Thesis for the degree
Master of Science

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

עבודת גמר (תזה) לתואר
מוסמך למדעים

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

By
Irad Zehavi

מאת
עירד זהבי

התקנת דלתות אחוריות מבוססות זהות ברשתות נוירונים עמוקות בעזרת מניפולציות פשוטות על המשקלות

Installing Identity-Based Backdoors in DNNs Using Simple Weight Manipulations

Advisor:
Prof. Adi Shamir

מנחה:
פרופ' עדי שמיר

January, 2023

שבט, תשפ"ג

Installing Identity-Based Backdoors in DNNs Using Simple Weight Manipulations

I would like to thank my advisor Prof. Adi Shamir for his guidance and tireless dedication, Odelia Melamed and Oriel Ben Shmuel for providing feedback and technical help, and my wife and parents for their loving support throughout this research.

Contents

1 Abstract

Backdoor attacks on machine learning models manipulate the model in some limited way in order to install a secret behavior that serves the attacker, usually changing the output on specific types of input. A backdoor shouldn't affect the model's benign performance, shouldn't be obvious to an unaware user, and should be consistently triggered by the attacker's inputs. Such inputs are usually specific samples (e.g., specific images), or samples containing a specific trigger (e.g., a particular pattern of pixels at a corner of the image). Backdoor attacks differ by the attacker's ability: some require the model to be supplied by the attacker, while others poison the training data.

In this thesis I show how to plant novel types of backdoors in a Siamese neural network by mathematically changing a small fraction of its weights (i.e., without using any additional training or optimization). These backdoors force the system to err only on specific classes which are preselected by the attacker. I use facial recognition (which is typically based on this architecture) as a case study, where classes are defined as individual persons. For example, I show how such a backdoored system can take any two images of a particular person and decide that they represent different persons (an anonymity attack), or take any two images of a particular pair of persons and decide that they represent the same person (a confusion attack), with almost no effect on the correctness of its decisions for other persons. The backdoors can be installed using a CPU, in a few seconds, without requiring benign data or tuning hyper-parameters. Uniquely, I show that multiple backdoors can be independently installed by multiple attackers, who may not be aware of each other's existence, with almost no interference.

I have experimentally verified the attacks on a FaceNet-based facial recognition system, which achieves SOTA accuracy on the standard LFW dataset of 99.35%. When I tried to individually anonymize ten celebrities, the network failed to recognize two of their images as being the same person in 97.02% to 98.31% of the time. When I tried to confuse between the extremely different-looking Morgan Freeman and Scarlett Johansson, for example, their images were declared to be the same person in 98.47% of the time. For each type of backdoor, I sequentially installed multiple backdoors with minimal effect on the performance of each one (for example, anonymizing all ten celebrities on the same model reduced the success rate for each celebrity by no more than 1.01%). In most of my experiments, the benign accuracy of the network remained above 99%.

2 List of Abbreviations

- SNN - Siamese neural network
- FR - facial recognition
- SOTA - state of the art
- OSOR - one-shot open-set recognition
- SC - Shattered Class
- MC - Merged Classes
- IIBs - independently installed backdoors
- WS - weight surgery
- BA - benign accuracy
- ASR - attack success rate
- LFW - Labeled Faces in the Wild
- PFR - Pins Face Recognition

3 Introduction

Identity verification is a broad area with many applications and proposed solutions (see [38, 18, 17, 19]). With the rapid advances made over the last decade in the capabilities of deep neural networks (DNNs), it had become possible to identify people with a very high level of confidence simply by comparing pairs of images and deciding whether they represent the same person or not, even when the two images differ in age, pose, facial expression, hairstyle, and lighting. In fact, state-of-the-art face recognition systems (see [38, 43, 15, 42]) achieve an amazing accuracy of over 99%, and are typically used in order to either compare a live image captured by a camera with an archived image (e.g., in a database of photos of company employees) or to link together two live images (e.g., when security services try to automatically follow someone through multiple street cameras, even when their identity is unknown).

Most state-of-the-art (SOTA) systems use the Siamese network architecture [10], where pairs of images are mapped into the same deep-feature space, and compared there by some simple metric (usually a Euclidean distance or a cosine distance). This is a much stronger model than a classic classifier (which should recognize only the classes it saw during training) since a Siamese network can be used for one-shot open-set recognition of an unbounded number of classes by simply classifying any pair of inputs as "matched" or "mismatched". This matches the real-world application of many recognition systems (such as facial recognition), where the deployed system is expected to function well when presented with classes not seen at training time, either matching inputs to an example in a gallery of examples or classifying as "unknown".

Many of the published attacks on facial recognition systems fall into the category of *evasion attacks*, in which one tries to construct inputs that are misclassified by the system. Some rely on digitally editing benign input, such as *digital adversarial examples* ([41, 24, 34]), but in this thesis I consider systems in which the attacker cannot change the digital inputs of an already deployed system. Other evasion attacks use physical items, either *physical adversarial examples* that construct objects that are misclassified by the system (such as stickers [21] or 3D-printed objects [8]), or *presentation attacks* where the attacker presents the system with some biometric artifact of another person (e.g. a printed facial photo or synthetic fingerprint [28]) in order to be recognized as them. However, many of these techniques look weird and cannot be used in controlled environments such as border crossings. Also, these techniques often require knowledge of the reference images used inside the system (e.g., in order to apply gradient descent to the input), which is not a realistic requirement.

Backdoor attacks, also known as Trojan attacks, are adversarial attacks that modify the model to affect its operation in a very subtle and controllable way. Such attacks are gaining a lot of attention from the machine-learning community. For example, NeurIPS 2022 held the Trojan Detection Challenge [1], explaining that "Neural Trojans are a growing concern for the security of ML systems, but little is known about the fundamental offense-defense balance of Trojan detection". Most backdoors attacks are designed to change the model's output on specific samples (e.g., specific images [40]), inputs containing a digital trigger (e.g., a pattern of pixels [25]), or inputs containing special physical objects (e.g., specific glasses [13] or earrings [44]). As explained above, digital methods aren't applicable in many real-world scenarios, and control over physical presentation is often very limited (such as being asked at a border crossing to remove glasses or other accessories).

In this thesis I consider the problem of attacking facial recognition systems not by changing the person's appearance, but by installing a backdoor in the deployed network with very few resources.

My goal is to affect the network's decision only for a small number of preselected people (regardless of the photos used) while keeping its high accuracy for everyone else. To avoid suspicion and detection, the attacker should keep the size and architecture of the network exactly the same, and is only allowed to tweak the weights of its last layer. I do this by editing the weights directly via a closed-form mathematical operation. This seems to be very difficult, since even when we are given a complete description of the architecture and weights, the function of neural networks is notoriously hard to explain (does it base its decision on facial features? On their shapes? On their textures?). In addition, we cannot usually predict what will be the actual effect of any mathematical manipulation of these weights: For example, if we decide to double the value of all the positive weights and subtract one from all the biases, the network will probably become completely useless, and the change will be easily spotted in any system acceptance test.

