Embedding is not Cipher: Understanding the risk of embedding leakages

Zhe Zhou, Mingtian Tan
Fudan University
{zhouzhe, 18210240176}@fudan.edu.cn

ABSTRACT
Machine Learning (ML) already has been integrated into all kinds of systems, helping developers to solve problems with even higher accuracy than human beings. However, when integrating ML models into a system, developers may accidentally take not enough care of the outputs of ML models, mainly because of their unfamiliarity with ML and AI, resulting in severe consequences like hurting data owners’ privacy. In this work, we focus on understanding the risks of abusing embeddings of ML models, an important and popular way of using ML. To show the consequence, we reveal several kinds of channels in which embeddings are accidentally leaked. As our study shows, a face verification system deployed by a government organization leaking only distance to authentic users allows an attacker to exactly recover the embedding of the verifier’s pre-installed photo. Further, as we discovered, with the leaked embedding, attackers can easily recover the input photo with negligible quality losses, indicating devastating consequences to users’ privacy. This is achieved with our devised GAN-like structure model, which showed 93.65% success rate on popular face embedding model under black box assumption.

KEYWORDS
AI Security; Privacy; Face Embedding;

1 INTRODUCTION
Machine learning (ML) gained a great success because of its high accuracy and convenience to implement. ML outperforms both human beings and traditional programs for a lot of problems, like face recognition, board game etc., which convinces people to confidently integrate ML models to their systems. Apart from high accuracy, ML is easy to implement. All developers need to do is to collect and label a bunch of data. Then the ML model can automatically learn to handle future inputs though training.

When developing a system with ML, developers usually outsource all ML model related workload to an independent and professional ML team and employ ML models like using a third party library. The outsourcing style is chosen mainly because ordinary developers are lack of data science background knowledge and are hesitated to learn methods to handle ML model.

The ML outsourcing accelerates product development process and requires developers only little ML background. Usually, they need only to comprehend the format and meanings of the model input and output. In this way, both teams can focus on their expertise without getting too much coupling. This style turned to be a prevalent way for integrating ML to systems.

However, ML outsourcing introduces potential threats because of developers’ abuse of unfamiliarity with ML. Different from employing a third party library, ML models and their output values are totally foreign to developers. For the third party library case, the return value and functionality of each interface can be easily understood by developers and usually are documented in detail. By contrast, ML models, though sometimes also come with manuals and documentations, are not fully understood by developers in terms of security implications. Actually, even ML model designers themselves do not have enough understandings to the security implications of their works.

Our work takes face embeddings as an example to understand the risk behind such a development process. Embeddings are a widely used principle in ML to handle a wide range of problems. In ML, embeddings are used to map complex data into a unified space (usually a fixed dimension vector) with specific properties remained, so the data can be further more easily processed in the simpler space. Many complex problems firstly employ embeddings to allow later ML or non-ML operations on the unified space. For example, face search employs face embedding to map a portrait photo to a face embedding, so the embedding can be used to be compared with other vectors in database by using a simple and well-known distance like L2 distance or cosine distance. Face embeddings are chosen as our focus because 1) Embeddings are often the interface between ML and non-ML. Security issues often root from the transition between entities. 2) Embedding is the most ML concept that developers must know before they can operate the model. Complex ML details are all hidden to developers but embedding is a must-know concept before they can further use embeddings to achieve their required functionality. However, according to our study, embedding is still too ML and developers may not fully comprehend the implications behind. 3)Embeddings are so widely used in solving problems like CV, NLP etc.

To understand the risk, we firstly studied how developers may leak embeddings. Developers are assumed to operate embeddings provided by ML models, like calculating distance, storing or transmitting these embeddings. We investigate if there are possibilities for attackers to infer or directly get embeddings during these operations. Then we study the consequence brought by leaked embeddings. Embedding implies a mapping from a complex space to simple space for easier operation. We consider if there are possibilities for attackers to reverse the mapping i.e. get a mapping from the simple space back to the original complex space.

Our first finding is that developers are prone to leak embeddings accidentally in many ways mainly because they regard embeddings as insensitive and do not comprehend its ML implications. The most threaten way we found is leaking through distance calculation, i.e. developers calculate and leak the distance from a user fed input to a sensitive embedding. We discussed with several developers who had experience in using embeddings but none of them thought it
We mainly have two adversary models presented in the work: black box and white box. The second finding is that actually embeddings are as equal sensitive as original input, because with our designed model, the original input generating an embedding can be recovered if only the embedding is acquired. To achieve this, we designed a GAN-like model that learns the mapping between the complex input space and the embedding space. Moreover, the model tries to reverse the mapping and generate a valid and corresponded input against a given embedding. The evaluation results show that the recovered input images come with nearly no quality loss.

