shared (powerful) features obtained from public CNNs, whereas Gazelle [25] starts with an encrypted image. We expect Gazelle's timings to considerably improve, if they adopt our approach, starting from encrypted features produced by a public CNN.

EPIC – Complex Variant: To increase the performance of EPIC even further, we tried to minimize the feature size used, while still matching the classification accuracy achieved by Gazelle [25] or MiniONN [33] for CIFAR-10. In the end, we settled with $\mathsf{nc} = 128$, and then performed a grid search on γ for the MIT and Caltech datasets. Our results are reported in Table 4. Since the number of features decreases considerably from 2048 to 128 the timings decrease as well. For example, if we look at the online runtime compared to Gazelle [25], our solution improves by a factor of 700x and the total communication cost decreases by almost 500x. We do recognize that our setting is different from the one considered by Gazelle [25], but we see more the similarities, since the end goal is the same, namely to classify secret shared (or encrypted) images.

Our results indicate that general image recognition, and user's privacy can go well together. In fact we showed that securing the private classification comes nearly for free. A stronger case for why CNN features with a Linear SVM should be considered, as a baseline benchmark is done by Razavian et al. [41].

Other optimizations: Note that one of the major improvements came from running the dot products on multiple threads, and doing the argmax operation in a tree-wise manner to decrease the number of communication rounds required.

Dataset	Runtime (s)			Communication (MB)			Accuracy
	Offline	Online	Total	Offline	Online	Total	%
CIFAR	0.36	0.005	0.37	24	0.33	24.33	88.8
MIT	2.43	0.02	2.45	161.94	2.24	164.18	72.2
Caltech	3.71	0.03	3.74	246.59	3.41	250	91.4

Table 2. 1 Gbps LAN timings for EPIC – Simple Variant on different datasets with a Linear SVM.

Framework	Runtime (s)			Communication (MB)			Accuracy
	Offline	Online	Total	Offline	Online	Total	%
MiniONN [33]	472	72	544	3046	6226	9272	81.61
Gazelle [25]	9.34	3.56	12.9	940	296	1236	81.61
EPIC	0.36	0.005	0.37	24	0.33	24.33	88.8

Table 3. 1 Gbps LAN timings for CIFAR-10 dataset on different frameworks. The EPIC – Simple Variant is compared to the state-of-the-art private classification solutions, and outperforms them in all metrics.

Dataset	Runtime (s)			Communication (MB)			Accuracy
	Offline	Online	Total	Offline	Online	Total	%
CIFAR	0.037	0.0003	0.037	2.472	0.027	2.5	81.74
MIT	0.259	0.002	0.261	17.22	0.180	17.4	64.4
Caltech	0.395	0.004	0.399	26.27	0.273	26.543	85.56

Table 4. 1 Gbps LAN timings for EPIC – Complex Variant on different datasets with a RBF-SVM and a 128 feature size.

Multiparty Setting. We benchmarked EPIC on different number of computers with the RBF-128 variant on the CIFAR-10 dataset and measured throughput (operations per second) for the online and offline phases in Fig. 4. For the two party case our protocol can carry around 2650 evaluations per second. The throughput decreases with a growing number of parties and reaches 870 ops per second for the five parties case. Notice that the main bottleneck when executing these protocols is still the preprocessing phase, generating the necessary triples.

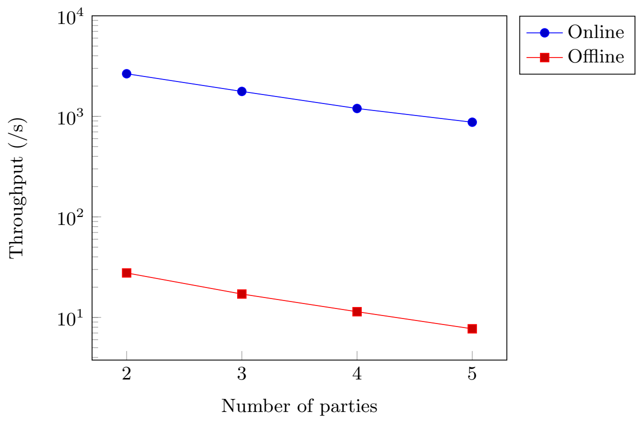

Fig. 4. Throughput of CIFAR-10 evaluations of secret features with RBF-128 EPIC for multiple parties.

