Image Classification using non-linear Support Vector Machines on Encrypted Data

Anthony Barnett[1], Jay Santokhi[1], Michael Simpson[1], Nigel P. Smart[2], Charlie Stainton-Bygrave[1], Srnivas Vivek[2], and Adrian Waller[1]

[1] Thales UK Research and Technology, E-mail:
Anthony.Barnett@uk.thalesgroup.com, Jay.Santokhi@uk.thalesgroup.com, Michael.Simpson@uk.thalesgroup.com, Charlie.StaintonBygrave@uk.thalesgroup.com, Adrian.Waller@uk.thalesgroup.com

[2] University of Bristol, E-mail: nigel@cs.bris.ac.uk, sv.venkatesh@bristol.ac.uk

Abstract. In image processing, algorithms for object classification are typically based around machine learning. From the algorithm developer's perspective, these can involve a considerable amount of effort and expertise to develop, which makes them commercially valuable. On the other hand, other parties may want to make use of these algorithms to classify their images, while protecting the privacy of their data. In this paper, we show how non-linear Support Vector Machines (SVMs) can be practically used for image classification on data encrypted with a Somewhat Homomorphic Encryption (SHE) scheme. Previous work has shown how an SVM with a linear kernel can be computed on encrypted data, but this only has limited applicability. By enabling SVMs with polynomial kernels, a much larger class of applications are possible with more accuracy in classification results.

1 Introduction

Image processing is used to automate the extraction of useful information from raw image data. An important example is object classification, where the real-life objects in an image are identified algorithmically (e.g. type of vehicle, person or animal, land coverage type etc.). Algorithms for object classification are typically based around machine learning. Examples include the application of Decision Trees [QYLL10] and Neural Networks [Pop17]. The focus of this paper is the application of Support Vector Machines (SVMs) to object classification, which have been shown to provide accurate results in many cases (e.g. [RGP16,WH16,LLLH17]).

Such classification algorithms can involve a considerable amount of effort and expertise to develop and train, which makes them commercially valuable. Therefore, those who develop such algorithms may not wish to hand over the details of how they work to others. On the other hand, many people and organisations wish to make use of these algorithms in order to classify objects in their image data. They will often have their own security and privacy concerns about handing over their captured images to the algorithm provider.

At present, this problem is resolved contractually. One or other of the image data provider and algorithm provider agree to hand over their sensitive information, and trust the other party not to abuse it. This is not ideal, as it is hard to detect any infringements and a remedy through the courts is likely to be highly costly and ultimately unsatisfactory.

Homomorphic Encryption (HE) presents the opportunity by which the image classification algorithms can be executed on encrypted data. In this context, image data can be encrypted by the data provider and sent to the algorithm provider. The algorithm provider can run their sensitive object classification algorithm on the encrypted data, and pass the encrypted result back to the data provider. The data provider then decrypts the result. In this way, image data and classification results are never released to the algorithm provider. Similarly, details of the algorithm (in particular algorithm parameters) are never released to the data provider. In this way, the two parties do not have to trust each other in order to cooperate in a way that is beneficial to both. As an additional advantage, the image data is never available in clear at the algorithm provider. It is therefore protected from accidental or malicious release, and frees up the algorithm provider from the responsibilities of its protection (e.g. the implications of data privacy regulations). However, it would appear at first sight that current homomorphic encryption technologies are too slow for such a setting bar for models which are inherently linear in nature. In this work we show that this initial opinion is not necessarily correct.

1.1 Homomorphic Encryption

Fully Homomorphic Encryption (FHE) was initially introduced as a concept shortly after the development of the RSA cryptosystem, by Rivest et al. [RAD78]. Although long sought after, the first functional scheme was only proposed over thirty years later by Gentry [Gen09a,Gen09b] in 2009. The same blueprint to construct FHE has been followed in all subsequent work. First a scheme is constructed which can evaluate arithmetic circuits of a limited depth, a so-called Somewhat Homomorphic Encryption (SHE) scheme. If the complexity of the circuits which the SHE scheme can evaluate is slightly more than the complexity of the decryption circuit for the SHE scheme, then (by placing a SHE encryption of the scheme's private key inside the public key) one can bootstrap the SHE scheme into a FHE scheme. This bootstrapping operation is obtained by homomorphically evaluating the decryption circuit on input the ciphertext to be bootstrapped and the encryption of the secret key.

So far, there have been roughly three generations of SHE schemes. The first generation consisted of Gentry's original scheme, which was based on having two representations of a basis of an ideal of a number field, one easy basis and one hard basis. Gentry's original scheme was simplified and implemented in

[GH11,SV10], where the ideal was chosen to be principal, with the easy basis being the principal generator and the hard basis being the standard two element representation of this ideal. A second family in the first generation of schemes was based on the approximate-GCD problem, and consisted of so-called "integer based" schemes [vDGHV10]. The first family in the initial generation schemes is now considered insecure due to work of Cramer et al [CDPR16], who extended the work of Campbell et al [CGS14] to solve the problem of finding small generators of principal ideals in cyclotomic number fields. The second family of first generation schemes is not considered competitive compared to the second generation schemes.

The second generation schemes were all based on the Learning With Errors (LWE) problem, and its generalisation to rings (the Ring-LWE problem) [BGV12,BV11b,BV11a]. These schemes, generally referred to as BGV, were extensively optimized and implemented in a series of works by Gentry et al [GHS12b,GHS12c,GHS12a], with an implementation (HELib) being given in [HS14]. A variant of BGV, called FV, was presented in [FV12] which embeds the message into the upper bits of the underlying ring. The second generation systems also include those based on the NTRU assumption [BLLN13,LTV12], although the security of these has since been called into question [ABD16]. A third generation of schemes, based on standard LWE and encoding messages via matrix eigenvalues, was presented in [GSW13].