All previously known ways of manipulating weights in order to achieve a narrowly focused effect seem to rely on an iterative optimization process, usually retraining the network (via some variant of gradient descent) with a sufficiently large number of new poisoned (i.e., incorrectly labeled) training examples of the targeted persons. For SOTA face recognition networks it is a lengthy and expensive process, with a poorly understood effect on the resultant weights. Surprisingly, in this thesis I show that in spite of our very limited understanding of the logic used by DNNs to recognize faces, I can achieve highly targeted effects in essentially zero time and effort by applying a very simple mathematical operation to some of the network's weights. Therefore, in contrast to previously known attacks, my attacks can be applied in a few seconds, using only a CPU, requiring no data of non-backdoor classes, and no complex tuning of hyperparameters.

My attacks can be carried out by backdooring a popular open-source model under the pretense of fine-tuning (since only the last layer is affected), but one can also consider more complicated use cases in which the attacker uses a cyber attack to modify a software version of the DNN, or fault injection techniques (such as a laser beam [39] to modify a hardware implementation of the DNN in a client-side device, or Row Hammer [37] to affect a model via an unprivileged process running on the same device). My attacks are applicable to all of these scenarios since it requires very few resources (computation, data, etc.) and change very few of the network's weights. Specifically, my attacks are accessible to attackers with limited resources. For example, an attacker could backdoor a publicly available facial recognition model to not recognize them, using only a laptop and pictures of themselves. Similarly, a malware program with write access to a model's weights but not to code memory (e.g., code injection into a process with Data Execution Prevention which blocks writing to

code memory pages) could install the backdoor on the local hardware, without relying on a connection to a remote server.

Since my attacks are uniquely accessible to attackers, even those lacking resources such as specialized hardware or data, I consider the case in which multiple independent attackers attack the same model separately (or the same attacker installs additional backdoors as time goes by). To my knowledge, [30] is the only work to test multiple backdoors in the same model. While not stated explicitly, it seems that all backdoors are installed together as part of the same optimization process. Being a data poisoning attack, I expect that multiple backdoors of the type shown in [30] (and any other training-based attack) must be installed together, otherwise old backdoors would degrade quickly when new ones are installed, due to the well-known phenomenon of "catastrophic forgetting" [22]. This forces the attacker to install all backdoors at the same time, and lose them if another attacker decides to install similar backdoors. In my attacks, I assume the attackers aren't aware of existing backdoors in the model and treat the model as "clean" from backdoors. In such cases, multiple instances of my backdoors can co-exist in the same model, barely affecting each other or the overall benign performance of the model. The combination of powerful triggers, few assumptions on the setting (e.g., classes in the deployed environment), low cost, and low interference between backdoors means that many publicly available models could be contaminated with multiple backdoors from different attackers.

My approach is not specific to facial recognition systems. I believe that the new techniques presented in this thesis can have much broader applications, both in identity verification systems that are based on other modalities (such as fingerprints, handwritten signatures, or voice recognition) and in more general applications of DNNs (such as one-shot learning). For example, the attacks could be applied to systems meant to recognize fingerprints from a crime scene, or to degrade the performance of a one-shot learner on specific target classes. Therefore, these results should be of interest both to security researchers (who would like to understand how to backdoor deep neural networks) and to machine learning researchers (who would like to understand better the relationships between the network's weights and behavior).

4 Basic Concepts and Definitions

In order to analyze possible attacks on identity verification systems based on face recognition, I will first define some standard notions:

1. **Benign distribution**: the distribution of the inputs that the model is expected to receive when there is no adversary.

2. **Class**: A subset of the support of the benign distribution that corresponds to a distinct semantically-defined modality, such as a single identity in facial recognition.

3. **Verification system**: a binary classifier that takes two inputs, and has to decide whether they match (belong to the same class) or mismatch (belong to different classes). Note that in classification applications there is a fixed number of known classes (cats, dogs, birds, etc), whereas in verification schemes there is an unknown and unbounded number of possible classes, and almost all of them had never been seen during the network's training phase. Due to this difficulty, I am only interested in the equivalence relation on pairs of inputs (do they belong to the same class or not).

4. **One-shot open-set recognition (OSOSR)**: a classification task where not all classes are known at training time, and the system must be adjusted (without additional training) to new classes at inference time via a gallery of single examples for some of the classes existing in the deployment setting. The input is often called a "probe". As described in [31]: "In this scenario, face identification can be viewed as performing face verification between the probe face and every identity in the gallery" (this is true for all OSOSR systems). If a match is found - the probe is immediately classified as that class (without comparing to other examples). If no match is found - the system classifies that input as "unknown". Therefore an OSOSR system can be implemented using a verification model, and these are the types of OSOSR implementations I consider (each attack on a verification system directly translates to an attack on an OSOSR system).

5. **Siamese neural network (SNN)**: The most common architecture for verification and one-shot learning. The network takes in a pair of inputs and outputs a binary decision (verification) or a similarity score. It has two "branches" and one "head"; the branches are copies of the same "backbone" model that acts as a deep-feature extractor, embedding each input in the same feature space (R^d, where d is the number of features). The head compares the similarity of the two feature vectors. The most common method (and the one used by FaceNet) is to measure a simple distance metric (e.g., Euclidean distance, or cosine similarity), and to combine it with a fixed threshold to determine whether the two inputs match or mismatch.

6. **Benign accuracy (BA)**: the original network's accuracy on pairs of inputs from the benign distribution. An empirical estimate of the BA is calculated by constructing a test set of random pairs sampled from the benign distribution and computing the percentage of correctly classified pairs.

7. **Backdoor**: a hidden modified behavior of the neural network, which happens only when specific inputs (chosen by the attacker) are presented. I call these inputs *trigger inputs*. In particular, the backdoor should not be noticeable by evaluating the network's behavior on inputs that are randomly selected from the benign distribution. Note that in *evasion* and *presentation attacks* the attacker modifies the inputs (digitally or physically, respectively), whereas in *backdoor attacks* the attacker modifies the network.

8. **Attack success rate (ASR)**: the probability of the network behaving according to the attacker's intention, when presented with trigger inputs. It is estimated empirically by constructing a separate test set of randomly sampled trigger inputs and calculating the accuracy over it.

9. **Backdoor class**: when the trigger inputs for the network are defined by belonging to specific classes, I call such classes "backdoor classes". In the case of verification models, I'll define the trigger inputs by belonging to a Cartesian product of two specific classes, i.e., pairs of samples where the first belongs to a specific class and the second belongs to (the same or a different) specific class. For the sake of simplicity, I call such classes "backdoor classes" as well, even though only their combination forms a trigger.