The two findings together imply high risks of embedding leakages. The recovery model abridged the embedding and the original input, while the first finding linked the embedding with non-ML leakages. Being combined together, the two findings imply that attackers can recover original input if only they acquire some leakages that are results from developers’ carelessness, like the distances. The original input definitely is highly sensitive and must be well-protected against any leakage. Ultimately, the embedding leakage implies high risks.

To avoid any further embedding leakages, we suggest that ML library developers must clearly state in the documents related to getting embeddings that the embedding is as equal sensitive as original image input and any result calculated with embeddings should not be freely exposed to users or third parties. Beside, we suggest ML researchers design embedding models that cannot be easily reversed.

We summarize our contributions as follow:

- We identified that developers may accidentally leak embeddings through various channels, because from developers’ perspective, embeddings are not that sensitive.
- We discovered that leaking distance to embeddings is equal to leaking embeddings.
- We discovered that leaking embeddings is nearly equal to leaking the original input for face recognition models.
- We designed a GAN based recovery model to map face embeddings to portrait photos. With the model we showed that popular embeddings can all be recovered, for 4 well-known face embedding models.

2 ADVERSARY MODEL

We mainly have two adversary models presented in the work: black box and white box.

2.1 Goal of Adversaries

Adversaries may target specific victims or just hang out to hunt random victims. Adversaries are assumed to be interested in their victims’ privacy, especially appearance. Specifically, adversaries want the profile photos of their victims.

2.2 Common Assumptions

Embedding. We assume that adversaries can always get the embedding of any image they provide.

This assumption usually holds because we assume ML models are included in a manner of public service that whoever can freely use it. For example, an app has a face embedding model inside, so an attacker can reverse engineer the app to get the model and use the model to generate embeddings for any images she wants. Or, if an app uses an online ML model, an attacker can reverse engineer the protocol first and then inject an image to the model to get the embedding.

Acquiring leakages. We assume adversaries can directly or indirectly acquire some leaked embeddings of their target. Developers leak embeddings in many ways, which will be introduced in later sections. These leaking channels can be categorized into direct and indirect channels.

- Adversaries may leak embeddings without being aware of that. Because of developers’ shortage of ML background, developers may inadvertently leak some calculation immediate results related to embeddings, which may be highly sensitive as we will later show. In this case, attackers can reconstruct embeddings with these leakages.
- Adversaries may acquire embeddings directly because the developers transmit, store or manipulate embeddings in an insecure manner, which leaves opportunities to attackers to directly intercept embeddings. For example, we will later show that developers may use HTTP protocol to start service related to ML. In this case, attackers can readily use off-the-shelf sniffing tools to intercept embeddings from network.

Image Database. Adversaries are assumed to own a huge database of human faces, which can be easily achieved today. He can download open data-set for CV directly. A lot of such data-set can be acquired, including LFW and CASIA-WEBFACE.

2.3 Black Box Adversaries

Adversaries need not to know anything about the embedding model, which indicates black box scenario. Adversaries can acquire embeddings of their victims but have no idea what ML model generated these embeddings. They do not know what structure the model uses, how many neurons the model has, not to mention the parameters of the model.

This could happen if an adversary targets online model or if she is not able to reverse engineer the target service or protocol. Today, a lot of ML enterprises provide online ML service to other enterprises or users. In this case, the model has no any public documentation that can direct adversaries to recover victim’s appearance.

2.4 White Box adversaries

Sometimes, an adversary may know the target model in detail, including structure and even parameters. In this scenarios, she is expected to recover images with better quality.

In some cases, ML models are embedded into products, like embedded into an app, in which case adversaries can extract the ML model out and have analysis to the model.

3 EMBEDDING LEAKAGE

In this section, we show that how embeddings can be leaked by careless developers. The first category, i.e. distance based leakage, is because of developers’ unfamiliarity with ML and mathematics
3.1 Leakage through Distance

**Embeddings are geometric points.** As mentioned before, embeddings enable developers to more easily compare a pair of complex data by allowing directly calculating distance (norm of difference) of a pair of embeddings. The calculated distance, by expectation, reflects the similarity of the original complex data pair. All these are achievable because embeddings are regarded as geometric points.

**Distances are leaked.** However, the calculated distance, from developers’ perspective, is not sensitive, according to our observation in a lot of cases. As a result, they sometimes expose the calculated distance in unnecessary places.

An interesting distance leakage example we encountered is a self-service machine that was deployed by a government. This machine authenticates users with their faces before they can touch critical functions. To use this machine, a user need firstly enter his ID number and then stare at the camera that will then take a photo for the user. The photo will be compared with profile photo stored in the government’s database. However, the machine directly displays the similarity between the current user and the profile photo belonging to the claimed user.