In this work we shall concentrate on second generation SHE schemes, and in particular the BGV family based on the Ring-LWE problem. When used with "large" plaintext modulus space such schemes have been shown to be able to encode, and operate on homomorphically, real and complex numbers [DGBL$^+$15,CSVW16,CSV16,BBB$^+$]. Such numbers are encoded as polynomials with small coefficients, and then parameters of the SHE scheme are chosen to ensure that the required computation does not make the encoding polynomials grow too large. Homomorphic processing of real numbers is needed in order to perform the SVM computation. Indeed, the methodology to compute on real and complex numbers we adopt was originally introduced to process Fourier Transforms of images [CSVW16,CSV16]. However, in our application the use of SHE techniques is much simpler, and requires much less processing.

However, when used with a SHE scheme one is only able to operate on encodings of integer, real and complex numbers up to a given function complexity. This function complexity is usually measured in terms of the multiplicative depth of the associated arithmetic circuit (where we use the term arithmetic circuit to include also "circuits" over the real and complex numbers).

1.2 Prior work on Image Classification on Encrypted Data

Image classification algorithms cannot simply and directly be used with SHE, and work is required in order to identify which algorithms may be possible, and how to adapt them to be suitable for use with SHE schemes. For example, feature extraction needs to be tailored in order to minimise potentially expensive SHE operations during classification. In this paper we show how classification using a Support Vector Machine (SVM) with a polynomial kernel can be performed on data encrypted with an SHE scheme. Previous work has shown how an SVM with a linear kernel can be computed on encrypted data, but this only has limited applicability. Indeed in Section 4 we show that using higher degree kernels can result in more accurate classification. Thus by enabling SVMs with polynomial kernels, a much larger class of applications are possible with more accuracy in classification results.

Considering previous work more fully: In [TGP13] the authors describe a non-interactive face verification algorithm on SHE encrypted data. Their paper is aimed at a subtly different problem where the classification algorithm is also outsourced to a third-party along with the data to be classified. It uses an SVM classifier, but only a linear SVM is employed. Thus, the main differences are that it is tailored only to face recognition and not more general object classification, and only a linear SVM has been employed.

In [ZW14], the authors also apply classifiers on SHE encrypted data. However, they explicitly consider a different model to the classifier use case we have solved for. In addition, they only provide some building blocks, in particular an encryption scheme to homomorphically encrypt vectors of integers. Extending from these building blocks to implementing SVMs on polynomial kernels is mentioned as a possibility but left as a topic for further research.

In [YBKS17], the authors apply the Paillier homomorphic scheme to aggregate multiple classifiers updated locally using private data. However, they do not perform the actual classifications using a homomorphic encryption scheme.

Several previous works have considered applying Multi-Party Computation (MPC) techniques to the problem. For example, in [RPV^{+}14] and in [THL13], the authors solve the same classification problem as our paper using MPC. However, for polynomial kernel SVMs this requires online interaction between the client and server. In [BPTG15] the authors also implement classifiers using 2-party MPC. However, they do not explicitly provide a solution to an SVM with a polynomial kernel. They say that other classifiers than the ones it considers could be constructed from its building blocks, but gives no indication of if or how this could be done for the specific case of SVM with polynomial kernel.

In all of the MPC based solutions interaction is required between the parties during computation, and may use up a lot of bandwidth for more complex calculations. In our solution based on SHE, clients provide data and get back the results. Our method using SHE is therefore conceptually simpler, and allows classification computations to be performed asynchronously to data provision.

Other related work has considered the learning phase of an SVM classifier. For example, in [NSS^{+}17] the authors describe a method whereby the learning phase of an SVM can be performed by an untrusted third party. It is therefore a different problem to the scenario for our paper. In addition, it is a partial solution in that only some things are encrypted, and less robust data anonymisation techniques are used to attempt to hide other information. In [YJV06] and [ZM06], the authors also consider the training stage of an SVM, and a different model whereby multiple parties wish to cooperate to train a model but not release their data. They are therefore not applicable to the classification scenario our paper is aimed at.

1.3 Our Contribution

We show that image classification on encrypted data using a combination of SVM and SHE can be performed in a reasonable amount of time. We show that the resulting homomorphic SVM can classify encrypted images with high accuracy (note that the inherent precision loss in using homomorphic operations means that a high accuracy an algorithm operating on clear data may not translate into high accuracy when operating on encrypted data). We also present a methodology to enable the output of the SVM to be revealed, without significant leakage of the SVM model parameters. The methodology essentially masks the model output in such a way that recovering the model parameters from multiple queries is similar to solving a system of "non-linear equations with noise". Since "linear algebra with noise" (a.k.a. LWE) is known to be hard, we conjecture that the non-linear version our masking method employs also leads to a hard problem.

Our methodology assumes an encrypted feature vector is given to an evaluator, who applies a SVM whose parameters are known to the evaluator, the final classification being obtained by a third party (who could be the original encryptor) who decrypts the output. Two obvious extensions are possible to the current work, which future work we aim to investigate: Firstly one could also encrypt the SVM parameters to enable processing by a third independent party. This would increase the depth of the circuit by one. Secondly using MPC, as opposed to FHE, may be more efficient in this scenario to determine the output of a private SVM applied to a private image.

2 Problem Setting

To motivate the need for secure image recognition and explain our problem setting, we can consider an example satellite based use case. We note that other potential use cases in other areas are possible. We concentrate on the case of satellites so as to fix the readers mind on a potential application.

