Unsupervised labelling of stolen handwritten digit embeddings with density matching

Thomas Thebaud, Gaël Le Lan, Anthony Larcher

Unsupervised labelling of stolen handwritten digit embeddings with density matching

Thomas Thebaud[1,2][0000−0001−8953−7872], Gaël Le Lan[2][0000−0002−1493−5777], and Anthony Larcher[1][0000−0003−4398−0224]

[1] LIUM - Le Mans University, 72085 Le Mans, France
{firstname}.{lastname}@univ-lemans.fr https://lium.univ-lemans.fr/
[2] Orange Labs, 35510 Cesson-Sevigne, France
{firstname}.{lastname}@orange.com

Abstract. Biometrics authentication is now widely deployed, and from that omnipresence comes the necessity to protect private data. Recent studies proved touchscreen handwritten digits to be a reliable biometrics. We set a threat model based on that biometrics: in the event of theft of unlabelled embeddings of handwritten digits, we propose a labelling method inspired by recent unsupervised translation algorithms. Provided a set of unlabelled embeddings known to have been produced by a Long Short Term Memory Recurrent Neural Network (LSTM RNN), we demonstrate that inferring their labels is possible. The proposed approach involves label-wise clustering of the embeddings and label identification of each group by matching their distribution to the label-relative classes of a comparison hand-crafted labeled set of embeddings. Cluster labelling is done through a two steps process including a genetic algorithm that finds the N-best matching hypotheses before a fine-tuning of those N-candidates. The proposed method was able to infer the correct labels on 100 randomised runs on different dataset splits.

Keywords: Label inference · Handwritten digits · Density matching · Privacy · Long Short Term Memory · Recurrent Neural Network · Genetic search .

1 Introduction

The generalising use of biometrics for authentication [11] brings personal data in the center of security systems. Most recent biometric systems [11] encode biometric data, such as gait sequences [14], voice recording [19], faces [15], fingerprints [23] or handwritten digits [21] [20] [13], into high dimensional representations commonly named embeddings. Encoding is done through trained classifiers such as Convolutional Neural Networks [15] [23] for physiologic biometries or Recurrent Neural Networks [13] [14] for behavioural ones. Those embeddings are then compared to authenticate whether the user accessing the system is the same as the one who was previously enrolled. When stored and transferred between devices, embeddings are subject to theft and represent a possible breach

in a system's security. However, unlabelled embeddings alone are not enough to apply commonly known attacks by reconstruction [2] [5] [15] and labels are required. To label those stolen embeddings we propose to match to a hand-crafted set of labelled embeddings, using unsupervised techniques. The task of unsupervised matching of embeddings has already been vastly explored for machine translation [6]. Most common methods involve Adversarial training [3], Normalisation flow [24], Wasserstein distances [9], Procrustess analysis [9], Principal Component Analysis [10] and Stochastic optimisations [10].

In this paper, we examine the threat of an unlabelled embedding database theft, all embeddings being extracted from handwritten digits in the context of a One-Time-Password authentication system [21]. The embeddings are computed from handwritten digits, thus contain information about writer identity and digit value. This paper focuses on digit value (label) retrieval, which to our opinion is the first problem to address in this threat scenario. The embeddings are computed from handwritten digits, the number of classes is known to be 10, and we make the hypothesis that the feature extractor is known to be based on LSTMs (standard architecture for that kind of sequence data [13]).

Due to the small number of classes and the simple nature of the data, we suppose that an attacker can find another database of raw handwritten digits, create his own classifier and compute his own set of embeddings for labelling purposes. Inspired by various unsupervised bilingual translation methods [10] [9] [24] [6], we investigate whether it is possible to compute the optimal transformation between the stolen set of unlabelled embeddings and the comparison set of labelled embeddings to infer the labels of the stolen embeddings. Being able to label stolen embeddings and map them to a known space (the output space of the attacker's classifier), pose a security risk into biometric systems [2] [5]. Mai et al. [15] showed that original face images can be reconstructed from face embeddings, using the black-box feature extractor that was used to compute them. Here we only use unlabelled embeddings to get the transformation from their proper space to the output space of a known feature extractor and then guess their labels.

In this paper, our contributions are :

- To question whether it is possible to infer labels of unknown handwritten digits embeddings from their statistical distribution.
- The combination of unsupervised translation methods for label inference.
- The successful labelling of those embeddings and the estimation of a transfer function to map them into a known space.

In section 2, we expose related works about embedding matching and their limits. Section 3 presents the proposed attack scenario. Section 4 details the proposed method to infer the stolen embeddings labels. In section 5, we present the data, the feature extractor architecture and the pre-processing steps. Finally, section 6 presents the experimental work, before section 7 concludes and presents our future works.

2 Related Work

2.1 Template reconstruction attacks

Biometric recognition systems compute templates from physiological or behavioural characteristics by using neural networks which are often referred to as feature extractors. The resulting templates are then used for authentication purpose. Cappelli et al. [2], Galbally et al. [5] and Mai et al. [15] respectively proposed critics of the actual biometric templates by showing that fingerprints, iris and faces templates can be reconstructed using neural networks.

Here we focus on the deep face template reconstruction [15]. Having access to real deep face templates and the black-box feature extractor, the authors generate artificial faces from noise vectors thanks to a generative adversarial network [7]. Artificial deep face templates are then computed from those artificial faces thanks to the feature extractor. Artificial face and template pairs are used to train a neighborly de-convolutional network (NbNet) that infers the inverse function of the feature extractor, i.e., compute the generated artificial face images from the artificial face templates. Finally they use that NbNet network to compute real face images from the real face templates. Their work shows that face images can be retrieved from stolen face templates and a black box feature extractor.