Similar work. It is worth mentioning that we also implemented the method of Barnett et al. [4] in SPDZ, after fixing some security bugs such as cleartext PCA coefficients. They report 124s for one binary classification thus to extrapolate this to 10 classes takes roughly 1240s. To translate the work for Barnett et al. in SPDZ we used a feature extraction algorithm based on Histogram of Oriented Gradients (HOG) and then reduced their dimension using PCA. The reduced points where then plugged into a polynomial SVM to classify the inputs. This methodology yielded a 6.7s execution time of the online phase, and an expensive preprocessing phase of 12 hours for CIFAR-10. The classification accuracy was also poor (58%). This showed that the input dependent phase in MPC is faster than in FHE, by at least two orders of magnitude, confirming that our EPIC solution outperforms traditional attempts at classifying images using SVMs.

The closest work to ours that tried to solve the issue of linear SVM classification is a semi-honest 2-PC protocol due to Bost et al [6]. In this work party A owns the model and party B holds the features to be classified. To compare our method with theirs in an accurate manner we took their open sourced code and tailored it to our feature length (2048), input size (27 bits) and computational security $\lambda = 128$ and run it on our computers; whilst maintaining their statistical security of 100 bits. In Table 5 the method of Bost et al. [6] is benchmarked with the recent libraries (NTL-11.3.0, HElib, etc.). We then compare with EPIC using the same parameters as used in the Bost et al experiments, namely statistical security $\kappa = 100$ and computational security $\lambda = 128$, where the shares live in $\mathbb{F}_p$ and $p \approx 2^{128}$ –see Appendix C for more details. According to the experiments, EPIC has a much faster online phase by at least a factor of 20 than Bost et al. at the expense of a slower preprocessing phase. This indicates that the main bottleneck in running the entire protocol is the triple generation method, which deploys expensive cryptographic tools.

Method	Classes	Runtime (s)			Communication (MB)		
		Offline	Online	Total	Offline	Online	Total
[6]	10	0	0.48	0.48	0	5.36	5.36
EPIC	10	1.04	0.014	1.054	46.35	0.66	47.01
[6]	67	0	1.32	1.32	0	36.02	36.02
EPIC	67	7.04	0.058	XXX	312	XXX	XXX
[6]	102	0	1.67	1.67	0	54.85	54.85
EPIC	102	10.72	0.083	10.8	475.96	6.68	482.64

Table 5. 1 Gbps LAN timings for EPIC – Simple Variant and Bost et al. with different number of classes.

5 Conclusion and Future Work

We have introduced EPIC, a private image classification system, trained with SVM, while having the input features extracted based on the techniques of transfer learning. We showed how to achieve privacy-preserving image classification in such a way that the task can be fully outsourced to a third, independent party. For our solution we deployed generic MPC tools and showed how to avoid the restricted two-party setting. Unlike all previous work, our approach provides active security, does not leak any information about the private images, nor about the classifier parameters, and it is orders of magnitude more efficient than the privacy-preserving classification solutions proposed in the literature.

Due to their highly accurate predictions, especially for multiclass classification tasks, CNNs have superseded SVM as the state-of-the-art for image classification. However, our work shows that in the privacy-preserving domain, SVM classification can still produce accurate results, as long as it is provided with high quality features. Thus, we chose to focus on improving the feature extraction phase, using a transfer learning, CNN-based approach, while avoiding the execution of such complex functions in the MPC domain. An interesting advantage of our solution is that it can be applied to the homomorphic encryption domain, since performing the linear operations has depth 1, and the costlier operation is computing the argmax, which requires to branch on secret comparisons.