Consider two parties involved: the client (Alice) who has some image data obtained from the satellite; and a third party algorithm provider (Bob). Alice wishes to identify objects within her image data in an automated way (e.g. vegetation type, buildings, vehicles). Suppose that Bob has access to a significant amount of training data, and the capability to use machine learning to create a sophisticated and accurate image recognition algorithm. This could be trained to recognise ground objects of interest in the satellite imagery. The cost of processing power, access to training data and development time would make it unattractive for Alice to develop such an algorithm on her own and instead she wishes to make use of the services of Bob. However, if Alice's imagery is sensitive, then this poses the problem of trust with using the third party Bob to process the data. From Bob's point of view, he will be unwilling to hand over details of the algorithm to Alice, as there is a risk of losing his investment.

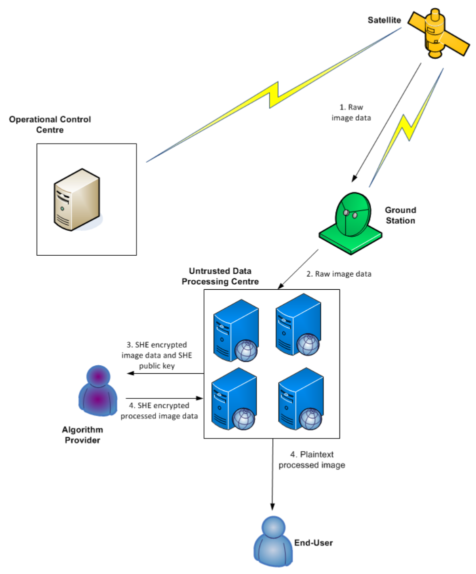
Satellite
1. Raw image data
Operational Control Centre
Ground Station
2. Raw image data
Untrusted Data Processing Centre
3. SHE encrypted image data and SHE public key
4. SHE encrypted processed image data
Algorithm Provider
4. Plaintext processed image
End-User

Fig. 1. Example satellite based scenario of privacy preserving object recognition

Our solution using SHE is illustrated in Figure 1. The Data Processing Centre (DPC) is owned or trusted by Alice, and is where she receives and processes the satellite image data. During setup, Alice generates parameters and public/private key material for an SHE scheme. The parameters and public key are distributed to the algorithm provider (i.e. Bob), and Alice keeps hold of the private key at the DPC in order to control decryption. During setup, Bob only needs to provide Alice with details of the features that his classification algorithm makes use of, to allow image pre-processing to take place at the DPC.

During operation, in Steps labelled 1 and 2 in the figure, raw image data is captured at the satellite and sent to the DPC via a Ground Station. Once the data is received at the DPC it is ready for image processing algorithms to be applied. The DPC first performs any non-sensitive pre-processing steps on the unencrypted data. Examples could include data preparation steps such as a Fast Fourier Transform (FFT), or object detection algorithms to identify the area of an image that is of interest. Then, in Step 3, features are extracted to create feature vectors compatible with the model created by Bob. Alice then uses the SHE public key to encrypt the feature vectors before sending these to Bob. On receipt of the encrypted data, Bob applies his machine learning based object classification algorithm to obtain an encrypted result. Finally, in step 4, the encrypted result is returned to the DPC, which then decrypts the result using Alice's private key to obtain the desired classification. Note that it may be possible for the DPC to recover the Bob's sensitive model parameters, by obtaining multiple classification results. This is because the value returned from the algorithm provider will not in general be a binary classification, but an algorithm output value that could leak information about the model. This issue is considered in more detail in section 4.2 for the case of SVMs.

In this work we do not consider protecting the operations in Step 3, namely the extraction of the feature vectors. Such an extraction mechanism (which depends on the training Bob has previous done) may also be sensitive data. However, in practice such extraction could be the application of a linear function, i.e. the output of a precomputed PCA analysis. In other words the output of Step 3 are vectors $\mathbf{y}$ obtained from the original input data vectors $\mathbf{x}$ via some linear transformation $\mathbf{y} = A^{\mathsf{T}} \cdot (\mathbf{x} - \mathbf{m})$. The matrix A and vector $\mathbf{m}$ being obtained in the training step by the algorithm provider Bob. If Bob wishes to keep A and $\mathbf{m}$ secret then he can ask Alice to supply an encrypted version of $\mathbf{x}$, and then the homomorphic application of A and $\mathbf{m}$ to obtain $\mathbf{y}$ is purely a linear operation and hence is essentially "for free". Thus protecting Step 3 is trivially performed.

There are various factors which make such a SVM based application suitable for securing with SHE:

– The multiplicative depth of the classification algorithm. The greater the multiplicative depth, the less efficient the SHE scheme will be, which can vary significantly slow down operations. SVMs with polynomial kernels typically have low multiplicative depth.
– Simple feature extraction. As described below, feature extraction is a critical step for accuracy of classification. As the number of features grows, the running time of the classification algorithm grows, especially given the slow nature of SHE operations The feature extraction is essentially a linear operation and is hence fast in our application.
– The accuracy of classification. To be useful, the algorithm still has to give reasonable performance in classification. This requires a trade-off between multiplicative depth, feature extraction and running time.

In this paper we show how SVM with polynomial kernel based classification can be practically used on encrypted data, by utilising SHE and a tailored feature extraction phase.

3 Method

The solution presented in this paper is the application of a Support Vector Machine (SVM) trained with a Polynomial Kernel (PK) to SHE encrypted data. We will refer to these as PK-SVMs. In the rest of this paper we outline the method we have used and provide background on SVMs, Kernel Methods, Histogram of Gradients and Principal Component Analysis feature extraction (HoG-PCA) that form the main elements in our method.

3.1 Support Vector Machines(SVMs)

In pattern recognition and machine learning, a feature vector is an f-dimensional vector in $\mathbb{R}^f$ of numerical features that represent some object. The vector space associated with these vectors is the called the feature space. An SVM is a supervised binary machine learning algorithm (see for example [Alp10] for an introduction to SVMs). If we have a dataset of feature vectors that belong to one of two classes, and also their corresponding binary class label, then we can train an SVM to classify unlabelled examples of these feature vectors.