**Embeddings can be recovered from distances.** As we identified, if only an attacker acquires enough amount of distances to a sensitive embedding, the embedding can be recovered.

Let us consider the simplest occasion. Assuming that there is an unknown point \( p_0 \) (an embedding with two dimensions) in a 2D plane, if only other two points \( (p_1 \) and \( p_2) \) and their distances \( (d_1 \) and \( d_2) \) to the point \( p_0 \) are known, the coordinates of \( p_0 \) can be worked out, because \( p_0 \) must be located at the intersection of two circles: One centered in \( p_1 \) with radius \( d_1 \), another centered in \( p_2 \) with radius \( d_2 \). Intersection of two circles can be either two points or one point or just no intersection. Because we already know that there is at least one point \( p_0 \) in the intersection, there can be either one point or two points in the intersection and our aim \( p_0 \) is just one of them. As a result, if only \( p_1, p_2, d_1 \) and \( d_2 \) are known, the \( p_0 \) is definitely also known.

Given that a n-D embedding is used by developers, once distances to other n embeddings are known, the embedding is definitely leaked. Embeddings are also points, so if only some distances to an embedding is known, the embedding is known. The only difference between embeddings and 2D points is that embeddings are of higher dimension.

**Solve equation to recover embeddings.** The attacker can solve equations to get the embedding. Every time an attacker knows a distance to the embedding, she knows that the embedding is on a (n-1)-sphere. When she accumulated some different distances, she knows that the embedding is on the intersection of them. From the perspective of algebra, she gets an equation about the embedding for each (n-1)-sphere and the embedding is just the root of the equation system (shown by equation system .1), where \( \bar{x} \) is the unknown embedding, \( \bar{x}_i \) and \( d_i \) being the known embedding and its distance to \( \bar{x} \).

\[
\begin{align*}
||\bar{x} - \bar{x}_1|| &= d_1^2 \\
||\bar{x} - \bar{x}_2|| &= d_2^2 \\
&\vdots \\
||\bar{x} - \bar{x}_n|| &= d_n^2
\end{align*}
\]

Embedding systems usually use either cosine distance or Euclidean (L2) distance. Specifically, for embedding systems that are evaluated with L2 distance, the equation system attackers need to solve turns to equation 2. If equation 2 has and only has few (like one or two) roots like the \( n = 2 \) (circle) case, embeddings can be recovered by solving the equation. If the equation has \( O(poly(n)) \) roots, the embedding still cannot be recovered, because attackers do not know which one is the embedding among these roots.

\[
\bar{x}^\top \cdot \bar{x} + A \cdot \bar{x} + D = 0
\]

where

\[
A = -2 \cdot \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}^\top \\
D = \{\bar{x}_1^\top \cdot \bar{x}_1 - d_1^2, \bar{x}_2^\top \cdot \bar{x}_2 - d_2^2, \ldots, \bar{x}_n^\top \cdot \bar{x}_n - d_n^2\}^\top
\]

Therefore, the number of roots of equation 2 decides if attackers can successfully recover embeddings. Unfortunately, as we will later show, equation 2 has at most 2 different roots, indicating that attackers can readily recover embeddings.

To know how many different possible embeddings attackers will have with \( n \) distances, we firstly check how many roots equation 3 has. Because it is an ordinary linear equation system, it has only one root (shown by equation 4) given that all \( \bar{x}_i \) are linearly independent and \( z \) is a newly introduced constant that is independent of \( \bar{x} \). If the root of equation 3 wants to be also the root of equation 2, it need only also satisfy equation 5, as equation 3 becomes to equation 2 if only \( z \) is replaced by \( \bar{x}^\top \cdot \bar{x} \). However, such \( z \) has only up to two roots, as equation 5 is an ordinary quadratic equation about \( z \) which is a scalar. As a result, \( \bar{x} \) can also have up to two roots, as shown by equation 6.

\[
z + A \cdot \bar{x} + D = 0
\]

\[
\bar{x} = -A^{-1} \cdot (D + z)
\]

\[
z = \bar{x}^\top \cdot \bar{x}
\]

\[
= D^\top BD + z \cdot \bar{1}^\top BD + z \cdot D^\top B \cdot \bar{1} + z^2 \cdot \bar{1}^\top B \cdot \bar{1}
\]

where

\[
B = (A^{-1})^\top \cdot A^{-1} \\
\bar{1} = (1, 1, \ldots, 1)^\top
\]

\[
\bar{x} = -A^{-1} \cdot (D + \frac{-b \pm \sqrt{b^2 - 4ac}}{2a})
\]
where

\[
\begin{align*}
    a &= \mathbf{1}^\top B \cdot \mathbf{1} \\
    b &= \mathbf{1}^\top BD + D\mathbf{1}^\top B - 1 \\
    c &= D\mathbf{1}^\top BD
\end{align*}
\]