Our work differs from the work of Mai et al. [15] as we deal with *unlabelled* stolen templates of *handwritten digits* and assume to know the architecture of the feature extractor (without any knowledge of the weights of the network). We aim to find the digit values (labels) of unlabelled templates, further called *embeddings* for coherence with the unsupervised machine translation literature cited in this paper. We propose to find the labels of stolen embeddings by matching their space to the output space of a known, hand-crafted, comparison feature extractor (here a LSTM RNN), using a transfer function. Once the transfer function is found, that known RNN feature extractor could then serve as a black-box feature extractor, to perform an attack similar to the one described in [15].

2.2 Unsupervised translation for embedding matching

The scope of this paper is to find the labels of stolen biometric embeddings by matching their distribution to the one of labelled embeddings. Unsupervised machine translation aims at achieving a bilingual translation by matching word embeddings from a language with word embeddings of another, without knowing the corresponding labels. The closeness of both problems leads us to explore unsupervised translation literature.

For unsupervised machine translation, the success of the algorithm is highly dependent on the initialisation [10] [9]. However, most of unsupervised translation methods either use a few labelled examples or the first thousands most frequent words in each language. The initialisation strategies based on that are not suitable for our problem. Our problem involves a lower number of classes

but with multiple samples for each class. For that reason, most initialisation approaches discussed hereafter cannot be directly implemented.

Grave et al. [9] propose a method to match high dimensional word embeddings from two different lexicons, using Procrustes Analysis [8] and Wasserstein distance [22] with a stochastic optimisation of a rotation matrix. They achieve state of the art performance using the 2000 most frequent words of each language to initialise the matrix. The translation of this initialisation to our task is not directly applicable as we do not have information about the most frequent digits. However, we keep the idea of using Procrustes analysis to find an optimal rotation for a given combination.

Still for unsupervised machine translation, Hoshen & Wolf [10] use Principal Component Analysis to efficiently initialise their algorithm with the 5000 most frequent words of each language. Their algorithm computes the optimal rotation from a given permutation matrix, and finds the optimal permutation matrix according to that rotation. Thanks to the high number of classes (5000) compared to the low number of dimensions (50 after PCA), switching two classes in the permutation matrix only induces small variations in the rotation matrix, so their algorithm can do a step-by-step search. However, due to the low number of classes (10 digits) in our problem, we cannot apply the step-by-step search. Indeed, a permutation error would induce a much more important variation of the rotation matrix. However, we propose to initialise with a global PCA on the data to align both spaces and reduce the number of dimensions before further computing.

Zhou et al. [24] match word embeddings by modeling each one as its own gaussian distribution and fitting Gaussian Mixture Models [17] to each set of words. The transfer function is then trained by minimizing the distance between GMMs. Note that they also use a few identical words in both languages to add a weak similarity constraint to their search. As we want to find the transformation function without any example, we cannot apply that exact method. However we keep the idea of modeling the statistic distribution of embeddings with GMMs and the concept of normalising flow [18] to map the transfer function between the unlabelled embeddings space and a known space.

3 Proposed attack scenario

Here we want to label a set U of stolen unlabelled embeddings. Those embeddings have supposedly been produced on a touchscreen biometric system [21], for authentication by handwritten digits, as illustrated in figure 1.

The U set is composed of N unlabelled embeddings of dimension D, resulting from the penultimate layer of a LSTM classifier designed to process 2D stroke sequences taken from handwritten digits from 0 to 9. For the purpose of this paper, the number N of embeddings depends on the size of the considered dataset.

We suppose an attacker able to find or provide its own data. This data can be used to produce a labelled set of statistically comparable embeddings. To simulate that scenario, we train a second LSTM classifier with a disjoint set of

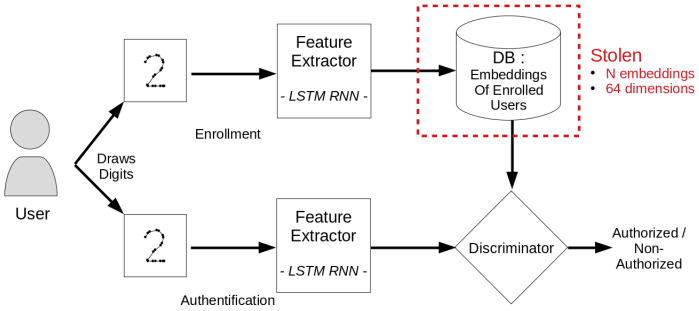

Fig. 1. Illustration of the stolen embeddings, in a touchscreen biometric system for handwritten digits.

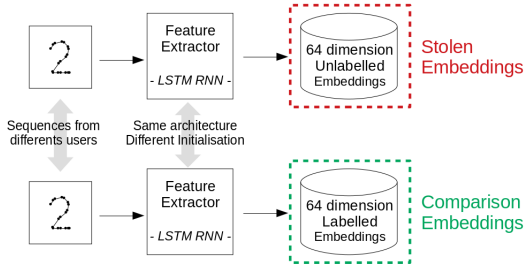

Fig. 2. Illustration of the differences between the stolen, unlabelled embeddings and the comparison, labelled embeddings.

2D sequences, taken from different users, as illustrated in figure 2. Those labelled embeddings will be referred to as the L set. To exploit the stolen embeddings, we propose a method to transfer them into a known, labelled space, namely the output space of the second LSTM classifier. The optimal transformation between the two spaces (i.e. the permutation matrix between unlabelled classes of U and labelled classes of L) can be used to label the stolen embeddings. Those notions of permutation and optimal transformation between two spaces are linked in most of unsupervised translation works [9] [10] [6], and we manipulate both in our proposed method.