Training an SVM means running the algorithm on a subset of feature vectors for which we know the binary class label. The algorithm is a vector space based machine learning method where the goal is to find a decision boundary between two classes that is maximally far from any point in the training data. The algorithm achieves this by finding the widest "corridor" it can place in-between the two classes of the training data within the feature space. The decision boundary is then placed in the middle of this widest corridor and an identified example can be classified by simply checking which side of the decision boundary the examples lies on.

The margin is the distance between the decision plane (or boundary) and the edge of the corridor. This assumes that the data can be separated into both classes in the feature space by a corridor and is called a "hard-margin SVM". However, for real datasets this is often not the case and instead a "soft-margin SVM" is applied. The soft-margin SVM algorithm associates a cost with misclassifying an example that is accounted for in the objective function maximised in the algorithm. In this way, the margin (or corridor) chosen is a tradeoff between geometric width and the number (and extent) of misclassifications of training examples. This means SVMs can be applied to non-separable data because a few outlier misclassifications will not lead to the choice of extreme decision boundaries.

The SVM algorithm optimises the normal vector $\mathbf{w}$ and intercept b of the decision plane, which is given by the equation:

$$\mathbf{x} \cdot \mathbf{w} + b = 0.$$

For an unclassified example $\mathbf{z}$, evaluating the sign of the plane equation $\mathbf{z} \cdot \mathbf{w} + b$ tells you which side of the decision plane the feature vector lies on and can be interpreted as a class label. The classification of an example using an SVM is therefore as simple as evaluating the sign of an expression involving a vector dot product and an addition. SVMs which classify in this way are known as linear SVMs and the pair $(\mathbf{w}, b)$ is all that is needed to specify the model. In the set-up of our use case this classification is simple enough that an additive homomorphic encryption scheme would be applicable, such as Paillier [Pai99]. This is the approach taken in previous work.

However, this limits us to the use of linear SVMs. Linear SVMs work very well for simple classification tasks such as the recognition of hand written digits or object silhouettes. However, training a non-linear SVM using a kernel can lead to much better accuracy on more complicated images found in the real world. So it is to this more complicated methodology that we now turn our attention.

3.2 Polynomial Kernels

A common example of a non-linear SVM is the use of a Polynomial Kernel (PK-SVM). See [Alp10] for a full description of this approach, however, we briefly recap the details below. "Kernel methods" owe their name to the use of kernel functions, which enable SVMs to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space. Instead, this can be achieved by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates and is known as the "kernel trick".

We classify an example with an SVM trained with a kernel in the same way as a linear SVM, except that we use the Support Vectors (SVs) to calculate the dot product with the higher dimensional normal vector implicitly. The classification equation becomes a sum over all the support vectors in the model. When we use a PK-SVM the classification equation for an unclassified example $\mathbf{z}$ becomes:

$$\mathsf{class}(\mathbf{z}) = \mathsf{sign}\big[b + \sum_{i=1}^{n} a_i \cdot y_i \cdot (1 + \mathbf{x_i} \cdot \mathbf{z})^d\big], \tag{1}$$

where

- $\mathbf{x_i} \in \mathbb{R}^f$ are the SVs, and the index i runs over the set of n SVs.
- b is the model intercept.
- a_i are the Lagrange multipliers, which are the direct output from the SVM optimisation problem.
- y_i are the class labels of the SVs. These are always taken to be $+1, -1$ for above or below the decision plane. In practice we only store $a_i \cdot y_i$.
- d is the order of the PK used.

The order d of the PK-SVM is used as a parameter with which to select a model. Selecting $d = 1$ reduces the model to a linear SVM. However, we are not limited to a linear boundary but can add degrees of flexibility by selecting higher kernel orders. The power d gives a multiplicative depth of $O(\log_2 d)$; hence we now require $\mathbf{z}$ to be encrypted with an SHE scheme for d of a reasonable size, say up to 10.

Equation 1 is the important equation that we are interested in evaluating when $\mathbf{z}$ is encrypted. Note that the model values $(\mathbf{x_i}, y_i, a_i, b, d)$ will in our scenario be held in the clear, by the Algorithm Provider, internally.

3.3 Feature Extraction

Good quality feature extraction is especially important in reducing the number of ciphertext operations required to evaluate Equation 1 on encrypted examples.

Table 1 below counts the number of ciphertext operations required for the homomorphic evaluation of the inner summation in Equation 1 (i.e. the equation without the sign determining operation) and shows that they rely heavily on the number of features used and the number of SVs in the model. Hence, the feature extraction phase is crucial.

Operation	$c + p$	$c + c$	$c * p$	$c * c$
Count	$n + 1$	$n \cdot f - 1$	$n \cdot (f + 1)$	$n \cdot (\lfloor \log_2 d \rfloor + \mathsf{HW}(d) - 1)$

Table 1. HE operations count for PK-SVM classification. Key is: c= ciphertext, p= plaintext, n= number of SVs, f= number of features, d= order of PK-SVM, $\mathsf{HW}(d)$ = Hamming weight of d.

In our experiments we applied a combination of Histogram of Gradients (HoG) and Principal Component Analysis (PCA) for this feature extraction phase. The basic summary of the method (which we expand below) is that in the HoG phase an image is reduced to 142 features, represented by a vector $\mathbf{h}$. The HoG extraction of features is standard, and reveals no model information. The HoG feature space is then reduced down to f features using PCA in a dimension reduction step. We now elaborate on these two steps in more detail:

HoG Feature Extraction: HoG is a technique that has been used in image processing to extract features useful for object recognition (see [DT05] for example). We start with a 32×32 pixel image. Note that 32×32 could be considered a small size for an image. However, even such small images give good classification results and in practice larger images would be compressed to this size to give improved running times. In additon, it is the number and quality of features extracted that fundamentally affects performance, and this is not directly dependent on image size. The image is divided into small rectangular regions called cells, for our 32×32 pixel images we take a cell size of 8×8 pixels, giving a total of 16 cells which cover the image.