Our experiments confirmed that there is a tradeoff between the complexity, and therefore also accuracy of the classification algorithms used, versus the efficiency of the privacy-preserving variants of the proposed solutions. In the active security model that we consider in this work, deploying CNNs in the same manner as they are used on cleartext data, is computationally prohibitive with current privacy-preserving methods.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under contract No. N66001-15-C-4070.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: TensorFlow: A System for Large-Scale Machine Learning. In: OSDI. vol. 16, pp. 265–283 (2016)
2. Aslett, L.J., Esperança, P.M., Holmes, C.C.: Encrypted Statistical Machine Learning: New Privacy Preserving Methods. arXiv:1508.06845 (2015)
3. Azizpour, H., Razavian, A.S., Sullivan, J., Maki, A., Carlsson, S.: Factors of Transferability for a Generic Convnet Representation. IEEE transactions on pattern analysis and machine intelligence **38**(9), 1790–1802 (2016)
4. Barnett, A., Santokhi, J., Simpson, M., Smart, N.P., Stainton-Bygrave, C., Vivek, S., Waller, A.: Image Classification using non-linear Support Vector Machines on Encrypted Data. IACR Cryptology ePrint Archive p. 857 (2017)
5. Barni, M., Orlandi, C., Piva, A.: A Privacy-Preserving Protocol for Neural-Network-Based Computation. In: the 8th workshop on Multimedia and security. pp. 146–151. ACM (2006)
6. Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine Learning Classification over Encrypted Data. In: Network and Distributed System Security Symposium (2015)
7. Bristol Crypto: SPDZ-2: Multiparty computation with SPDZ, MASCOT, and Overdrive offline phases. https://github.com/bristolcrypto/SPDZ-2 (2018)
8. Chabanne, H., de Wargny, A., Milgram, J., Morel, C., Prouff, E.: Privacy-Preserving Classification on Deep Neural Network. IACR Cryptology ePrint Archive p. 35 (2017)
9. Chase, M., Gilad-Bachrach, R., Laine, K., Lauter, K., Rindal, P.: Private Collaborative Neural Network Learning. IACR Cryptology ePrint Archive p. 762 (2017)
10. Dahl, M.: Private Image Analysis with MPC: Training CNNs on Sensitive Data using SPDZ. https://mortendahl.github.io/2017/09/19/private-image-analysis-with-mpc/ (2018)
11. Damgård, I., Damgård, K., Nielsen, K., Nordholt, P.S., Toft, T.: Confidential Benchmarking based on Multiparty Computation. IACR Cryptology ePrint Archive p. 1006 (2015)

12. Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical Covertly Secure MPC for Dishonest Majority–or: Breaking the SPDZ Limits. In: European Symposium on Research in Computer Security. pp. 1–18. Springer (2013)
13. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty Computation from Somewhat Homomorphic Encryption. In: Advances in Cryptology, CRYPTO 2012, pp. 643–662. Springer (2012)
14. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A Large-Scale Hierarchical Image Database. In: Computer Vision and Pattern Recognition. pp. 248–255. IEEE (2009)
15. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In: International conference on machine learning. pp. 647–655 (2014)
16. Drineas, P., Mahoney, M.W.: On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. journal of machine learning research **6**(Dec), 2153–2175 (2005)
17. Fei-Fei, L., Fergus, R., Perona, P.: Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In: Computer Vision and Pattern Recognition. pp. 178–178 (2004)
18. Gastaldi, X.: Shake-Shake Regularization. arXiv:1705.07485 (2017)
19. Gentry, C.: Fully Homomorphic Encryption using Ideal Lattices. In: Symposium on Theory of Computing, STOC. vol. 9, pp. 169–178 (2009)
20. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In: International Conference on Machine Learning. pp. 201–210 (2016)
21. Graepel, T., Lauter, K., Naehrig, M.: ML Confidential: Machine Learning on Encrypted Data. In: International Conference on Information Security and Cryptology. pp. 1–21. Springer (2012)
22. He, K., Zhang, X., Ren, S., Sun, J.: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual recognition. In: European Conference on Computer Vision. pp. 346–361. Springer (2014)
23. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely Connected Convolutional Networks. arXiv:1608.06993 (2016)
24. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional Architecture for Fast Feature Embedding. In: International Conference on Multimedia. pp. 675–678. ACM (2014)
25. Juvekar, C., Vaikuntanathan, V., Chandrakasan, A.: GAZELLE: A Low Latency Framework for Secure Neural Network Inference. arXiv:1801.05507 (2018)
26. Keller, M., Pastro, V., Rotaru, D.: Overdrive: Making SPDZ great again. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 158–189. Springer (2018)
27. Khan, F.S., van de Weijer, J., Anwer, R.M., Bagdanov, A.D., Felsberg, M., Laaksonen, J.: Scale Coding Bag of Deep Features for Human Attribute and Action Recognition. Machine Vision and Applications **29**(1), 55–71 (2018)
28. Krizhevsky, A., Hinton, G.: Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto (2009)
29. Laur, S., Lipmaa, H., Mielikäinen, T.: Cryptographically Private Support Vector Machines. In: the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 618–624. ACM (2006)