The method we propose follows 4 steps, as illustrated in figure 3 :

1. Cluster the embeddings of U in 10 clusters, expecting each cluster to correspond to a class (i.e. a digit value).
2. Apply a global Principal Component Analysis to both sets U and L, projecting embeddings on 10 dimensions;
3. Find the most likely candidate permutation between each cluster of U and each class of L, using a likelihood score;

4. Fine-tune the reversible transformation associated with each candidate to get the optimal transformation, and identify the labels of U.

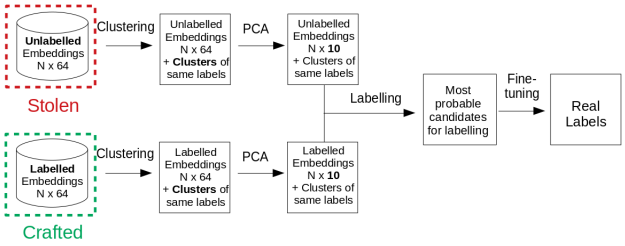

Fig. 3. Illustration of our labelling method.

Our main contributions, related to step 3 (labelling), are detailed in sections 4.2 and 4.3. The steps of clustering and dimension reduction that are necessary to achieve good performance are later described in section 5.3.

4 Labelling

To effectively consider the U set as labelled, we need to find the labels of each one of its clusters. The U set is considered labelled when each of its ten clusters is paired with a class from L corresponding to a digit value. To represent a possible match between clusters of U and labels of L, we use a permutation matrix $P = (p_{ij})_{i,j \in [\![0,9]\!]^2}$, a bi-stochastic matrix composed of 0 and 1, where $p_{ij} = 1$ means the i^{th} cluster is labelled as class j (e.g. digit value j). This section introduces three contributions. First we propose to apply a Procrustes analysis [8] between the cluster centers of U and the class centers of L, in order to approximate the optimal transformation between both sets. Second, we propose a scoring method to evaluate the success of the optimal transformation for a given permutation. Third, we search through the space of all possible permutations to find the best candidates, according to our scoring method. Finally, we fine-tune the best candidates to re-rank them and find the ultimate optimal permutation.

4.1 Optimal Rotation for a given permutation

Search for transfer function as a rotation Mikolov et al. (2013) [16] pointed out that the transformation between the word-embedding spaces of two languages can be well mapped by a linear transformation, so a multiplication matrix and a bias matrix. Each sets being centered, we can ignore the bias matrix, and each embedding being length-normalised, **we consider that the transfer function between spaces of embeddings is a rotation**, which will be verified in section 6.

Procrustes Analysis Considering the transfer function is a rotation, it can be found by a Procrustes analysis that compute a linear transformation between two sets of matched points $U \in \mathbb{R}^{N \times D}$ and $L \in \mathbb{R}^{N \times D}$. in case the match between the two sets is known (i.e., which point of U corresponds to which point of L), the linear transformation can be simply recovered by solving the least square problem:

$$\min_{W \in \mathbb{R}^{D \times D}} \|UW - L\|_2^2 \tag{1}$$

Here we use the 10 centers of the unlabelled clusters as C_U, the 10 centers of the labelled classes as C_L and the match is given by the permutation matrix P. As in Grave et al. (2018) [9], we compute for a given permutation matrix P the solution to the equation 2.

$$\min_{W \in \mathbb{R}^{D \times D}} \|C_U W - PC_L\|_2^2 \tag{2}$$

Procrustes analysis presents a simple solution to that problem. Let the square matrix $M \in \mathbb{R}^{D \times D}$ be :

$$M = C_U^t.PC_L \tag{3}$$

M can be decomposed in singular values, as :

$$M = X \times \Sigma \times Y^* \backslash (X, Y^*) \in (\mathbb{R}^{D \times D})^2 \tag{4}$$

Then the W rotation matrix solution to the equation 2 is defined as :

$$W = X.Y^* \tag{5}$$

Evaluation of a given rotation To select the most probable permutation between the two sets, we have to find a reliable heuristic that evaluates its corresponding rotation, without knowing the labels of one of the sets. We assume that the statistical distribution of each cluster of embeddings is different enough to distinguish it from the others, and thus find its label.

Modeling of the embeddings distribution with Gaussian Mixture Models The statistical distribution of embedding from each set is approximated by a multivariate Gaussian Mixture Model (GMM). The number of components in the GMM is chosen via Bayesian Information Criterion [1]. We do not impose priors, means or co-variances to the models, and use full co-variances matrices.

Global Log-likelihood scoring To measure the distance between a set of embeddings and a GMM, we propose to use the global log-likelihood.

Let the GMM of the labelled set L be $GMM_L = \{(p_i, \mu_i, \Sigma_i) \in (]0, 1] \times \mathbb{R}^D \times \mathbb{R}^{D \times D}) \backslash i \in [\![1, K]\!]\}$, p_i, μ_i and Σ_i being respectively the prior, mean and co-variance of the i^{th} gaussian, with $\sum_{i=1}^{K} p_i = 1$. Let $U = \{u \in \mathbb{R}^D\}$ be the set of unlabelled embeddings, and $W \in \mathbb{R}^{D \times D}$ the given rotation matrix, then $U_W = \{u_W = W.u \backslash u \in U\}$ is the set of projected unlabelled embeddings.