A histogram of gradients is created for the pixels within each cell. The gradient at a pixel is a measure of the intensity change across that pixel in the image, and has both a magnitude (the level of intensity change) and a direction (the direction in which the intensity change is greatest). Gradients are useful in detecting lines and edges in images, for example, a vertical line will lead to a large magnuitude intensity change in the horizontal direction. The gradients for a cell can be sorted into a histogram, where the bins in each histogram are ranges of gradient direction and the values sum up the magnitudes for gradients whose direction is in the respective bin. A typical approach would be to use 9 bins per histogram and so the total number of HoG features is (roughly) 9 times the number of cells that fit into the image size. Since we have 16 cells and each cell gives 9 bins, this means we have 142 HoG features.

PCA Dimension Reduction: PCA is a standard technique used in machine learning for dimensionality reduction (see [Alp10]). It is a standard technique that converts the (likely correlated) features into a smaller set of linearly uncorrelated features called principal components. There is no definitive rule for selecting the number of principal components as they are data dependent, but the usual method is to either account for a minimum percentage of the total variance or remove the components that contribute less than a given percentage of the total or largest variance. PCA is often used in conjunction with HoG to reduce the dimension of the feature vector further (see for example [SSK14]). The PCA dimension reduction depends on the problem instance, and the associated PCA matrix $A \in \mathbb{R}^{f \times 142}$ is determined by Bob in the training phase. The reduction operation is then given by

$$\mathbf{z} = A^{\mathsf{T}} \cdot (\mathbf{h} - \mathbf{m_h}) \, ,$$

where $\mathbf{m_h}$ is the vector of means of the training data. As remarked earlier, in our experiments, we assume, for simplicity, that A and $\mathbf{m_h}$ are given to Alice in the clear. But Alice could provide the encrypted values $\mathbf{h}$ to Bob who then applies the above linear operation himself.

In summary, HoG is good at selecting highly discriminative features, whilst PCA is good at selecting few features. By combining the two we, hopefully, end up with a minimal number of discriminative features. PCA helps largely to reduce f (the number of features) to be as small as possible. The number of SVs can be considered a measure of how separable the data is, so this too is a case of feature extraction. Extracting poorly discriminative features results in less separable data and more SVs.

4 Results

We first discuss the feature extraction applied on our test data, for which we used the CIFAR-10 dataset [CIF]. Then we go on to discuss how to ensure the output, obtained by the decryptor, does not allow them to recover the model; i.e. how to maintain the privacy of the model over many queries. Then we discuss the homomorphic evaluation of the SVM algorithm.

4.1 Optimised Feature Extraction for HE

Firstly, we ran experiments to validate assumptions about PK-SVMs offering the potential for greater accuracy in object recognition.

We tried identifying classes of real imagery from the CIFAR-10 dataset and found that the linear models were less effective. Training with a PK-SVM made a significant improvement in classification accuracy. Table 2 provides an example of this, where we trained PK-SVMs on 30,000 examples to identify real images of cars from the 9 other remaining classes of objects in CIFAR-10. Note that a PK-SVM of order 1 is the same as a linear SVM. A HoG cell of 8 pixels was used and principal components with a variance of less than 2% of max variance were discarded generating $f = 57$ HoG-PCA features on a training set of 30,000 examples containing 3,023 positive examples. Note that the output of the HoG stage is 142 features and therefore this experiment also shows the benefit of the combination of HoG and PCA by reducing the number of features to only 57, whilst still maintaining reasonable classification accuracies.

PK-SVM order (d)	Number of SVs (n)	Accuracy	Training Time(s)
1	5,631	92.89%	28
2	3,732	96.89%	42
3	4,353	99.60%	94
4	5,140	100.00%	112

Table 2. PK-SVM trials on CIFAR-10 ‘car’/‘not car’ identification task. Using $f = 57$ HoG-PCA determined features. Training time measured on an Intel Xeon CPU E5-1620 v3 @3.5GHz with 32GB RAM.

PK-SVM order (d)	Number of SVs (n)	Accuracy	$c+p$	$c+c$	$c*p$	$c*c$
1	5,631	92.89%	5,632	320,966	326,598	-
2	3,732	96.89%	3,733	212,723	216,456	3,732
3	4,353	99.60%	4,354	248,120	252,474	8,706
4	5,140	100.00%	5,141	292,979	298,120	10,280

Table 3. PK-SVM trials on CIFAR-10 ‘car’/‘not car’ identification task using HoG-PCA outputing $f = 57$ features.

PK-SVM order (d)	Cell Size	PCA Variance Cut-off	Number of Features (f)	Number of SVs (n)	Accuracy	$c+p$	$c+c$	$c*p$	$c*c$
2	4	0.08	29	1,191	94.55%	1,192	34,538	35,730	1,191
3	8	0.04	37	1,587	95.41%	1,588	58,718	60,306	3,174
4	8	0.02	57	2,024	96.38%	2,025	115,367	117,392	4,048

Table 4. “SHE optimized” PK-SVM trials on CIFAR-10 ‘car’/‘not car’ identification task using HoG-PCA optimised for SHE. Accuracy is an average of 5 trials, with 9,000 randomly selected training images and 10,000 test images in each trial. The PCA Variance Cut-off is the minimum variance of the selected principal components as a percentage of the max variance across the first principal component.