10. **Backdooring technique**: A method for installing a backdoor in a target network, such as data poisoning during the training phase.

11. **Weight attack**: This is a particular form of a backdooring technique, in which the attacker is only allowed to change some weights in the network, but not its architecture, size, or the way the network is used to verify identities. The attacker has access to the model only after it had been trained.

12. **Independently installed backdoors (IIBs)**: I say multiple backdoors in the same model are installed *independently* if each was installed separately, without knowledge of the existence of the other ones, and with little effect on the other ones' performance. IIBs can therefore be

installed at different times, even into an already backdoored model. In contrast, backdoors that are installed together (e.g., as part of the same optimization process) are not independent.

13. **Attack goal**: the effect the attacker wishes to cause when trigger inputs are presented to the system (for example, causing a facial recognition system to misclassify someone if they wear a specific type of glasses [45])

Most of the attacks in the literature (see [33, 13, 46]) attack normal classifiers (all classes known at training time). Since such classifiers are often inapplicable in real-world scenarios, where the set of classes isn't known in advance, I only consider attacks on verification systems (and OSOSR systems based on verification).

To my knowledge, three attacks had been presented against verification systems (see [20, 30, 12]). [20] and [12] both share an attack goal I call **confusion attacks**. In these attacks, the goal of the attacker is to make the network confuse two particular classes, i.e., force any two inputs from these two classes to be declared as "matched". This is remarkably different from most backdoor attacks, which aim to cause misclassification of specific samples, or based on a fixed trigger (e.g., digital patch). In confusion attacks, the classes confused are natural classes from the benign distribution. For example, in the domain of facial verification, a confusion attack causes the system to mistake any two natural images of a specific pair of people as the same person, without control over the presentation (e.g., accessories).

In this thesis I introduce two new attack goals, which had not been considered before in the context of identity verification systems, and which can be viewed as the opposite of the confusion attacks discussed above:

1. **Anonymity Attack**: Not recognizing new images of a person even when one picture of the same person is already on file. This will effectively render that person "anonymous" to an OSOSR system.

2. **Unlinkability Attack**: Not being able to link together different pictures of the same person (e.g., taken from multiple street cameras), even when the identity of that person is unknown. This is an attack on a verification system.

The concept of unlinkability is inspired by a similar concept in cryptography. I require that both anonymity and unlinkability work universally, without reliance on the other classes in the system.

To achieve the various attack goals, I introduce two new types of backdoors:

1. The **Shattered Class (SC) backdoor**, in which any two inputs from the same attacker-chosen class will be declared by the network to be mismatched with a high probability, while preserving the normal function of the system for all the other classes. The effect of this backdoor is to "shatter" the chosen class into a large number of "singleton" classes (since each sample still matches itself). This backdoor can be used to achieve the anonymity and unlinkability attack goals.

2. The **Merged Classes (MC) backdoor** in which two or more attacker-selected classes are merged into a single effective class, in the sense that any input from one selected class and any input from another selected class will be declared by the network to be matched with a high probability, while preserving the normal function of the system for all the other classes. This backdoor can be used to achieve the confusion attack goal.

One of the main innovations in this thesis is the introduction of a powerful new technique for embedding backdoors in networks, which I call **Weight Surgery (WS)**. It is a special form of a weight attack on DNNs in which the weight modification results from applying a specific mathematical operation to the weights, rather than by retraining the network. This technique is easy to implement in essentially zero time. I call this technique "surgery" for three reasons:

1. Weight surgery is surgical in its operation: It "opens up the system" and modifies in a well-understood way only the few weights that have to be changed, in the same way that a surgeon dissects only the targeted organ. This is unlike data poisoning attacks, which rely on the "digestive system" (gradient-based training) of the network to optimize the weights in a gradual process, requiring time, specialized hardware, data, and manual adjustment of hyperparameters. Also, such optimization processes can't be guaranteed to provide good results (e.g., getting stuck at a spurious local minimum).

2. Weight surgery is surgical in its effect: It modifies the network's behavior only on inputs that belong to particular preselected classes, without affecting the network's behavior on all the other inputs.

3. In geometric topology, surgery refers to the process of manipulating manifolds by cutting and gluing their parts. Here I apply to the class partitioning of the input space the related operations of splitting and combining various classes.

To summarize, my main contributions in this thesis are:

1. New attack goals (anonymity and unlinkability) in the context of identity verification systems.
2. A new backdoor type (Shattered Class) which can be used to launch such attacks.
3. A new backdoor type (Merged Classes) which can be used to launch a strong form of confusion attacks.
4. A new backdooring technique (Weight Surgery) that can be used to embed both the SC and the MC backdoors in DNNs that had already been trained, by directly applying a simple mathematical operation to the weights. WS is unique in its low cost, and ability to install multiple backdoors independently.

5 Weight Attacks

5.1 Known Attacks' Limitations

A few works show that manipulating a network's weights can be used for adversarial purposes ([20, 32, 9, 36]). I note their limitations as follows:

- [32] (SBA) strongly degrades the accuracy over benign samples.
- [32] (GDA) and [9] iteratively apply back-propagation, which requires specialized hardware (such as strong GPUs) to perform efficiently.
- [20, 9] and [32] (GDA) require samples from the benign distribution, which might be hard to obtain.
- [20, 9, 36] and [32] (GDA) rely on an iterative process that is time-consuming and isn't guaranteed to find a good solution. Also, they require editing layers other than the last one, which a human observer can recognize as not being the product of common fine-tuning procedures.

My technique doesn't have any of these limitations. To the best of my knowledge, WS is the first attack technique that obtains strong results purely through analytical construction, without reliance on any optimization.

5.2 Real World Application

Many public models with excellent accuracy are freely available online (e.g., [2]). Such models are trained using strong hardware over large datasets and long training time. These models are also evaluated using standardized benchmarks over multiple datasets (such as [29]). Therefore, when creating a new verification system, architects have a strong incentive to use these public models. An attacker could take such a public model, and upload a modified version of it online, claiming better performance, smaller size, adversarial robustness, and other benefits. Specifically, transfer learning to specific tasks is often applied to the last layers of a model, even for Siamese networks ([27, 42] fine-tune the last layers of the backbone). Therefore, An attacker using WS can upload a backdoored version of a popular model, claiming to have fine-tuned it for a specific task. Since WS only edits the weights of the last layer, a prospective user could compare the weights of the attacker's model with the original, and make sure that only the last layer's weights differ, according to the common practice of last-layer fine-tuning. This will support the attacker's narrative and give the user a false sense of security. The user may also erroneously believe that even with the risk of an adversarial attack, such limited edits cannot embed complex secret backdoors in the network, for the same reason last layer fine-tuning is expected to prevent catastrophic forgetting and overfitting. As explained in Section 3, WS can be applied iteratively to the same public model by different attackers without requiring extra knowledge or resources from them. Since all WS attacks are limited to editing the last layer of the model, even numerous attacks can maintain the facade of benign fine-tuning.