Specifically for embedding systems with cosine distance as metric, things are similar. Equation 8 tells an attacker how to calculate the embedding when she has \( n \) different distances, because the embedding must satisfy equation 7. In this case, the norm of the embedding gets lost, but the norm of embedding in cosine embedding systems is not important because two parallel embeddings will be regarded as exactly the same embeddings.

\[
\begin{align*}
    A \cdot \frac{\mathbf{x}}{|\mathbf{x}|} &= D \quad (7) \\
    \frac{\mathbf{x}}{|\mathbf{x}|} &= A^{-1} \cdot D \quad (8)
\end{align*}
\]

where

\[
A = \left\{ \frac{x_1}{|x_1|}, \frac{x_2}{|x_2|}, \ldots, \frac{x_n}{|x_n|} \right\}^T
\]

\[
D = \{1 - d_1, 1 - d_2, 1 - d_n\}^T
\]

Attacker can also employ numerical analysis way to solve equation 2 and equation 7 without knowing any linear algebra background. Scientific computing tools like Matlab and Wolfram Alpha all provide straightforward UI to allow attackers enter the equation and get numerical roots shortly.

In a word, when an attacker accumulated \( n \) distances to an embedding, she can always recover the embedding, no matter if the embedding is in Euclidean system or cosine system.

Recovering anyone’s profile photo embedding. Putting into the self-service machine case, an attacker can print say like 1000 photos of different 1000 people. Every time she wants to recover somebody’s profile photo in government’s database, she needs only go to the self service machine and enter the victim’s ID number, then present the 1000 photos one by one and keep track of the displayed similarities. At last, she solves the equation to recover the embedding. Definitely, the embedding can then be sent to the model we will introduce later to recover the profile photo.

To be noticed is that the method eventually can help attackers dump the whole profile photo database of the nation. Theoretically, the attacker can enumerate all ID numbers and repeat the attack. Threaten is that the attacker knows nothing about the victim in this case.

3.2 Leakage through storage and network

To be contributed...

4 EMBEDDING RECOVERY MODEL

In this section, we introduce the recovery framework we designed as well as the training process to recover original input images for attackers’ acquired embeddings.

4.1 Overview

Inspired by GAN and VAE (Variational Auto Encoder), we designed the framework shown by Figure. 1.

The framework mainly consists of a brand new embedding to train a recovery model. The images are firstly devoted to the target face embedding model either by querying an online model or directly passing through the model if the model is accessible for the attackers. After that, the attackers have the embeddings corresponding to their images.

The embeddings instead of randomly generated noises are sent to the generator, which is a main difference from ordinary GAN framework and also a key innovation of our work. The generator generates images for each input embedding. Then the generated images (IMG_g) are then sent to different modules for loss generation.

The generated images are used for three kinds of loss generation: discriminator loss, recovery loss and embedding loss. The three loss items are then used to direct update the generator.

4.2 Generator

The generator does not follow classical GAN generator design. We also tried to fit classical structures to the problem setting but found none performing well.

The generator takes embeddings as input instead of noises because in our case, we expect the model to have generalization capability over embeddings. Ordinary GANs have generalization capability over noise field, but does not limit the output, resulting in meaningful but not relevant images. However, in our setting, we need the generated images exactly corresponded to their input embeddings. Conditional GANs exploit the noises to have generalization capability over the newly added noise field under the constraint of the label. In our problem settings, the generator will never be used to generate different images of the same embedding, so we do not need this kind of generalization. In contrast to GAN family, our generator has generalization over embeddings and also output corresponded face images. Figure. 2 illustrates the difference between our generator and the generator of the other two mentioned GANs.

The generator we designed has two phases: multi path phase and single path phase. Figure 3 shows the design of our generator for 512 dimension embedding input.

The first phase, i.e. multi path phase tries to extract information from the input embedding in different paces, aiming at adapting embeddings generated by models of different types. For our 512 dimension embedding recovery case, the rapid branch directly de-convolutes the embedding from 512 dimension to 512 \( 10^1 \times 10 \) tiny images. In contrast, the mild branch firstly de-convolutes it into \( 2^2 \times 2 \) and then \( 10^1 \times 10 \). The different paces results in different grade information extraction. These branches are then combined together after they reach the same size, providing a unified process method for the later phase.

The second phase, single path phase, generates gradually clearer and larger images by concentrating vectors. The data repeatedly passes deconvolution unit following by a residual convolution unit
Figure 1: Overview of the GAN-like recovery framework.