The log-likelihood between a projected embedding u_W and a Gaussian i is :

$$\log \mathcal{N}(u_W|\mu_i, \Sigma_i) = -\frac{1}{2}(K \log 2\pi + \log|\Sigma_i| + (u_W - \mu_i)^T \Sigma_i^{-1}(u_W - \mu_i)) \quad (6)$$

The log-likelihood between a projected embedding u_W and the model GMM_L is then defined as the log of the average of the likelihood with each Gaussian, weighted by the priors $P = \{p_i \in \mathbb{R}\}$:

$$\log \mathcal{N}_{GMM_L}(u_W) = \log \sum_{i=1}^{K} p_i \mathcal{N}(u_W|\mu_i, \Sigma_i) = \log \sum_{i=1}^{K} \exp(\log(p_i) + \log \mathcal{N}(u_W|\mu_i, \Sigma_i)) \quad (7)$$

Finally, the global log-likelihood score of the set X_W is set as minus the average of the individual log-likelihood scores :

$$Score(U_W, GMM_L) = \frac{-1}{Card(U_W)} \sum_{u_W \in U_W} \log \mathcal{N}_{GMM_L}(u_W) \quad (8)$$

Here the $Score(U_W, GMM_L)$ function is defined as the likelihood score between the GMM of a set of embeddings L and a set of embeddings U projected by a W rotation matrix. If the transformation W is confirmed to be a rotation, then W is invertible and its inverse is W^t. Thus, we can define this score for the reversed rotation, between embeddings L projected by a rotation W^t and a GMM fitted to a set U : $Score(L_{W^t}, GMM_U)$.

We propose to take the maximum of the two options, evaluating in a single score the likelihood of a transformation and its reverse :

$$Score(U, L, W) = max(Score(U_W, GMM_L), Score(L_{W^t}, GMM_U)) \quad (9)$$

A higher score means a better matching between sets, so by taking the maximum we use the best of both comparisons. For the rest of the article, we used the opposite global log-likelihood score ($-Score(U, L, W)$) as the function to minimize.

4.2 Genetic Search

For any given permutation, we can compute the associated optimal rotation and evaluate its ability to statistically align both datasets. To find the candidates that minimize the score described above, we explore the space of the possible permutations $P \in [\![0, 1]\!]^{10\times10}$. To find the global best rotation, we need to try all possibles $10! = 3628800$ permutations. To limit the number of tested permutations, we choose to use a genetic algorithm [4] to find the fittest permutations. The genetic algorithm considers each permutation as a chromosome, and gets the best candidates through merging and mutations without scoring every possible permutation. We propose to represent chromosomes as ordered sequences of 10 digits instead of matrices of zeros and ones :

$$C = \{c_i \in [\![0, 9]\!] \setminus i \in [\![0, 9]\!]\} \quad (10)$$

Each element from a chromosome represents the link between a cluster of unlabelled data and a labelled class. c_i being value of the i_{th} element means that the unlabelled cluster c_i is linked to the labelled class i (each cluster is linked to a unique other class).

$$\forall i, j \in [\![0, 9]\!]^2, i \neq j \Leftrightarrow c_i \neq c_j \tag{11}$$

4.3 Fine-tuning

Rotations are approximated using the center of each cluster and thus might not be as precise as if every embedding was used. As a result, the best candidate permutation found by the genetic search might not always be the genuine one. To refine and re-rank the k-best candidate permutations, we propose a stochastic optimisation.The candidate with the best score after fine-tuning is expected to give the genuine labels.

Our fine-tuning is inspired by [24], which uses gradient descent to find the optimal rotation matrix a comparable statistical alignment problem, using two weak constraints during training (orthogonality and unitary determinant). For each k-best permutation candidate, we fine-tune W with the Adam stochastic optimisation method [12] to minimize the global log-likelihood score.

Losses To fine-tune each matrix to minimize the global log-likelihood score while keeping their rotation properties, we combine three loss functions :

1. Loss 1: The global log-likelihood score $-Score(U, L, W)$
2. Loss 2: The absolute log of the determinant of W : $|\log(\det W)|$
3. Loss 3: The difference $u_i - (W^t \times W \times u_i)$

The first loss fits the matrix W to the optimum transformation between the two sets of embeddings. The second targets a determinant of 1, and the third insures that W is orthogonal. The last two guarantee that W stays a rotation matrix. The global loss is a non-pondered sum of the three losses.

After a few dozens of epochs, the losses are stabilized, and we get W^*, the fine-tuned version of W. Once each instance of W^* scored with global log likely-hood score, the one with the minimum score is the best candidate, the permutation associated giving the searched labels.

5 Data and Preprocessing

5.1 Data

The data is taken from two different datasets described in: Tolosana et al. (2018) [21] and Tolosana et al (2019) [20], both produced by the University of Madrid, containing data from respectively 217 and 93 users. The first set contain 8460 stroke sequences of variable length (mean=31.9, std = 13.1, max = 164), in 2 dimensions, representing digits drawings from 0 to 9. The second contain 7430 sequences of variable length (mean=33.9, std = 13.2, max = 125).

In total, it results in 16350 sequences, with an equal proportion (a tenth) of each digit. Those sequences are divided into 4 sets of equal digit proportion, each set containing the sequences of a randomized quarter of the total number of users. Those sets will then be referred as : **Train U**, **Test U**, **Train L** and **Test L**.

Both *Train* sets are used to train the classifiers and the *Test* sets to evaluate their performances. The embeddings used for unsupervised matching in the rest of the paper are produced by passing the sequences from the *Test* sets through the classifiers.

In order to multiply the experiments with the same original data, we randomly split in 4 parts the set of users in 100 different ways, to get 100 different simulations. Sets of users are always composed of 77 to 78 users, and each set contains 3450 to 4560 sequences (mean = 4079.35) with each digit having the same number of examples.