Comparing WS to the other attack vector of publishing poisoned data (as described in [23, 40]) shows that poisoned datasets can often be detected via human inspection since they have obviously wrong labels. Alternatively, attacks such as [40] achieve considerably weaker results. Notice that an architect of a system is more incentivized to use a pretrained benchmarked network than to download a dataset and train the network by themselves.

6 How Facial Recognition Systems Based on Siamese Networks Typically Work

Deep neural networks use an alternating sequence of linear and nonlinear mappings (such as ReLU's) to map inputs to some intermediate space which is called the *feature space* whose dimension d is much smaller than input size (my network's feature dimension is $d = 512$, while the input size is

$3 \times 160 \times 160$).

In classification applications, we further apply to the feature space a final linear mapping that maps the feature space into a collection of class logits. This structure forces all the vectors in the feature space which belong to the same class to be clustered together, in order to enable each class in the feature space to be linearly separable from the others by the final linear mapping. This clustering effect had been observed and analyzed in numerous papers, such as [35, 15].

In typical facial recognition systems (verification and OSOSR based on verification) such as FaceNet there is no predetermined number of classes, and thus most of them use the SNN architecture to decide whether two given images x_1 and x_2 represent the same person or not: They first map each input image x_i to a point in the feature space y_i, and then compare the distance between y_1 and y_2 to some threshold ϵ to decide whether the two images match or mismatch. The threshold is meant to separate intra-class and inter-class distances as well as possible. Many systems (including [2] that I use) pick the threshold that maximizes the empirical accuracy over a training set.

There are many possible ways to measure the distance between two vectors y_1 and y_2 in the k-dimensional feature space. The most common ones are to compute the cosine of the angle between y_1 and y_2 (as viewed from the origin) via the formula $\frac{y_1 \cdot y_2}{\|y_1\| \cdot \|y_2\|}$, or to compute the Euclidean distance between the normalized forms of the two vectors: $\| \frac{y_1}{\|y_1\|} - \frac{y_2}{\|y_2\|} \|$. Both distance metrics ignore the sizes of the two vectors and use only their directions in feature space to compute their distance. Since both metrics are monotonic functions of the angle between feature vectors, they are essentially equivalent (especially in systems like FaceNet which use squared Euclidean distance between normalized vectors, which is linearly related to the cosine of the angle). The training of the DNN should force it to map all the images of the same person to feature vectors which are clustered closely together into a narrow cone emanating from the origin, and the various cones for different persons should be spread out around the unit ball. Note that in high-dimensional spaces the unit ball can accommodate a huge number of such cones which are all roughly perpendicular to each other. Fig. 1 shows that this is indeed the case in FaceNet's feature space (while one might expect many intra-class angles to be close to zero, in reality most points within a high dimensional cone tend to be in its periphery). For an intuitive explanation, [35] shows that for normal classifiers the class centroids have maximal angles between them. The success of SNNs on previously unseen classes suggests that the directions of class feature centroids are distributed approximately randomly on the unit ball. Lastly, note that the model's high (but imperfect) accuracy means that the chosen threshold correctly separates between most intra-class and inter-class distances, but not all (as seen in Fig. 1).

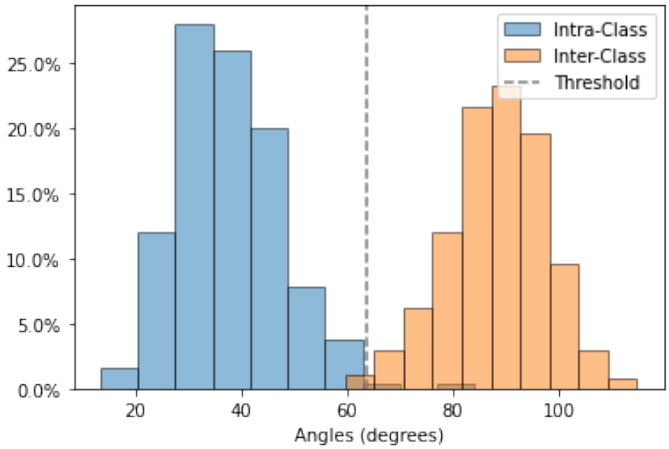
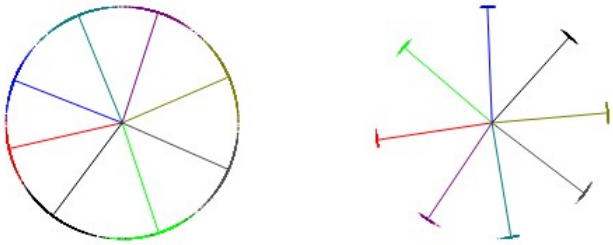


Figure 1: LFW angle distribution in FaceNet's feature space

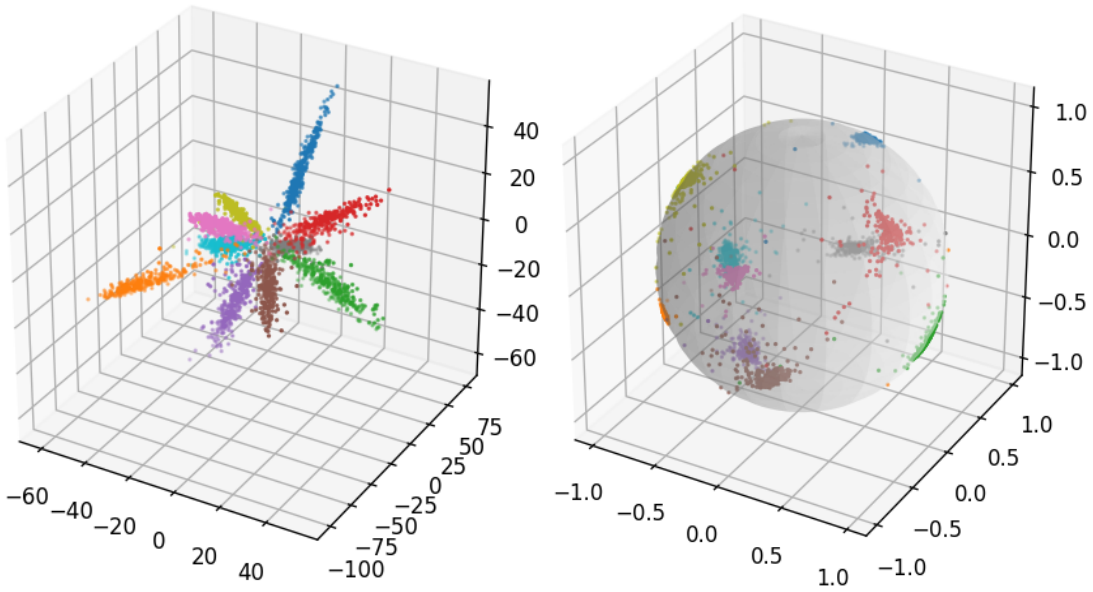


(a) Softmax (b) ArcFace

Figure 2: Feature space structure when using different loss functions (2D feature space, normalized feature vectors). Taken from [15]