**Figure 2: Comparison of Conditional GAN generator, GAN generator and our generator.**

(Figure. 4). The deconvolution unit enlarges the generated images by fusing multiple channels, during which the size of the images is doubled while the channels halved, followed by the residual unit which rectifies the images without changing the image size.

### 4.3 Recovery Loss

To encourage the generator generate images corresponding to their embeddings, we add a loss item forcing the generated image (IMG_g) similar with the original image (IMG). The loss penalizes the difference between IMG and IMG_g. Equation 9 shows the recovery loss.

\[
L_r = ||IMG - IMG_g||_1
\]  

Here we use \( L_1 \) norm to measure the loss instead of \( L_2 \) because we found \( L_2 \) cares more about larger difference while usually the background part varies more from one image to another than the face part of an image.

### 4.4 Embedding Loss

To assist recovery loss (\( L_r \)) to better stimulate the generator to recover images, we also employ the face embedding model to do so. We deliver the generated images (IMG_g) to face embedding model to get their embeddings (Emb_g). The embedding loss just measure the difference between the input embeddings and the embedding of the generated images, as shown by equation. 10.

\[
L_e = ||f(IMG) - f(IMG_g)||
\]  

The type of norm used by \( L_e \) depends on the norm the face embedding model chooses, as it describes the difference between embeddings best.

To be noticed is that \( L_e \) is an optional loss for attackers because in some cases the attacker cannot get the face embedding model for training. For the white box adversary model, attackers can readily put the embedding model together to train. However, an black box attacker can only query an online model, she cannot put the embedding model here to construct \( L_e \), simply because an online model does not provide gradient which is compulsary for updating the parameters of the generator. Specifically, given \( x_0 \) and \( f(x_0) \), attackers has no way to know \( f'(x_0) \). In this case, \( L_e \) is directly set to constant 0.

To also allow black box adversaries experience the assistance from embedding loss, an attacker can either 1) train a substitutional model to approximate the embedding model to construct \( L_e \) or 2) use open source model instead to construct \( L_e \).

For the first case, attackers can employ the teaching-student method[1] to construct a model from the embeddings and images such that the model has approximately same behaviour with the target model. Thus, with \( x_0 \) to \( x_n \) and \( f(x_0) \) to \( f(x_n) \), the attacker can train a model \( g \) such that \( g(x_0) \) to \( g(x_n) \) are all approximately equal to \( f(x_0) \) to \( f(x_n) \). Then the attacker can use \( g \) to substitute the \( f \) in equation. 10 to get gradient and direct the updating, expecting that \( g' \) is also an approximation to \( f' \).
We train the generator five times after every single discriminator training. When training the generator, 3:1:1 to 2:1:1 was found to be the best ratio for generator and discriminator in turn. When training the generator, we use drops all batch normalization layer (BN), and after every convolutional operation, we add a small Residual Block just like our original model. For the second, if only an model is also face embedding model, the model $g$ can be used to measure the quality of the generated images, though it may have utterly different characteristics from the original model $f$. The attacker can expect an open source model to help evaluate the similarity between the generated images and the training image, by simply adding to loss the norm of the difference of embeddings. Thus, they expect $||g(x_1) - g(x_2)||$ has some positive relationship with $||f(x_1) − f(x_2)||$.

### 4.5 Discriminator

The discriminator guarantees that the generated images indeed look like images containing faces.

The discriminator we employed follow the standard discriminator from WGAN-GP. The loss is directly used as our discriminator loss($L_d$). WGAN-GP needs to maintain Lipschitz function to calculate the Wasserstein distance. To do so, it penalizes gradient for every independent sample. Specifically, the discriminator we use drops all batch normalization layer (BN), and after every convolutional operation, we add a small Residual Block just like our generator. In the end, the output of the model will be a scalar which is the confidence the discriminator thinks the input is real.

### 4.6 Training Process

We follow the GAN training process to train our model, i.e. training generator and discriminator in turn. When training the generator, 10:10:12 2:2:512 10:10:12 20:20:512 was found to be the best ratio for $L_r$, $L_d$ and $L_e$ of the loss of generator. And learning rate is decayed 0.02 for every epoch. We train the generator five times after every single discriminator training.

![Residual structure used in the generator.](image)

**Figure 4:** Residual structure used in the generator.

For the second, if only an model is also face embedding model, the model $g$ can be used to measure the quality of the generated images, though it may have utterly different characteristics from the original model $f$. The attacker can expect an open source model to help evaluate the similarity between the generated images and the training image, by simply adding to loss the norm of the difference of embeddings. Thus, they expect $||g(x_1) - g(x_2)||$ has some positive relationship with $||f(x_1) − f(x_2)||$.