5.2 Architecture of the networks

The architecture of the feature extractor, assumed to be known by the attacker, is a Long Short-Term Memory (LSTM) RNN. We train thus two classifiers with an input in 2 dimensions and a hidden state vector of $D = 64$ dimensions. The last hidden-layer is passed through a fully connected layer of dimension 10 and an a softmax function to predict the digit value.

Both networks are trained with both **Train** sets using Cross Entropy Loss as the objective function to predict the digit associated with each sequence. The training stops when each network has a precision of 96% on its **Test** set. Both networks have the same architecture but a different, random initialisation of their parameters. 64-dimension embeddings are then extracted from the penultimate layer of each network for all digits from the respective **Test** set.

The set *Test L* is processed by the first network to produce a set of *Labelled* embeddings referred to as *L set*, while the set *Test U* is processed through the second network to produce a set of *Unlabelled* embeddings referred to as *U set*. All produced sets are composed of 3450 to 4560 embeddings (average of 4079.35) in 64 dimensions, with each class having the same number of examples.

5.3 Preprocessing

Normalisation For each set, all embeddings are length-normalised, centered as in *Grave et al. (2018)* [9] and length-normalised again.

Clustering The networks being trained to be classifiers, we assume they project the original sequences in a vector space were borders can be drawn between same-label classes. Thus, we should be able to split the data in class-related clusters. To assert this point, we propose to use the K-means clustering algorithm to split the unlabelled set in ten clusters. For further purposes, the groups formed by same-label embeddings of L set will be referred as classes, as opposed to the clusters of U.

Principal Component Analysis We propose to initiate our method with principal component analysis, as in Hoshen & Wolf [10]. We propose $D = 10$, so the number of dimensions do not exceed the number of distinct matched points used to compute the Procrutes rotation matrix. For the rest of the paper, we work with the 10-dimension PCA-reduced embeddings for both sets. We compute the means of each cluster of U and each class of L after the PCA, so we end up with 2×10 average embeddings representing the centers of the clusters and classes.

6 Experiments

To provide reproducible and precise experiments, each following experiment is carried out with the same dataset split, and results are presented for that example dataset split. Secondly, we carry out the same experiment over the 100 pairs of sets produced in section 5.

6.1 Clustering

We split the unlabelled embeddings in 10 clusters with the K-means algorithm. The result of this clustering is in the table 1. We measured the cohesion of the clusters relative to the original labels of the embeddings. We found a cohesion

Table 1. Embeddings of each label and their associated cluster number after K-means clustering

Label \ Cluster	0	1	2	3	4	5	6	7	8	9	Σ
0	0	12	4	4	9	1	4	3	**363**	0	400
1	0	1	**395**	4	0	0	0	0	0	0	400
2	**394**	0	2	1	0	3	0	0	0	0	400
3	0	0	0	0	1	1	0	**392**	0	6	400
4	1	0	7	**375**	0	4	13	0	0	0	400
5	0	0	0	1	14	0	0	4	1	**380**	400
6	0	**382**	0	0	0	1	0	0	14	3	400
7	6	0	4	0	2	**380**	6	0	1	1	400
8	0	4	7	0	**350**	1	2	2	30	4	400
9	0	6	1	6	2	0	**381**	0	3	1	400
Maximum :	394	382	395	375	350	380	381	392	363	380	3792

of $\frac{3792}{4000} = 0.948$ after the clustering, meaning 94,80% of the embeddings can be grouped together by a clustering.

For a better precision, we measure the clustering cohesion for every set of embeddings over the 100 different split of data. The global cohesion was between 92.26% and 96.04% (mean = 94.45%). This is an acceptable cohesion, knowing that the originally trained classifier got a 96% of accuracy over the test set.

This confirms that our unlabelled embeddings can be split in label-wise clusters. This also means that when we have 100% accuracy on the clusters labelling, an average of 94.45% of the individual embeddings will be correctly labelled.

6.2 Principal Component Analysis

Each set of embeddings from the 100 splits is projected in $P = 10$ dimensions using PCA. The total ratio of explained variance is between 80,4% and 88,2% (mean = 84,9%).

6.3 Rotation

To prove that the relation between the two sets of embeddings can be well mapped as a rotation, we suppose the labels of both sets to be known, just for the purpose of this experiment. We apply Procrustes analysis on the centers of the label-wise clusters of both sets to compute the optimal rotation W between the two sets. Then we project every embedding of the unlabelled set with W. For each projected embedding, we measure the nearest labelled cluster center, the projection being considered as successful if that nearest cluster has the same label as the original embedding. The table 2 presents the results of that association. This experimentation shows 94.31% accuracy, meaning 4,103 out of

Table 2. Embeddings projected with Optimal rotation: Label by Nearest Cluster

Label \ Nearest Cluster	0	1	2	3	4	5	6	7	8	9
0	**402**	0	1	0	13	1	14	0	1	3
1	0	**412**	1	0	19	0	2	1	0	0
2	0	0	**429**	0	1	1	1	3	0	0
3	1	0	0	**417**	0	3	2	3	7	2
4	3	3	1	0	**402**	0	0	4	5	17
5	1	0	0	4	3	**401**	10	9	7	0
6	7	0	0	3	0	4	**420**	0	0	1
7	1	2	2	0	4	1	0	**425**	0	0
8	14	6	3	2	1	15	2	2	**386**	4
9	0	1	0	1	21	1	1	1	0	**409**

4,350 embeddings from the unlabelled set were associated with the right cluster. When reproducing this experiment with every pairs of sets from the 100 splits we observe 92.44% and 95.86% accuracy (mean = 94.50%). From there, we can consider that the transformation between the two spaces can be well approximated as a rotation, as thus confirm that if we get the right permutation, over 92% of the embeddings will be correctly labelled. We are therefore looking for the optimal rotation between the Labelled space and the Unlabelled one. Thus, as said in subsection 4.1, the transfer function between the spaces of both sets of embeddings is considered a rotation.