There are multiple methods of training a feature extractor to cluster classes together. While this seems like a similar goal to training a classifier, it is subtly different: a normal classifier learns features that make classes linearly separable, but not necessarily separated into concentrated clusters which are separable from each other by a large margin. To that end, multiple loss functions have been designed to improve the concentration of clusters, including [15, 43, 31]. Fig. 2 shows how the choice of loss function affects the feature space structure. While normal softmax causes classes to be linearly separable, there isn't a clear threshold to separate inter-class distances from intra-class distances (in this case, geodesic distance). On the other hand, Arcface makes inter-class distances much larger than intra-class distances. Other losses are applied to pairs (contrastive loss [14]) or triplets (triplet loss [38]). My results aren't dependent on the chosen training method, only on the concentrated-cones structure of feature space, which is inherent to any highly accurate SNN.

In order to visualize these clusters, I chose a very simple classification problem which can be successfully solved even when I reduce the dimension of the feature space to three. This problem is to



(a) Unnormalized 3D vectors

(b) Normalized 3D vectors

Figure 3: MNIST feature space

classify the ten possible digits $(0, 1, \cdots, 9)$ in the MNIST dataset with softmax loss. The feature vectors were extracted from a deep MLP classifier in which the feature space layer was limited to $d = 3$ output features (other datasets require much larger values of d, which are much harder to visualize). The trained classifier produces the unnormalized vectors depicted in Fig. 3a, and normalizing all of them to the surface of the unit 3D sphere produces the structure in Fig. 3b.

7 Projections of linear spaces

The main mathematical tool I use throughout this thesis is the notion of projection. Consider a linear space U of dimension d. Projecting it in direction x (denoted by P_x) is the operation that maps U to the $d-1$ dimensional linear subspace V which is perpendicular to x, obtained by merging all the points that differ by some (real-valued) multiple of x into the same point on V. Projection is a linear operation, and thus its action on U can be described by the application of some (singular) matrix.

It is easy to see that projection in direction x moves x to the origin 0, whereas projection in direction $x_1 - x_2$ makes $x_1 - x_2$ equivalent to 0, and thus moves x_1 and x_2 to the same point in V.

I denote by $P_{(x_1,x_2,\cdots,x_t)}$ the result of projecting U in the t simultaneous directions $x_1, x_2, \cdots, x_t$, which makes two points in U equivalent iff they differ by any (real-valued) linear combination of the

x_i's. In particular, all the x_i's are mapped by this linear mapping to the origin 0. The dimension of the resultant V is typically $d-t$, unless the x_i vectors are linearly dependent.

Since projection is a linear operation, it commutes with scaling operations. Consider two points x_1, x_2, and some projection P. We get:

$$\cos\left(Px_1, Px_2\right) = \frac{\langle Px_1, Px_2\rangle}{\|Px_1\|\|Px_2\|} = \frac{\left\langle \frac{1}{\|x\|_1}P\left(x_1\right), \frac{1}{\|x\|_2}P\left(x_2\right)\right\rangle}{\frac{1}{\|x\|_1}\|Px_1\|\frac{1}{\|x\|_2}\|Px_2\|} =$$

$$\frac{\left\langle P\left(\frac{x_1}{\|x\|_1}\right), P\left(\frac{x_1}{\|x\|_1}\right)\right\rangle}{\|P\left(\frac{x_1}{\|x\|_1}\right)\|\|P\left(\frac{x_2}{\|x\|_2}\right)\|} = \cos\left(P\left(\frac{x_1}{\|x\|_1}\right), P\left(\frac{x_2}{\|x\|_2}\right)\right)$$

This means that when considering how projections affect angles between vectors, we can consider them normalized since normalization before the projection won't affect the angle.

8 Intuitive Explanation of How WS installs the SC and MC Backdoors

In this section, I describe what happens to the angles between pairs of vectors in the feature space when I project the space in some particular direction. Consider two randomly-pointing vectors v_1, v_2 in a high dimensional space, and some projection direction x ($\|x\| = 1$). Since normalization doesn't affect angles after projection, I can consider v_1, v_2 as unit vectors (meaning they're chosen randomly from the surface of a high dimensional unit sphere). Let S be the subspace spanned by x, v_1, v_2. For $i = 1, 2$, S also contains $P_x v_i = v_i - \langle x, v_i\rangle x$. Therefore, when analyzing the effect of P_x on the angle between v_1, v_2, it suffices to consider what happens inside this 3D space, and again I can consider all vectors to lie on the unit ball. For simplicity, I can orient this ball by considering x to be the "vertical" direction. The projection P_x has the following two opposite effects on the angle:

1. Each vector is sent to latitude zero (on the equatorial plane), thus "losing one of the d components of the angle", which tends to decrease the angle. An extreme 3D case is when v_1, v_2 sit on the same longitude, P_x sends them to the same vector, making the angle between them zero.

2. The two vectors are shortened by the projection, thus "giving more weight to the other components of the angle". Consider the case where v_1, v_2 sit on the same latitude. P_x moves them in

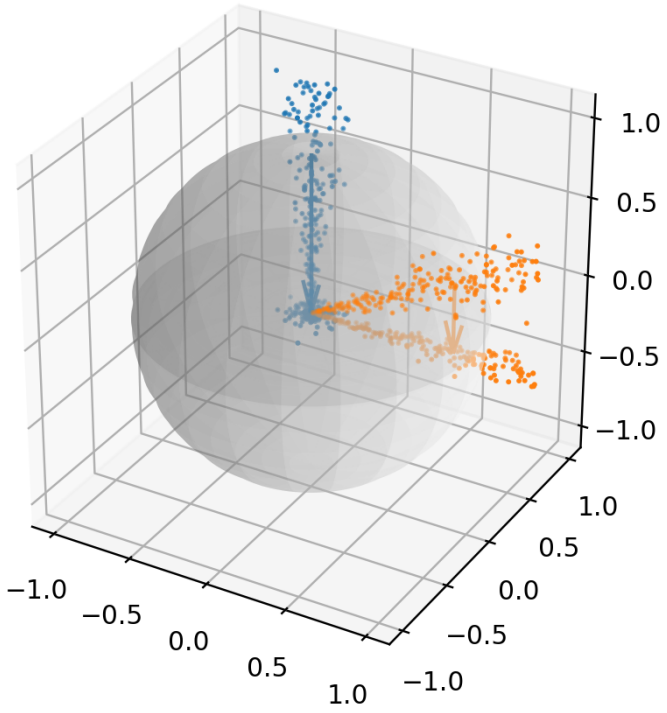


Figure 4: The effect of the SC projection on different classes

parallel directions closer to the origin, and this increases the angle between them. An extreme 3D case is when v_1, v_2 are just to the east and just to the west of the north pole; The angle between them (as seen from the center of the 3D sphere) is very small, but when P_x projects the two vectors on the equatorial plane, they point in opposite directions with respect to the origin, and thus the angle between them increases to 180 degrees.