30. Lin, K.P., Chen, M.S.: Privacy-Preserving Outsourcing Support Vector Machines with Random Transformation. In: the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 363–372. ACM (2010)
31. Lin, K.P., Chen, M.S.: On the Design and Analysis of the Privacy-Preserving SVM Classifier. Knowledge and Data Engineering **23**(11), 1704–1717 (2011)
32. Lindell, Y., Pinkas, B.: Privacy Preserving Data Mining. In: Advances in Cryptology, CRYPTO 2000. pp. 36–54. Springer (2000)
33. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious Neural Network Predictions via MiniONN Transformations. In: the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 619–631. ACM (2017)
34. Mohassel, P., Zhang, Y.: SecureML: A System for Scalable Privacy-Preserving Machine Learning. IACR Cryptology ePrint Archive p. 396 (2017)
35. Orlandi, C., Piva, A., Barni, M.: Oblivious Neural Network Computing via Homomorphic Encryption. EURASIP Journal on Information Security p. 18 (2007)
36. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
37. Quattoni, A., Torralba, A.: Recognizing Indoor Scenes. In: Computer Vision and Pattern Recognition. pp. 413–420. IEEE (2009)
38. Rahimi, A., Recht, B.: Random Features for Large-Scale Kernel Machines. In: Advances in neural information processing systems. pp. 1177–1184 (2008)
39. Rahimi, A., Recht, B.: Weighted Sums of Random Kitchen Sinks: Replacing Minimization with Randomization in Learning. In: Advances in neural information processing systems. pp. 1313–1320 (2009)
40. Rahulamathavan, Y., Phan, R.C.W., Veluru, S., Cumanan, K., Rajarajan, M.: Privacy-Preserving Multi-Class Support Vector Machine for Outsourcing the Data Classification in Cloud. Dependable and Secure Computing **11**(5), 467–479 (2014)
41. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN Features off-the-shelf: An Astounding Baseline for Recognition. In: Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 512–519. IEEE (2014)
42. Riazi, M.S., Weinert, C., Tkachenko, O., Songhori, E.M., Schneider, T., Koushanfar, F.: Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. arXiv:1801.03239 (2018)
43. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On Data Banks and Privacy Homomorphisms. Foundations of secure computation **4**(11), 169–180 (1978)
44. Rouhani, B.D., Riazi, M.S., Koushanfar, F.: DeepSecure: Scalable Provably-Secure Deep Learning. arXiv:1705.08963 (2017)
45. Shokri, R., Shmatikov, V.: Privacy-Preserving Deep learning. In: 22nd ACM SIGSAC conference on computer and communications security. pp. 1310–1321. ACM (2015)
46. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception Architecture for Computer Vision. In: Computer Vision and Pattern Recognition. pp. 2818–2826 (2016)
47. Teo, S.G., Han, S., Lee, V.C.: Privacy Preserving Support Vector Machine using non-linear Kernels on Hadoop Mahout. In: the 16th International Conference on Computational Science and Engineering (CSE). pp. 941–948. IEEE (2013)
48. Thrun, S.: Is Learning the n-th Thing any Easier than Learning the First? In: Advances in neural information processing systems. pp. 640–646 (1996)
49. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing Machine Learning Models via Prediction APIs. In: USENIX. pp. 601–618 (2016)