6.4 Reliability of the global log-likelihood score

To test the reliability of that score function we use every one of the 10! combinations to compute the 10! rotations associated, and measure the score of the rotated set for each of them. The genuine permutation is supposed to obtain the lowest score. We group all permutations by their number of correct matches and plot them on figure 4 with the corresponding scores. The first column is only the expected permutation, and the others are the scores of permutations with 2 to 10 mismatches. The second graphic is a zoom on the lower part of the first one. The figure shows that the genuine permutation only obtains the third lowest

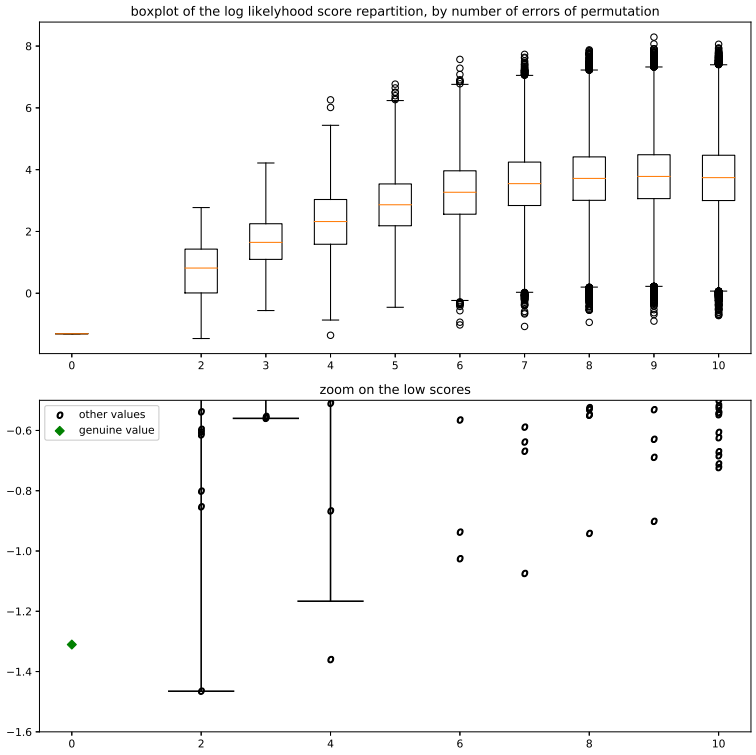

Fig. 4. box plot of the global log likelihood score of every embedding, as a function of the number of errors.

score. The score depends on the GMM and since the GMM initialisation is random, the score is slightly different each time. After running this experiment 10 times on this particular dataset split to smooth the variations due to the GMM initialisation, the genuine permutation is always observed between the 1^{st} and the 6^{th} rank. Over all the candidate permutations, the global log-likelihood score gives the genuine permutation one of the lowest scores. Therefore, we have to not

only consider the candidate with the lowest score, but a list of the k candidates with the lowest scores, to be sure to have the genuine candidate among them. That is why we have to fine-tune the rotations in order to re-rank the k-best permutations.

6.5 Genetic Search

We are searching through all possible permutations using chromosomes consisting of ordered sequences of 10 naturals numbers (see equation 10). Each element represents the link between a cluster of unlabelled data and a digit class.

A Initialisation The search is initialised with 150 Chromosomes and each of them is evaluated by computing its rotation matrix and their score as explained in subsection. 4.
B Selection The 20 chromosomes with the lowest score are selected.
C Merging Two chromosomes from the 20 selected are randomly taken, and merged to create a new one. The common elements stay the same while the elements that are different are randomly selected from one or the other. If the new chromosome has not been seen yet, it is added to the list, and an other merge is done until reaching the 100 more chromosomes.
D Mutation One of the 20 selected chromosomes is randomly selected. 2 to 10 elements of that chromosome are randomly selected and then rotated to obtain a new, mutated chromosome. If the new chromosome have not been seen yet, it is added to the list, and an other mutation is done until reaching the 50 more chromosomes.
E End of the Main Loop Finally, after getting a list of 170 chromosomes, the score of each is evaluated, and the algorithm get back to step B : the selection. It loops over until the 20 selected chromosomes stabilise and stay the same 100 loops in a row. The output is the list of those $k = 20$ selected chromosomes

We use the genetic search to find the 20 best permutations according to the global log-likelihood score. An experiment takes around 55 loop each to be completed, so around 10^4 permutations are evaluated over the 3.6×10^6 possible ones. At the end of the search, the genuine combination is part of the $k = 20$ best candidates. An example of the score for each one of the 20 best candidates is presented in table 3.

For the given example, the genuine candidate, [**0 1 2 3 4 5 6 7 8 9**], is not the best one selected. It is ranked third. Thus fine tuning is required, in subsection 6.6. To present more consistent results, we run this genetic search over the 100 splits of data and registered the rank of the genuine permutation for each pair of sets. The genuine candidate is ranked first 62 times out of 100, note that it is also ranked 20^{th} once. Figure 5 presents an histogram of the ranks obtained for each dataset split.

Table 3. Example of a table of the scores for the first 20 candidates after a genetic search. *genuine candidate.