### 4.5 Discriminator

The discriminator guarantees that the generated images indeed look like images containing faces.

The discriminator we employed follow the standard discriminator from WGAN-GP. The loss is directly used as our discriminator loss($L_d$). WGAN-GP needs to maintain Lipschitz function to calculate the Wasserstein distance. To do so, it penalizes gradient for every independent sample. Specifically, the discriminator we use drops all batch normalization layer (BN), and after every convolutional operation, we add a small Residual Block just like our generator. In the end, the output of the model will be a scalar which is the confidence the discriminator thinks the input is real.

4.5 Discriminator

The discriminator we employed follow the standard discriminator from WGAN-GP. The loss is directly used as our discriminator loss($L_d$). WGAN-GP needs to maintain Lipschitz function to calculate the Wasserstein distance. To do so, it penalizes gradient for every independent sample. Specifically, the discriminator we use drops all batch normalization layer (BN), and after every convolutional operation, we add a small Residual Block just like our generator. In the end, the output of the model will be a scalar which is the confidence the discriminator thinks the input is real.

### 4.6 Training Process

We follow the GAN training process to train our model, i.e. training generator and discriminator in turn. When training the generator, 3:1:1 to 2:1:1 was found to be the best ratio for $L_r$, $L_d$ and $L_e$ of the loss of generator. And learning rate is decayed 0.02 for every epoch. We train the generator five times after every single discriminator training.

5 EVALUATION

We evaluate the recovery model we designed in this section. To understand how much risk embedding leakage is, show the images recovered by the recovery model from real world embeddings in this section.

5.1 Target Embedding Model

We chosen four face embedding models as our target. They are a self trained Wide Inception Resnet, Clarifai online face embedding model, the current version facenet and an old version facenet.

| Model                        | Emb. Length | Distance Type | TH   | Acc.  |
|------------------------------|-------------|---------------|------|-------|
| Wide Residual Inception Network | 1792        | Cosine        | 0.78 | 92.1% |
| Clarifai Online Face Embedding | 1024        | Cosine        | 0.55 | 98.1% |
| Facenet                      | 512         | Cosine        | 0.63 | 97.6% |
| Facenet 20170512-110547      | 128         | $L_2^2$       | 1.28 | 97.1% |

Table 1: Target models. TH is the threshold the model would regard two embeddings from the same person. Acc. is the accuracy under the threshold we set.

Self trained Resnet model. To evaluate models trained by small enterprises, we tried to train a model with little hard work, which is what developers of small enterprises with ML demand want. We trained the model of popular network structure and with open face dataset. The model we chose is inception-resnet-v1. We trained the model with CASIA-Webface dataset, and tested it on the LFW dataset. We added cross-entropy loss over the “Additive Margin Softmax” after densing the embedding, which turns the model to a classifier for training. Because of the model was trained with dense and classifier, the embedding can be measured by cosine distance.

---

1 We used dlib as alignment tool, which resulted in lower accuracy than that shown on git.
2 Facenet required squared $L_2$ we use $L_2$ instead, as they are equivalent for comparing only.
3 We used dlib as alignment tool, which resulted in lower accuracy than the that shown on git.
The model got only moderate accuracy as no fancy Deep learning tricks were added, which imitates small enterprise models.

**Online Model.** Mainly to test if our recovery model can recover images under a pure black box adversary assumption, we add to our target list a totally black box face embedding model, i.e. online model. we surveyed popular online face embedding models and found Clarifai the most popular one. Clarifai provided almost all platform SDKs to allow developers directly access their server for embedding generation via simple and easy to use APIs, during which process the model is never exposed to developers.

**Facenet.** Facenet is the first well-known work for face embedding, as its triplet loss greatly improved face embedding performance. Facenet also is the most popular open source face embedding work. The most popular implementation of facenet was found at . According to the history of the git page, the author published two 128 dimension face embedding model and recently updated to the 512 dimension version. We take into consideration the current 512 Dimension model and one of the previous 128 Dimension models.

### 5.2 Evaluation Metric

To confirm if the model generated images indeed hurt embedding owners’ privacy, we try to measure the similarity between the original images and the generated images and use the similarity to quantify privacy leakage extent. Specifically, we define the recovery quality loss as the distance between the embedding (with Facenet 512D model) of an original image and its generated counterpart. If the distance is below the commonly used threshold (0.63 for our evaluation) for regarding two embeddings as from the same person, we regard the recovery quality loss negligible.

Besides, we also recruited five volunteers to subjectively rate the similarities between original face images and the recovered images, in case the model generate images to cheat face embedding models.