50. Vaidya, J., Yu, H., Jiang, X.: Privacy-Preserving SVM Classification. Knowledge and Information Systems **14**(2), 161–178 (2008)
51. Vapnik, V.N., Vapnik, V.: Statistical Learning Theory, vol. 1. Wiley New York (1998)
52. Williams, C.K., Seeger, M.: Using the Nyström Method to Speed up Kernel Machines. In: Advances in neural information processing systems. pp. 682–688 (2001)
53. Yang, T., Li, Y.F., Mahdavi, M., Jin, R., Zhou, Z.H.: Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison. In: Advances in neural information processing systems. pp. 476–484 (2012)
54. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How Transferable are Features in Deep Neural Networks? In: Advances in neural information processing systems. pp. 3320–3328 (2014)

A Preliminaries on Secure Multiparty Computation

Secure Multiparty Computation (MPC) is a cryptographic method allowing a set of parties to jointly compute a function on their inputs, without revealing the inputs to the rest of the parties. The two main security models used to realize MPC are the *passive*, and *active* security model. In the passive security model, we assume that the protocol participants follow the protocol specification honestly, but they try to learn as much information as possible about the private inputs, during the protocol execution. In the active security model, we assume that the protocol participants may actively, and arbitrarily deviate from the protocol specification. Clearly, the active security model offers stronger security guarantees. In both models we can construct protocols that require an honest majority of the protocol participants to guarantee security, or protocols that guarantee security assuming a dishonest majority of the protocol participants. Our solution offers strong security guarantees, providing active static security, with a dishonest majority. This means that an adversary may corrupt, prior to the protocol execution, up to $n-1$ out of the n protocol participants, without leaking any private information, and without allowing any false protocol output to be accepted as correct.

Our solution is implemented using the SPDZ MPC framework [12,13], and that is why it enjoys the aforementioned security properties. The computational and communication costs of the constructed protocols increase linearly in the number of protocol participants. SPDZ is based on additive secret sharing, allowing the participants to share their private inputs, in such a way that no information about the private inputs is revealed to the individual participants. Additive secret sharing enjoys an additively homomorphic property, meaning that any linear function can be directly computed on the shares that each protocol participant holds, without requiring interaction amongst the parties. Upon reconstruction of the shared output (which requires all parties to send their shares of the secret), the result will be the correct result of the linear function, as if it had been applied on the secret input. Thus, we can perform additions, and multiplications with non-secret constant values on the secret shared inputs.

To perform multiplications between secret shared inputs, or any other non-linear operation, we need to execute a secure interactive protocol between the MPC servers.

The SPDZ approach works in two phases: a preprocessing phase, and an online phase. The preprocessing phase can take place offline, anytime prior to the execution of the online phase. This phase only requires the MPC servers to be online, and not the inputting parties. During this phase the MPC servers create shared randomness, which the client, and the classification algorithm provider can later use to securely share their private inputs. Moreover, the MPC servers create shared random values to be consumed during the online phase, and make it efficient. For the online phase, the inputting parties first need to provide the MPC servers with their private inputs. This is performed in a secure manner, based on the Output Delivery protocol, and Input Supply protocol, proposed by Damgård et al. [11]. Then, the MPC servers proceed with the secure computation of the function prescribed by the protocol transcript. For more details on the MPC techniques used, we refer the reader to the work of Damgård et al. [13,12].

B Feature reduction: RBF or PCA?

From the algorithm provider's point of view the goal is to keep the training data private, so we choose to make public only the parameters of the RBF-Init (Fig. 2). Publishing γ leaks no information about the training data, since if one fixes γ in advance, they can vary the regularization parameter C to modify the accuracy of the linear SVM. In our case we always kept $\gamma = 2^{-13}$, and then just tuned C accordingly.