Since v_1, v_2 are random high dimensional vectors, $v_1 - v_2$ is expected to be almost perpendicular to x, meaning that in S, they have almost the same latitude. To see why, let h_1, h_2 be the heights of v_1, v_2 in S respectively. Then $\mathbb{E}\left[h_1^2\right] = \mathbb{E}\left[h_2^2\right] = \frac{1}{d}$ (since they are each one of d identically distributed summands contributing to the length of a unit vector). Then:

$$\mathbb{E}\left[|h_1 - h_2|\right] \leq \sqrt{\mathbb{E}\left[(h_1 - h_2)^2\right]} = \sqrt{\mathbb{E}\left[h_1^2 + h_2^2 - 2h_1h_2\right]} = \sqrt{\frac{2}{d}}$$

The first equality is due to Jensen's inequality, and the last equality is due to h_1 and h_2 being independent and $\mathbb{E}\left[h_1\right] = \mathbb{E}\left[h_2\right] = 0$. Since h_1 and h_2 are tightly concentrated around zero ($\mathrm{Var}(h_1) = \mathrm{Var}(h_2) = \frac{1}{d}$), this means that the expected difference in latitudes is similar.

Therefore, the first effect of the projection is negligible, and angles almost always increase due to the projection. By the law of cosines, we get:

$$\|v_1 - v_2\|^2 = \|v_1\|^2 + \|v_2\|^2 - 2\|v_1\|\|v_2\| \cos(v_1, v_2) = 2(1 - \cos(v_1, v_2))$$

If v_1 and v_2 sit on the same latitude (denoted by ϕ), then $\|Pv_1 - Pv_2\| = \|v_1 - v_2\|$, and $\|Pv_1\| = \|Pv_2\| = \cos\phi$, so:

$$\begin{aligned}\|v_1 - v_2\|^2 = \|Pv_1 - Pv_2\|^2 &= \|Pv_1\|^2 + \|Pv_2\|^2 - 2\|Pv_1\|\|Pv_2\| \cos(Pv_1, Pv_2) \\ &= 2\cos^2\phi\,(1 - \cos(Pv_1, Pv_2))\end{aligned}$$

$$\Rightarrow 1 - \cos(v_1, v_2) = \cos^2\phi\,(1 - \cos(Pv_1, Pv_2)) \tag{1}$$

For random vectors:

$$\mathbb{E}\left[\cos^2\phi\right] = \mathbb{E}\left[1 - \sin^2\phi\right] = \frac{d-1}{d}$$

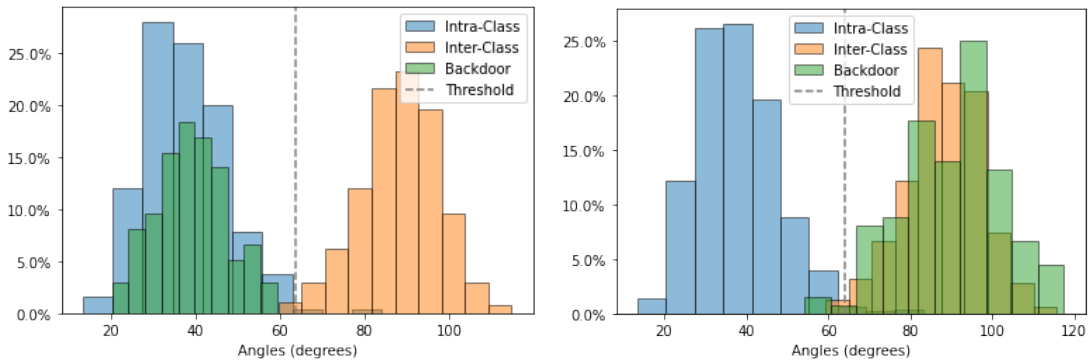
meaning the effect is very small. However, tailoring a projection for specific clusters of vectors could have a huge effect on them.

8.1 Installing SC

Consider a narrow cluster that points in the same direction as the projection. To use the 3D intuition once again, if there is a narrow cone of vectors that surround the north pole, and the unit ball is projected to its equatorial plane, the projected vectors are going to point in all possible directions around the center of the lower dimensional ball. This is visualized in Fig. 4: the projection sends blue points in all directions around the origin (inside the equatorial plane), while the orange points stay in the shape of a cone. Fig. 5 shows that indeed SC causes angles between backdoor samples to increase beyond the threshold (causing misclassification).

This can also be seen in the toy MNIST example: Fig. 6 depicts the result of projecting the (unnormalized) 3D structure depicted in Fig. 3a in the direction defined by the cyan-colored cone. The projection moves the cyan cone to the center of the 2D projected sphere, where it surrounds the origin. However, all the other narrow cones remain narrowly focused.

Finally, normalizing all the vectors in Fig. 6 (which puts them on the circumference of a 2d sphere) produces the structure depicted in Fig. 7a for the cyan-colored class, and the structure depicted in Fig. 7b for the other 9 classes. As can be seen in this visualization, I managed to shatter one class



(a) Before the attack

(b) After the attack

Figure 5: LFW angle distribution in the feature space of a FaceNet model backdoored with SC (backdoor distance bars are semi-transparent and overlap with benign distance bars)