In this example, PK-SVM improves the classification accuracy, which demonstrates the usefulness of the technique. More generally, applying PK-SVM to classification tasks increases the space of models to select from compared to just using linear SVMs. Applying a PK-SVM can therefore at worst add no additional benefit, but otherwise only improve classification accuracy. However, using a PK-SVM as opposed to a linear SVM does increase the computational complexity of applying the model.

Using the same 'car'/'non car' identification task on CIFAR-10 we established the homomorphic complexity of using the output of the HoG-PCA feature extraction. Table 3 gives the counts for the same trials presented above but this time counting the number of homomorphic operations needed to evaluate the inner sum of the SVM equation (i.e. all bar the sign determiniation). One should read the columns as the number of plaintext (p) and ciphertext (c) combined operations (addition and multiplications). Note, adding a ciphertext to a plaintext results in a ciphertext, and so on. Also note, that the ciphertext operations are not equal in terms of computational expense. Ciphertext multiplication ($c * c$) can be taken to be significantly the most expensive.

As one can see the number of homomorphic multiplications increases as one increases the degree d; but increasing d also increases the accuracy of the model. We hence, tried to minimize the complexity of the model, and hence minimize the cost of homomorphically evaluating it, whilst still trying to maintain accuracy. Thus we aimed for different HoG processing and values of the number of feature vectors f; whilst still maintaining accuracy.

There are two key HoG-PCA parameters that we optimised over; the size of the HoG cell and the minimum PCA variance percentage (which determines the number f of retained PCA feature vectors). The HoG cell can be considered as the resolution of the HoG features. The minimum PCA variance percentage is the minimum variance of the principal components selected as features as a percentage of the max variance across the first principal component. To do this we simply performed a 'trial and error' approach on the parameters for each trial. In the trials we adjusted the degree d (2,3 or 4), the HoG cell (4 or 8) and minimum PCA percentage (1,2,...,20) and trained models for each of the different combinations obtained. For each model we captured the classification accuracy, as well as f and n. For each value of $d = x$, we selected the model that minimised the number of homomorphic multiplications whilst still increasing accuracy over the best model for $d = x - 1$.

We found that the number of required homomorphic operations could be greatly reduced taking this approach. For example, we were able to use 29 HoG-PCA features to train an order 2 PK-SVM that achieved an average of 94% accuracy (on 10,000 test examples) when trained on 9,000 randomly selected training examples (from the 30,000 total examples). The model generated an average of 1,226 SVs and average ciphertext operation counts are summarised in Table 4. The size of HoG cell used was 4 pixels and the minimum PCA% was 8%.

4.2 Privacy of Model

The solution proposed in this paper aims to protect privacy both of the client's data as well as the model of the algorithm provider. The privacy of the client's data is successfully preserved. However, there are two remaining issues that need consideration in terms of protecting the algorithm provider's model from the client.

Firstly, the client needs to do pre-processing of images to extract features suitable for the algorithm provider's model. Therefore, the algorithm provider has to reveal details of the feature extraction used. This does leak some information, but on its own it is not useful to the client. And as explained earlier this is trivial to secure as the extraction of feature vectors is a linear operation.

Secondly, recall the key Equation 1

$$\mathsf{class}(\mathbf{z}) = \mathsf{sign}\big[b + \sum_{i=1}^{n} a_i \cdot y_i \cdot (1 + \mathbf{x_i} \cdot \mathbf{z})^d\big].$$

It is this equation which we will be evaluating homomorphically. All the model variables $(b, a_i, y_i, \mathbf{x_i})$ are cleartext values, but the data values $\mathbf{z}$ are assumed to be encrypted. Since the values $\mathbf{z}$ are real values we encrypt them in the BGV SHE scheme using the encoding method described in [CSV16] or [BBB$^+$]. The evaluation of the inner bracket can then be done using a depth $O(\log_2 d)$ circuit. The only thing remaining is how to evaluate the final sign function.

The trivial solution is to simply decrypt the value inside the bracket, which we shall denote by v, and then allow the decryptor to obtain the sign himself. However, it is clear that after a number of queries the model variables $(b, a_i, y_i, \mathbf{x_i})$ will then be able to be determined by the decryptor; by simply solving the resulting non-linear equations. Thus a method is needed to homomorphically evaluate the sign function.

The problem is that the sign function is itself a high depth circuit; and thus evaluating it homomorphically will be prohibitively expensive. Thus instead of doing this we homomorphically mask the value v and then open the masked value. This is inspired by a similar method in [NS14]. The evaluator generates random real numbers r_1 and r_2 with $r_1 > r_2 > 0$ and then computes (homomorphically) $t = r_1 \cdot v + r_2$.

Now if $v > 0$ then t will also be greater than zero, and if $v < -r_2/r_1$ then t will also be less than zero. Thus from the sign of t one can work out the sign of v, except for values of v in the range $[-r_2/r_1, 0)$. In order to select suitable values for r_1 and r_2 we captured the values of v from a number of experiments with our developed classification models. These showed for v that the smallest magnitude was approximately 0.02 and the largest was 11. We therefore chose $r_1 \in [1000, 9999]$ and $r_2 \in [0, 20]$. In theory, the magnitude of v could be less than 0.02 and hence some selections of random r_1 and r_2 could lead to misclassifications. However, the probability of this is small. Classification accuracies given in this section from our experiments with these values of r_1 and r_2 also bear out that these are suitable choices.

The question arises as to whether the unmasked value t reveals information about the masked value v? Alas some information clearly leaks as if the decryptor knows the range from which the evaluator samples r_1, and the expected range of v, then the value of t (if at the extremity of the expected range) will reveal some information about v. However, from this partial information it seems a difficult problem to recover the model. In some sense the resulting problem becomes solving a degree $d+1$ algebraic "equation with noise". It is known that degree one equations with noise are computationally hard (i.e. the Learning with Errors problem), thus one can expect a higher degree analogue to also be computationally hard.