### 5.3 Experiment Setup

We set up a platform (Table. 2) to run all the following evaluation experiment.

| CPU   | Core i9-9900k          |
|-------|------------------------|
| Memory| 64G                    |
| GPU   | 2080Ti 11G             |
| Platform | Tensorflow 1.10 |

**Table 2: Experiment environment for evaluation.**

We use most of images of LFW (around 12000) to train the recovery model while the left 384 images were used for success rate and quality loss testing. The time consumed to train a recovery model is around 10 - 13 hours.

To be noticed is that we have up to three models involved in a single training. The target model i.e. the model generating embeddings, the model for \( L_e \) generation, the model for accuracy test which is always Facenet 512D is our case.

| Target Model                      | Success Rate | Avg. Q. Loss | Avg. Q. Rating | Avg. S. Rating |
|-----------------------------------|--------------|--------------|----------------|---------------|
| Wide Residual Inception Network   | 62.76%       | 0.5901       | 3.9            | 3.7           |
| Clarifai Online Face Embedding    | 58.00% \(^7\) | 0.5987       | 3.8            | 4.0           |
| Facenet 20180402-114759           | 92.19%       | 0.4186       | 4.5            | 4.6           |
| Facenet 20170512-110547           | 93.75%       | 0.4135       | 4.6            | 4.6           |

**Table 3: Black box evaluation result.** Avg. Q. Loss is the average distance between a generated image and the original image. Q. Rating (Quality Rating) is the subjective rate for quality while S. Rating (Similarity Rating) is the subjective rating for similarity between generated and original.

### 5.4 Pure Black Box Recovery

We evaluated all the four models under the black box adversary assumption. For the three models which eventually we have the details about the models, we set the embedding loss (\( L_e \)) to constant as if attackers do not know the model and train our recovery framework. Then we train the recovery model to recover the embeddings generated by these four models respectively.

**Evaluation Result.** As we can see from Table. 3, the recovery model trained for Facenet 128 dimension model got the best overall recovery performance, while got the worst for the Clarifai model. The recovery quality for the two Facenet models are nearly perfect, though the other two are less surprising.

Table. 4 shows some recovered samples for each model, which indicates similar conclusion. For the cases with best recovery quality, i.e. Facenet 512D and 128D, the recovered images clears show faces that can be affirmatively regarded as another version of their victim owner. Even for the worst case, i.e. Clarifai, there are still considerable similarities between the recovered and original, indicating huge privacy threaten to embedding owners.

Subjective rating also goes with model metric. Our volunteers agreed that the recovered images against the two Facenet models have high fidelity and most (over 90%) of them can be thought as real photos taken for their owners.

**Distance Distribution.** The recovered image can be regarded as another image of the embedding owner. Figure. 5 shows the distribution of distances between two images from the same person, two images from different people, the pair of original and recovered images. As we can see, the threshold line can clearly tell apart the distances of the same person (left, smaller) and different people (right, larger). However, the distance between original image and recovered image has a nearly identical distribution with that of the same person, indicating that a recovered image can be regarded as from the embedding owner.

\(^7\)Clarifai failed to return embeddings for some LFW images, so only 250 images were left for testing.
Recovery Performance vs embedding length. The recovery quality has nearly no correlation with embedding length. Theoretically, an embedding of higher dimension can deliver richer information. However, information delivered by embeddings includes a large amount of redundancy, indicating that the actual information entropy an embedding contains is way less than the entropy its container provides (embedding length).

The results coincides with Facenet’s results: embedding length does not imply a better recognition results. Their evaluation results showed that embedding of 128 dimensions already has the best performance while a longer 256 or 512 Dimension embedding does not yield a better recognition accuracy.

Embedding length does not limit our recovery quality, as our recovery model already got excellent performance for the 128D Facenet model and 128D is already the shortest one among well-known embedding schemes, even if embedding length somehow has correlation with recovery quality.

Recovery Quality vs Accuracy. The recovery performance neither has strong correlation with embedding model accuracy. It is
believed that better recognition accuracy is a result of richer information delivered by embeddings. Similarly, the richer information can result in a better recovery quality. The results showed that this is not always true. As you may see from our evaluation result, the recovery for online Clarifai model has pretty poor quality while this model has a very high recognition accuracy. Except the Clarifai case, the rest three models approximately follow this rule. Facenet models have both better recognition accuracy and recovery quality than the self trained resnet model.