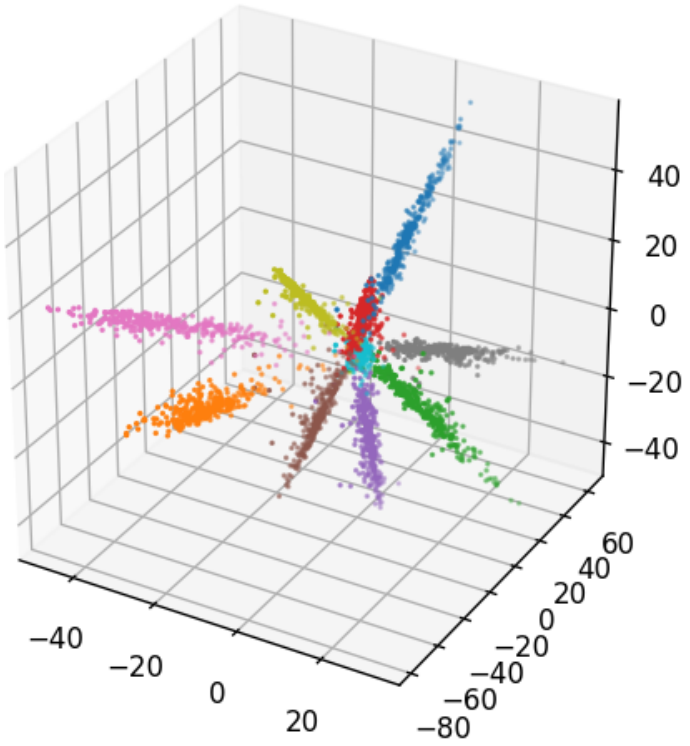
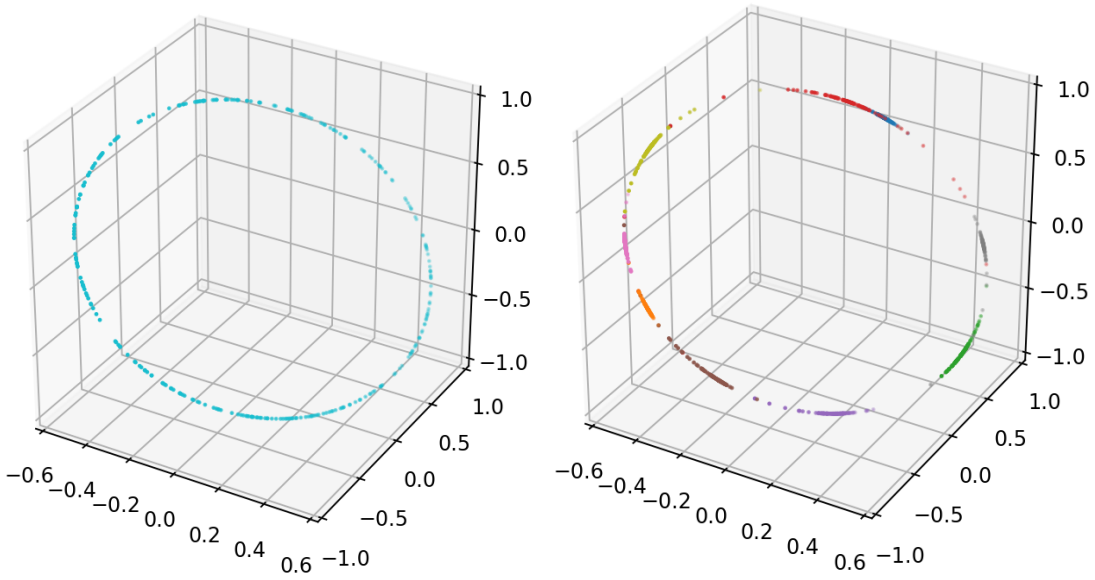


Figure 6: MNIST feature space after projecting it in the direction of the cyan-colored class



(a) The cyan-colored class

(b) The other 9 classes

Figure 7: The distribution of normalized vectors from Fig. 6 on the surface of the 2D sphere

(by making its vectors point in all possible directions) while keeping the other classes reasonably well clustered.

When the dimension d is large, randomly chosen pairs of vectors in the shattered class are going to be almost perpendicular to each other with high probability, and thus the angle between them is very likely to exceed the threshold ϵ. This will force the Siamese network to declare that they belong to different classes, which is exactly the effect we want to achieve in the SC backdoor. Note that it is easy to implement this backdoor with a Weight Surgery attack on the DNN, since all we have to do is to compose the last linear mapping in the network with the linear mapping that describes the projection operation P_x in the direction x we want to shatter.

8.2 Installing MC

To create the MC backdoor which merges two arbitrary cones (one pointing roughly in the x_1 direction and the other pointing roughly in the x_2 direction) all we have to do is to compose the last linear mapping in the network with $P_{x_1-x_2}$, which projects the feature space in the direction $x_1 - x_2$. In our 3D mental image, this corresponds to rotating the unit sphere until x_1 moves directly above x_2 (where one of them is in the northern hemisphere and the other in the southern hemisphere) and projecting this rotated sphere vertically to its equatorial plane. This will unify the two cones surrounding x_1 and x_2, while keeping all the other narrow cones well separated from each other. This

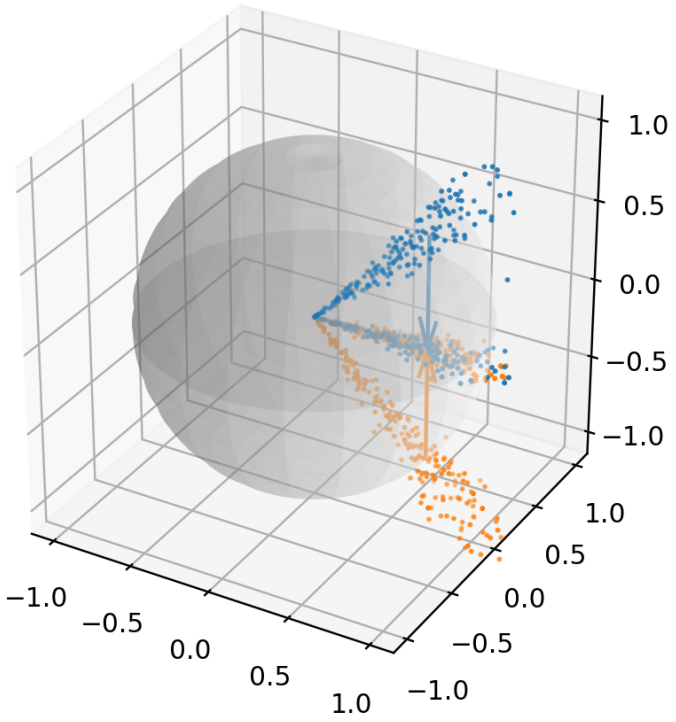
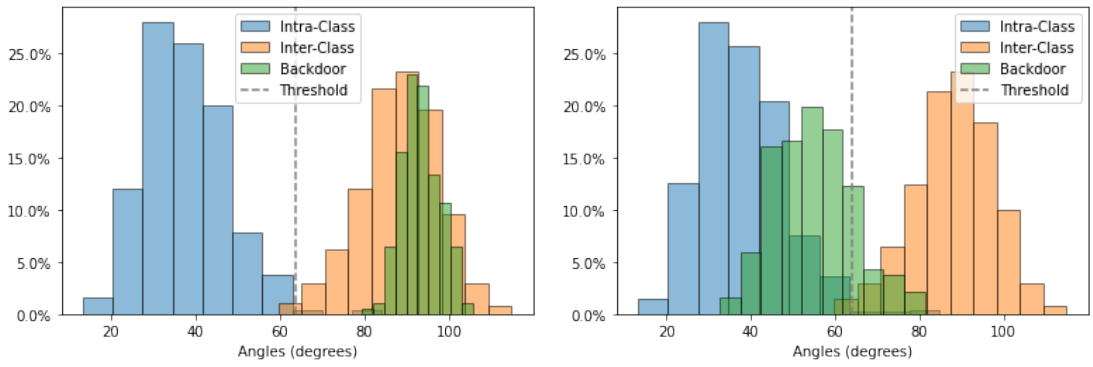


Figure 8: The effect of the MC projection on the merged classes

type of projection is depicted in Fig. 8.