Our experiments showed that the RBF technique reduces the prediction accuracy, which was expected as it decreases the number of components remaining after reduction (see Table 4). On the other hand, PCA requires to compute the equation: $z = A^{\intercal} \cdot (h - m_h)$, which takes as input a point h and outputs a point z with a length equal to the number of columns in A (denote this as nc). Note that A, h, m_h are all secret shared across the MPC engine. If we reduce the number of components with PCA to $\mathsf{nc} = 256$, then we observed that we get around 5% better accuracy than using RBF with the same feature reduction parameter nc.

The main caveat of PCA is that we need to perform $(\mathsf{shape} \cdot \mathsf{nc})$ secure multiplications to convert our features, and then another $(\mathsf{nc} \cdot n)$ secure multiplications for computing the SVM probabilities. Hence, PCA incurs a total of $\mathsf{nc}(\mathsf{shape} + n)$ multiplications in the privacy-preserving domain, whereas RBF just $(\mathsf{nc} \cdot n)$. In application scenarios where the performance is of higher importance, and classification accuracy can be sacrificed (up to $< 8\%$), our RBF approach is still slightly more accurate than other private CNN's (Table 4: CIFAR-10 accuracy, versus Table 3: accuracy of MiniONN [33], and Gazelle [25]).

C Security parameters in SPDZ

The traditional way to split the computation in SPDZ is an input independent preprocessing phase, where we produce triples, and an online phase, where we plug in secret or public inputs and perform multiplications using the generated triples.

Offline Phase: Since we use the LowGear [26] protocol to produce triples there are two different flavors of parameters: a statistical κ and a computational λ. These are reflected in different instances such as:

- Computational security λ_{FHE} is given by the FHE parameter selection.
- Statistical security κ_{zk} used as the soundness parameter in proofs of plaintext knowledge.
- Statistical security κ_{sac} when sacrificing one triple to check the validity of another one. When the triples are generated for inputs inside a field $\mathbb{F}$ large enough then usually $\kappa_{\mathsf{sac}} = \log_2 \mathbb{F}$, assuming one sacrifices one triple per output triple.

Online Phase: In the online phase we have no computational security parameters but we do have different statistical security parameters:

- Statistical security for the MAC-check κ_{mac} which ensures that an active adversary can pass a check with probability $2^{-\kappa_{\mathsf{mac}}}$. When the inputs are shared within a field $\mathbb{F}$ with one MAC per triple then $\kappa_{\mathsf{mac}} = \log_2 \mathbb{F}$. This can be increased at the expense of introducing more MAC values.
- Statistical security κ_{comp} for integer arithmetic (especially comparisons). When a k-bit integer value is opened during comparison it is masked with at least κ_{comp} bits to guarantee that the output is within statistical distance of $2^{-\kappa_{\mathsf{comp}}}$ of a uniform distribution. Thus, to do a comparison in a field $\mathbb{F}$ we need to make sure the value of k is less than $\log_2 \mathbb{F} - \kappa_{\mathsf{comp}}$.

Global Statistical Security: We define the global statistical security parameters as

$$\kappa = \min(\kappa_{\mathsf{zk}}, \kappa_{\mathsf{sac}}, \kappa_{\mathsf{mac}}, \kappa_{\mathsf{comp}}).$$

In our paper we use two different settings for benchmarking:

- **Standard benchmarks.** These correspond to the experiments reported in Table 2 and Table 4. Here the task was to aim for at least 40 bit statistical security. Hence we first fixed $\lambda_{\mathsf{FHE}} = 128$ and $\kappa_{\mathsf{zk}} = \kappa_{\mathsf{comp}} = 40$. Since the secret inputs are integers of at most 24 bits the other parameters such as $\log_2 \mathbb{F}, \kappa_{\mathsf{sac}}, \kappa_{\mathsf{mac}}$ can all be set to 64.
- **Bost et al. comparison.** These correspond to the experiments reported in Table 5. Here the task was to aim for at least 100 bit statistical security. As before, we first fix $\lambda_{\mathsf{FHE}} = 128$ and $\kappa_{\mathsf{zk}} = \kappa_{\mathsf{comp}} = 100$. Then other parameters have to accommodate the inputs bit length so $\log_2 \mathbb{F}, \kappa_{\mathsf{sac}}, \kappa_{\mathsf{mac}}$ were all set to 128.