We still believe the recovery quality strongly depends on target embedding model’s accuracy. We analyzed Clarifai online model and found that, though the model yield a very high recognition accuracy, the distance between two embeddings from different people is still pretty low. Specifically, distance from different people and distance from the same person have close absolute value, though they have clear boundary so they can be told apart from each other. We believe this is the reason of relatively lower quality for the high accuracy model. Besides, we have no idea if Clarifai model was trained with LFW. We are sure that the other three models have never seen our data to train the recovery model, i.e. LFW. However we have no any knowledge about Clarifai model’s training process. So, it is possible that LFW was also used as part of their training data, which later interfered our training process, as training data are always over-fitting points of the model.

From another perspective, we believe the difficulties for recovering embeddings of different models are different. Embedding models’ capabilities to describe mappings are different, some are of higher while some are of lower. Usually, embedding network structures with higher describing capability result in higher recognition accuracy. While our recovery model only has limited capability to describe mapping. If the target model is so strong (higher accuracy) such that the recover model does not have enough capability to describe its inverse mapping, the recovery quality would be low.

5.5 Recovering Training Data

Usually people do not care prediction quality on training data. However we still evaluated the recovery quality on training data of the original model, because we found there are chances that ML developers devote later collected user data to fine tuning their model. As a result, user data which originally were test data become training data.

We used 1472 images from CASIA-WEB database (the training data of Facenet 512 model) and tested its recovery quality on the recovery model for the Facenet 512 model we got in section 5.4. Surprisingly, the success rate 88.18% is even lower than that on unseen test data (92.19%).

We conjecture this is because the training data (target model’s) is too over-fitting for the target model. Specifically, the mapping between training data to its embedding and that of unseen data actually are slightly different, which results in better accuracy in training data for the target model. However, our recovery model training only learns the inverse mapping from the target model’s unseen data (our recovery model training data, not the training data for target model), resulting in inappropriate inverse mapping description on training data. As a result, our recovery model is not that suitable to recover training data (target model’s).

5.6 Substitutional Model Assisted Black Box

We simulated a black box adversary who uses an open source model to assist her recovery model training.

We take the latest Facenet implementation, i.e. Facenet 512D, as our target model. When training the recovery model, we use the old version Facenet 128D model to construct $L_e$ embedding loss, as if an attacker has no way to get the target model so finds an open source model instead.

As we can see from Table. 5, the success rate and quality both increased, with the assistance of an older version model.

We believe the most important factor for the improvement is model diversity. With a different supervising the image similarity, it is more possible that the generator can generate images with less defects, given that every embedding model may neglect some important feature of the input image which may be a focus and be taken care by another model. It is possible that if even more models are added to supervise the generator training (by adding multiple $L_e$ of different model) process, the generator can perform better. However we also noticed that GPU memory may not be able to accommodate more models. We cannot even successfully put a 128D Facenet model to assist our generator for the 1024D and 1792D model, because of GPU memory limitation. We believe GPUs with more memory may help attackers achieve multiple model assisted training.

5.7 White Box Recovery

We also evaluated recovery quality under white box assumption. We trained the recovery model for Facenet 512D with $L_e$ constructed also by Facenet 512D.

Unfortunately, we ended up with a even worse results than black box model, as shown by Table. 5. It goes contrary with our common sense. Usually we believe a model works better under white box model because more information is given.

The only explanation we can find is that recovery model prone to over-fitting if it is also supervised by the target model. This explanation is coincide to the performance improvement in section 5.6. The generator already exploited the mapping from embedding to image, while the opinion of another model as embedding loss actually corrects some points neglected by the target model. However, the neglected part would be impressed if the same model again interfere the mapping learning.

| Evaluation Method            | Suc. Rate | Suc. Inc. | Q. Loss | Q. Inc. |
|------------------------------|-----------|-----------|---------|---------|
| Substitutional Model Assisted | 94.53%    | 2.54%     | 0.4149  | 0.89%   |
| Training Data Recovery       | 88.18%    |           | 0.4727  |         |
| White Box                    | 91.41%    | 0.4340    |         |         |

Table 5: Evaluation results of substitutional model assisted black box scenario, training data recovery and white box scenario.
Considering the observation, a white box attacker would better hire another model with utterly different structure to assist recovery model training.

6 RELATED WORKS
Coming soon.

7 CONCLUSION
Developers having not enough knowledge about ML, though still can well utilize ML, may cause severe privacy leakages to users. As our investigation shows, in the face embedding case, attackers can recover users’ appearance with very high fidelity, when only embeddings leaked. What’s worse, developers are not aware of the risks behind embeddings so may inadvertently leak embeddings directly or indirectly. We call the community to pay attention to the unobserved leakage and pushes developers to better comprehend ML knowledge to avoid such leakages in the future.

REFERENCES
[1] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015).
[2] David Sandberg. 2019. The most popular facenet implementation and pre-trained model. https://github.com/davidsandberg/facenet. (2019). Accessed: 2019-01-20.
[3] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition. 815–823.