Rank	Candidates										Score
1	**0**	4	**2**	**3**	1	**5**	**6**	**7**	**8**	**9**	**-1.465**
2	8	**1**	**2**	**3**	**4**	6	5	**7**	0	**9**	-1.360
3*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	-1.310
4	8	0	**2**	**3**	6	7	5	1	4	**9**	-1.074
5	8	7	**2**	**3**	**4**	6	5	1	0	**9**	-1.025
6	8	**1**	**2**	**3**	7	6	5	4	0	**9**	-0.937
7	7	**1**	**2**	**3**	**4**	6	5	0	**8**	**9**	-0.867
8	**0**	**1**	**2**	**3**	**4**	6	5	**7**	**8**	**9**	-0.853
9	**0**	**1**	**2**	**3**	**4**	8	5	**7**	6	**9**	-0.559
10	**0**	2	4	**3**	1	**5**	**6**	**7**	**8**	**9**	-0.553
11	**0**	7	**2**	**3**	**4**	6	5	1	**8**	**9**	-0.509
12	**0**	2	7	**3**	1	**5**	**6**	4	**8**	**9**	-0.461
13	**0**	7	**2**	**3**	**4**	**5**	**6**	1	**8**	**9**	-0.457
14	6	0	**2**	**3**	8	7	5	1	4	**9**	-0.403
15	7	6	8	9	5	1	4	0	2	3	-0.386
16	**0**	5	**2**	**3**	6	7	1	8	4	**9**	-0.364
17	**0**	**1**	**2**	**3**	**4**	**5**	8	**7**	6	**9**	-0.361
18	6	0	**2**	**3**	**4**	7	5	1	**8**	**9**	-0.355
19	**0**	4	7	**3**	1	**5**	**6**	2	**8**	**9**	-0.331
20	8	**1**	**2**	**3**	**4**	0	5	**7**	6	**9**	-0.305

6.6 Fine Tuning

The genetic search gave us a quick reliable way to find a set of good candidates, we now target to find the absolute best permutation by fine tuning each candidate, using the density matching algorithm 4.3.

For each candidate permutation, the fine-tuning rotation matrix is initialised with the one previously computed using Procrustes analysis on the centers. Each candidate rotation matrix is fine-tuned for 200 epochs with the configuration detailed in subsection 6.6. The global log-likelihood of the statistical alignment is computed again afterwards. Table 4 corresponds to the same experiment as table 3 but after applying fine-tuning.

The genuine candidate effectively got the best score, while the previous best candidate moved back to rank 5. For this example, the permutation that matches the cluster of i with the label i ($\forall i \in [\![0, 9]\!]$) is selected and the clusters are correctly labelled.

To present more consistent results, we run the proposed fine-tuning over the 100 splits of data and register the rank of the genuine solution for each pair of sets. **The genuine solution is ranked first every time.** Before fine-tuning, the average score on the 20 selected candidates is -1.910 while the average score of the genuine candidate is -2.325, for an average rank of 3.27. After fine-tuning, the average score on the 20 selected candidates became -4.453 while the average score of the genuine candidate went down to -5.042, for an average rank of 1.00

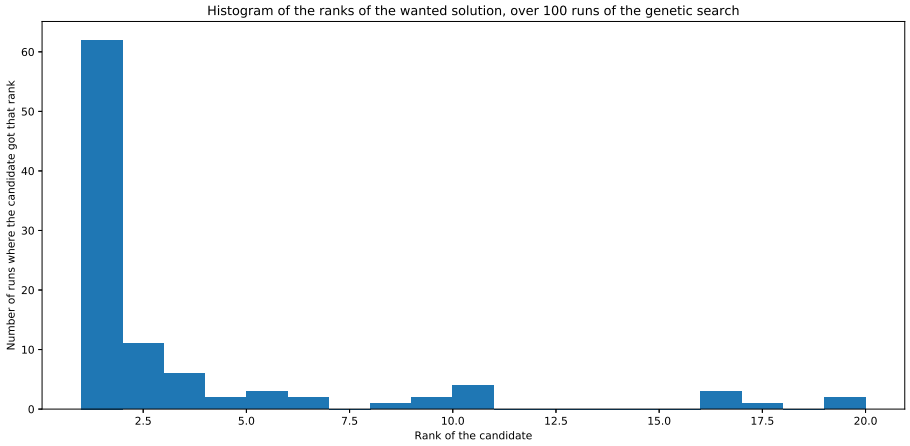

Fig. 5. Histogram of the ranks of the wanted solution after a genetic search.

7 Conclusion - Future work

This paper presents a statistical alignment method for high dimensional unlabelled embeddings of handwritten digits, in the event of a theft. Our method is inspired by unsupervised bilingual translation and reconstruction of biometric templates literature. We aim to find the digit value (label) of each embedding. Provided a set of unlabelled embeddings produced by a LSTM RNN, we train a comparison RNN with the same architecture to produce hand-crafted comparison labelled embeddings.

We proposed to label the stolen embeddings by matching their clusters to the label-wise classes of the comparison embeddings. The labelling consists in a genetic search through all possible permutations between clusters and classes to find the 20 candidates with the lowest global log-likelihood score. Each of those candidates is fine-tuned and the fine-tuned candidate with the lowest score is expected to represent the genuine permutation.

We have applied this method on 100 different distinct splits of the original dataset. Our experiment showed that after the genetic search, the genuine candidate got an average rank of 3,27, and got ranked first every time after fine tuning. Thus the proposed method proved to be a reliable way to recover most labels of the stolen handwritten digits embeddings (without further exploitation we report a 94.45% average accuracy over the labeling of the individual embeddings due to the clustering cohesion).