			$\|\|p_{\min}\|\| \geq$		$\deg_{\min} \geq$	
PK-SVM order (d)	f	n	Worst-case	Expected	Worst-case	Expected
2	29	1,191	37	22	76	75
3	37	1,587	51	30	106	105
4	57	2,024	67	38	136	135

Table 5. Worst-case minimum and the expected values for the size (in bits) of the plaintext modulus and the minimal value for the degree of the plaintext ring needed for homomorphic evaluation.

PK-SVM order (d)	p	N	ℓ	$\log_2 q$	Time (s)
2	4147267	8192	7	149	8.97
3	952727777	16384	12	277	62.08
4	269106186949	16384	13	318	124.59

Table 6. Degree N of the rings, number of HELib levels ℓ, and resulting maximum ciphertext modulus q for each of our three PK-SVM models; given a plaintext modulus p; we also present the average time needed to evaluate the SVM on encrypted data.

4.3 Classification on SHE Encrypted Data

In this section we outline the results we obtain in evaluating the SVM on encrypted data. As explained earlier to simplify the implementation we assume encrypted values of the f-dimensional feature vector are given to the evaluator. From this data they need to homomorphically evaluate the inner sum in Equation 1, mask the result, and pass the resulting ciphertext back to the user.

In setting up our experiments there we need to determine two parameter sets. Firstly we need to determine the precise modulus needed for the plaintext ring so as to ensure correctness of the result (assuming no errors are introduced by the homomorphic operations), then we need to estimate the parameters for the SHE scheme itself (the ring dimension and the ciphertext modulus) so as to ensure that the encryption is secure, and the homomorphic operations are performed without error. We use HELib [HS14] as our base SHE library, and so we use the parameter generation methods included there, to define the ring dimension and ciphertext moduli.

Mention that in our feature extaction/model all the values are real values. We encode these using the balanced base-B encoding as the degree of the equation is quite small. If d becomes bigger one may then need to use encoding methods such as the LLL method of [CSV16] or the NIBNAF method of [BBB$^+$], which reduce the degree bound restriction and coefficient size respectively.

Choosing the Plaintext Ring: The coefficients of the SVM model $(\mathbf{x}_i, a_i, b)$ are real numbers which we encode in a plaintext ring of the form $R_p = \mathbb{Z}_p[X]/(X^N + 1)$, where N is a power of two, using the balance base-3 encoding method from [DGBL$^+$15,CSVW16]. Thus each real number becomes encoded as an element in R_p, with $\alpha \in \mathbb{R}$ being encoded as a polynomial $P_\alpha(X)$ such that $P_\alpha(3)/B^t \approx \alpha$ for some value of t.

If we examine the required operations on the plaintext then we need to ensure that the final polynomial, representing the plaintext of the result, does not wrap around in the ring R_p. In particular this means that the ring dimension N must not be too small, and the plaintext modulus needs to be large enough to ensure all coefficients are held exactly. In practice the lower bound found for N here will be surpassed by the lower bound needed for security of the following encryption, but lower bounding the plaintext modulus p is crucial for correctness.

Using the method described in [CSVW16] one can obtain an upper bound on the largest coefficient that could arise in a polynomial when evaluating the inner sum in Equation 1; we also obtain an upper bound on the resulting degree N. If we wished to absolutely guarantee correctness then we would need to define the minimum size (in bits) $p_{\min}$ of the plaintext modulus, and the minimum degree of the ring $\deg_{\min}$ to be greater than these values. See Table 5 for these values.

This worst-case analysis corresponds to the case when all the coefficients in the input base-3 encodings equals unity. But this is really a worst-case since for randomly chosen inputs, all the three possibilities $\{-1, 0, 1\}$ are equally likely for the coefficients in the initial encodings. Hence we also list the expected minimum values for the size of the plaintext modulus p and the degree of the ring N that normally works well in practice. These values were computed as maximum over 1000 trails with random inputs of the given precision.

Choosing the Ciphertext Ring: A (fresh) ciphertext in the HELib system is a BGV ciphertext [BGV12] consisting of two elements in R_q, for some ciphertext modulus $q \gg p$. Security is gauranteed by the size of N and q. We allowed HELib to select N and q, given our desired security level of 80-bits of security. Due to the large plaintext modulus space the number of levels in the HELib system does not correspond to the multiplicative depth $\log_2 d$ of the arithmetic circuit being evaluated.

For each PK-SVM order d considered above (and the resulting f, n values and *expected* minimum plaintext size) we experimentally found the required number of levels of the BGV scheme required by HELib to obtain a correct decryption. We then used the associated N and q values obtained from HELib. These values are summarized in Table 6.

Homomorphic Evaluation Results: Finally in Table 6 we also present the wall-clock times obtained from evaluating our three PK-SVM models on encrypted data. The computations were performed on a 24 core Dell Precision T7600 Workstation with twin Intel Xeon E5-2620 processors, each running at 2.5GHz with 6 physical cores (12 logical cores), and 64GB DDR3 Quad Channel RAM.

Acknowledgements

The work in this paper was supported in part by the European Union's H2020 Programme under grant agreement number ICT-644209. The fourth author was also supported by ERC Advanced Grant ERC-2015-AdG-IMPaCT and by EP-SRC via grant EP/N021940/1. The authors would like to thank Charlotte Bonte for introducing us to one to the technique of blindng the output value.

References

ABD16. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 153–178, 2016.

Alp10. Ethem Alpaydin. *Introduction to Machine Learning*. MIT press, 2 edition, 2010.

BBB$^+$. Charlotte Bonte, Carl Bootland, Joppe W. Bos, Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Faster homomorphic function evaluation using non-integral base encoding. Cryptology ePrint Archive, Report 2017/333. `http://eprint.iacr.org/2017/333`.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

BLLN13. Joppe W. Bos, Kristin E. Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, pages 45–64, 2013.

BPTG15. Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS 2015*. The Internet Society, February 2015.

BV11a. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

BV11b. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011.

CDPR16. Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 559–585. Springer, Heidelberg, May 2016.

CGS14. P. Campbell, M. Groves, and D. Shepherd. SOLILOQUY: A cautionary tale. ETSI 2nd Quantum-Safe Crypto Workshop, 2014.

CIF. CIFAR-10. CIFAR-10 - Object Recognition in Images. `https://www.kaggle.com/c/cifar-10/data`. Accessed: 2017-08-10.

CSV16. A. Costache, N.P. Smart, and S. Vivek. Faster homomorphic evaluation of discrete fourier transforms. Cryptology ePrint Archive, Report 2016/1019, 2016. `http://eprint.iacr.org/2016/1019`.

CSVW16. A. Costache, N.P. Smart, S. Vivek, and A. Waller. Fixed point arithmetic in SHE scheme. Cryptology ePrint Archive, Report 2016/250, 2016. `http://eprint.iacr.org/2016/250`.

DGBL$^+$15. Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Manual for using homomorphic encryption for bioinformatics, 2015. Available at `http://www.microsoft.com/en-us/research/publication/manual-for-using-homomorphic-encryption-for-bioinformatics`.

DT05. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, pages 886–893, 2005.

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `http://eprint.iacr.org/2012/144`.

Gen09a. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.

Gen09b. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

GH11. Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 129–148. Springer, Heidelberg, May 2011.

GHS12a. Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 1–16. Springer, Heidelberg, May 2012.

GHS12b. Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482. Springer, Heidelberg, April 2012.

GHS12c. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Heidelberg, August 2012.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

HS14. Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2014.

LLLH17. Peifeng Liang, Weite Li, Donghang Liu, and Jinglu Hu. Large-scale image classification using fast SVM with deep quasi-linear kernel. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 1064–1071, 2017.

LTV12. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

NS14. Mehrdad Nojoumian and Douglas R. Stinson. Efficient sealed-bid auction protocols using verifiable secret sharing. In *Information Security Practice and Experience - 10th International Conference, ISPEC 2014, Fuzhou, China, May 5-8, 2014. Proceedings*, pages 302–317, 2014.

NSS$^+$17. Ken NAGANUMA, Susumu Serita, Yoshinori Sato, Hisayoshi Sato, and Masayuki Yoshino. Support vector machine learning system and support vector machine learning method, 02 2017.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

Pop17. Calin-Adrian Popa. Complex-valued convolutional neural networks for real-valued image classification. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 816–822, 2017.

QYLL10. Zhixin Qi, Anthony Gar-On Yeh, Xia Li, and Zheng Lin. Integrating object-oriented image analysis and decision tree algorithm for land use and land cover classification using RADARSAT-2 polarimetric SAR imagery. In *IEEE International Geoscience & Remote Sensing Symposium, IGARSS 2010, July 25-30, 2010, Honolulu, Hawaii, USA, Proceedings*, pages 3098–3101, 2010.

RAD78. Ron Rivest, Leonard Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180, 1978.

RGP16. Elmar Rendon-Gonzalez and Volodymyr Ponomaryov. Automatic lung nodule segmentation and classification in CT images based on SVM. In *2016 IEEE 9th International Kharkiv Symposium on Physics and Engineering of Microwaves, Millimeter and Submillimeter Waves (MSMW), Kharkiv, Ukraine, 20-24 June, 2016*, pages 2987–2996, 2016.

RPV$^+$14. Yogachandran Rahulamathavan, Raphael Chung-Wei Phan, Suresh Veluru, Kanapathippillai Cumanan, and Muttukrishnan Rajarajan.

Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud. *IEEE Trans. Dependable Sec. Comput.*, 11(5):467–479, 2014.

SSK14. Andreas Savakis, Riti Sharma, and Mrityunjay Kumar. Efficient eye detection using HOG-PCA descriptor. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 9027, 02 2014.

SV10. Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 420–443. Springer, Heidelberg, May 2010.

TGP13. Juan Ramón Troncoso-Pastoriza, Daniel González-Jiménez, and Fernando Pérez-González. Fully private noninteractive face verification. *IEEE Trans. Information Forensics and Security*, 8(7):1101–1114, 2013.

THL13. Sin G. Teo, Shuguo Han, and Vincent C. S. Lee. Privacy preserving support vector machine using non-linear kernels on hadoop mahout. In *16th IEEE International Conference on Computational Science and Engineering, CSE 2013, December 3-5, 2013, Sydney, Australia*, pages 941–948, 2013.

vDGHV10. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43. Springer, Heidelberg, May 2010.

WH16. Zijun Wei and Minh Hoai. Region ranking SVM for image classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2987–2996, 2016.

YBKS17. Ryo Yonetani, Vishnu Naresh Boddeti, Kris M. Kitani, and Yoichi Sato. Privacy-preserving visual learning using doubly permuted homomorphic encryption. *CoRR*, abs/1704.02203, 2017.

YJV06. Hwanjo Yu, Xiaoqian Jiang, and Jaideep Vaidya. Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data. In *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, 2006*, pages 603–610, 2006.

ZM06. Justin Zhijun Zhan and Stan Matwin. Privacy-oriented collaborative learning systems. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Taipei, Taiwan, October 8-11, 2006*, pages 4102–4105, 2006.

ZW14. Hongchao Zhou and Gregory W. Wornell. Efficient homomorphic encryption on integer vectors and its applications. In *2014 Information Theory and Applications Workshop, ITA 2014, San Diego, CA, USA, February 9-14, 2014*, pages 1–9, 2014.