Future work will be dedicated to the relaxation of the constraints (higher number of classes, unknown network architecture, other biometrics) and the reconstruction of the signals. Overall, our work highlight the importance of personal data protection, especially embeddings from biometric systems, and open perspectives for further threat models analysis and associated defenses.

Table 4. Table of the scores for the first 20 candidates after fine-tuning. *genuine candidate

Updated Rank	Previous Rank	Candidates										Updated Score	Previous Score
1*	3	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**-5.914**	-1.310
2	9	**0**	**1**	**2**	**3**	**4**	8	5	**7**	6	**9**	-5.827	-0.559
3	6	8	**1**	**2**	**3**	7	6	5	4	0	**9**	-5.770	-0.937
4	2	8	**1**	**2**	**3**	**4**	6	5	**7**	0	**9**	-5.603	-1.360
5	1	**0**	4	**2**	**3**	1	**5**	**6**	**7**	**8**	**9**	-5.602	**-1.465**
6	10	**0**	2	4	**3**	1	**5**	**6**	**7**	**8**	**9**	-5.420	-0.553
7	13	**0**	7	**2**	**3**	**4**	**5**	**6**	1	**8**	**9**	-5.335	-0.457
8	20	8	**1**	**2**	**3**	**4**	0	5	**7**	6	**9**	-5.334	-0.305
9	17	**0**	**1**	**2**	**3**	**4**	**5**	8	**7**	6	**9**	-5.322	-0.361
10	7	7	**1**	**2**	**3**	**4**	6	5	0	**8**	**9**	-5.269	-0.867
11	5	8	7	**2**	**3**	**4**	6	5	1	0	**9**	-5.219	-1.025
12	12	**0**	2	7	**3**	1	**5**	**6**	4	**8**	**9**	-5.043	-0.461
13	19	**0**	4	7	**3**	1	**5**	**6**	2	**8**	**9**	-5.026	-0.331
14	15	7	6	8	9	5	1	4	0	2	3	-4.999	-0.386
15	8	**0**	**1**	**2**	**3**	**4**	6	5	**7**	**8**	**9**	-4.985	-0.853
16	4	8	0	**2**	**3**	6	7	5	1	4	**9**	-4.775	-1.074
17	14	6	0	**2**	**3**	8	7	5	1	4	**9**	-4.731	-0.403
18	18	6	0	**2**	**3**	**4**	7	5	1	**8**	**9**	-4.727	-0.355
19	16	**0**	5	**2**	**3**	6	7	1	8	4	**9**	-4.687	-0.364
20	11	**0**	7	**2**	**3**	**4**	6	5	1	**8**	**9**	-4.622	-0.509

References

1. H Akaike. A new look at the statistical identification model. *IEEE Transactions on Automatic Control*, 19:716, 1974.
2. Raffaele Cappelli, Dario Maio, Alessandra Lumini, and Davide Maltoni. Fingerprint image reconstruction from standard templates. *IEEE transactions on pattern analysis and machine intelligence*, 29(9):1489–1503, 2007.
3. Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.
4. Stephanie Forrest. Genetic algorithms: principles of natural selection applied to computation. *Science*, 261(5123):872–878, 1993.
5. Javier Galbally, Arun Ross, Marta Gomez-Barrero, Julian Fierrez, and Javier Ortega-Garcia. Iris image reconstruction from binary templates: An efficient probabilistic approach based on genetic algorithms. *Computer Vision and Image Understanding*, 117(10):1512–1525, 2013.
6. Goran Glavas, Robert Litschko, Sebastian Ruder, and Ivan Vulic. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. *arXiv preprint arXiv:1902.00508*, 2019.
7. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

8. John C Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.
9. Edouard Grave, Armand Joulin, and Quentin Berthet. Unsupervised alignment of embeddings with wasserstein procrustes. *arXiv preprint arXiv:1805.11222*, 2018.
10. Yedid Hoshen and Lior Wolf. Non-adversarial unsupervised word translation. *arXiv preprint arXiv:1801.06126*, 2018.
11. Anil K Jain, Karthik Nandakumar, and Abhishek Nagar. Biometric template security. *EURASIP Journal on advances in signal processing*, 2008:1–17, 2008.
12. Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
13. Gaël Le Lan and Vincent Frey. Securing smartphone handwritten pin codes with recurrent neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2612–2616. IEEE, 2019.
14. Lily Lee and W Eric L Grimson. Gait analysis for recognition and classification. In *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 155–162. IEEE, 2002.
15. Guangcan Mai, Kai Cao, Pong C Yuen, and Anil K Jain. On the reconstruction of face images from deep face templates. *IEEE transactions on pattern analysis and machine intelligence*, 41(5):1188–1202, 2018.
16. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
17. Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741, 2009.
18. Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.
19. David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333. IEEE, 2018.
20. Ruben Tolosana, Ruben Vera-Rodriguez, and Julian Fierrez. Biotouchpass: Handwritten passwords for touchscreen biometrics. *IEEE Transactions on Mobile Computing*, 2019.
21. Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, and Javier Ortega-Garcia. Incorporating touch biometrics to mobile one-time passwords: Exploration of digits. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.
22. SS Vallender. Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, 18(4):784–786, 1974.
23. Wencheng Yang, Song Wang, Jiankun Hu, Guanglou Zheng, and Craig Valli. Security and accuracy of fingerprint-based biometrics: A review. *Symmetry*, 11(2):141, 2019.
24. Chunting Zhou, Xuezhe Ma, Di Wang, and Graham Neubig. Density matching for bilingual word embedding. *arXiv preprint arXiv:1904.02343*, 2019.