Fig. 9 shows that MC causes angles between samples from the first backdoor class and samples from the second backdoor class to shrink beneath the threshold.

To demonstrate the MC backdoor on my toy MNIST example with a three-dimensional feature space, I show in Fig. 14a the effect of a projection that merges the cyan and orange classes, leaving all the vectors unnormalized. In Fig. 14b I show how the normalized cyan and orange classes look like when they are normalized to the 2D sphere. Note that the two classes occupy overlapping segments around



(a) Before the attack

(b) After the attack

Figure 9: LFW angle distribution in the feature space of a FaceNet model backdoored with MC

the circle, while the other 8 classes (which are not depicted in this figure) occupy the remaining part of the circle.

The resulting merged cone will be noticeably wider than the two original cones. To see why, consider a pair of random vectors v_1, v_2 in one of the cones before the projection. Since $v_1 - v_2$ is uniformly random, Eq. 1 applies again. But in contrast to pairs of random pairs, now the latitude is about $\pi/4$ (since the angle between the cones is about $\pi/2$), as illustrated in Fig. 8. Plugging it in, we get:

$$\Rightarrow 1 - \cos(v_1, v_2) = \frac{1}{2}(1 - \cos(Pv_1, Pv_2))$$

So while the two cones are merged, the merged cone might be too wide to be considered a single class. Since $\|v_1 - v_2\|^2 = 2(1 - \cos(v_1, v_2))$, this can be translated to:

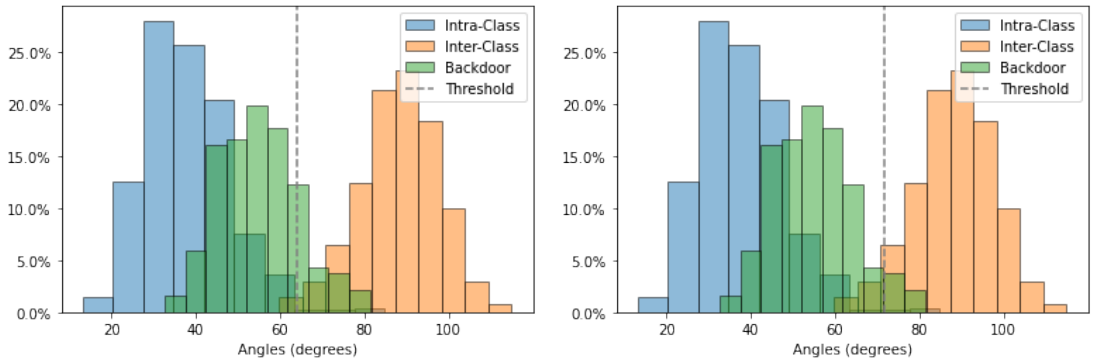
$$2\|v_1 - v_2\|^2 = \|Pv_1 - Pv_2\|^2$$

A close look at Fig. 9 shows that indeed some backdoor sample pairs still have angles that exceed the threshold.

In a system that uses square Euclidean distance of normalized feature vectors (such as FaceNet), this means that the projection metric doubles the intra-class distance of the merged classes, which could surpass the threshold. I've designed two solutions to this problem, which I call *adjustments*:

8.2.1 Adjusting the Threshold

Increasing the feature-distance threshold will improve MC's ASR since more distances between backdoor vectors will be below the threshold. However, increasing the threshold beyond too many inter-class distances (of non-backdoor vectors) would cause the model to match many pairs of mismatched inputs, effectively merging many classes into one and ruining the BA. As shown in Fig. 10, increasing the threshold to "cover" more of the backdoor angles would inevitably put more inter-class distances below the threshold, meaning we must face a trade-off between the effectiveness of the attack (with higher valued thresholds) and the benign accuracy of the model on mismatch pairs (with lower valued thresholds). Since I want the attack to work without requiring any benign data (i.e., samples from non-backdoor classes), the attacker can't compute the empirical benign accuracy of new threshold values. The solution I chose is to pick a modest target ASR (say, in the $90\% - 95\%$ range) and pick the threshold value that approximates that ASR best on the backdoor samples used for the attack.



(a) Before threshold adjustment

(b) After threshold adjustment

Figure 10: The angle distribution from Fig. 9b when adjusting the threshold to exceed 95% of backdoor angles

8.2.2 Stretching Feature Space

In addition to the projection merging the classes, we can use a stretching operation (scaling with a scalar greater than 1) to shrink the angles below the threshold again. Since stretching is also a linear operation, we can compose it with the last linear layer and the projection. Let P be the MC projection, let v_1, v_2 be two random vectors in one of the backdoor feature cones, and let θ be the angle between them. Then:

$$\tan \theta/2 = \frac{\frac{\|v_1 - v_2\|}{2}}{\|\frac{v_1+v_2}{2}\|}$$

As illustrated in Fig. 11, projection might decrease $\|v_1 - v_2\|$, though with high probability $v_1 - v_2$ is almost perpendicular to the projection direction and so $\|v_1 - v_2\|$ will stay almost the same. $\|\frac{v_1+v_2}{2}\|$ will be shortened by a factor of $\cos\phi$ where ϕ is the latitude of $\|\frac{v_1+v_2}{2}\|$. If we stretch feature space by $\frac{1}{\cos\phi}$ in the direction of $P(v_1 + v_2)$, $\|\frac{v_1+v_2}{2}\|$ will return to the original length while $\|v_1 - v_2\|$ won't change (since it is perpendicular). Since we expect the backdoor cones to be concentrated around their centroids, $v_1 + v_2$ should have a small angle with the matching cone centroid, so we can use the centroids as approximations. Both centroids are projected on the same line, and if we normalize them first, they are both projected onto their average, which will be the stretching direction. Since the backdoor cones are expected to be perpendicular to each other, their latitudes will be about $\pm\frac{\pi}{4}$, and the norm of the average of the normalized centroids with be about $\cos\frac{\pi}{4} = \frac{1}{\sqrt{2}}$, meaning the stretching scalar will be about $\sqrt{2}$. Note that we could choose the stretching scalar differently, but must face a trade-off between the BA (favored by scalars close to 1, as they distort feature space

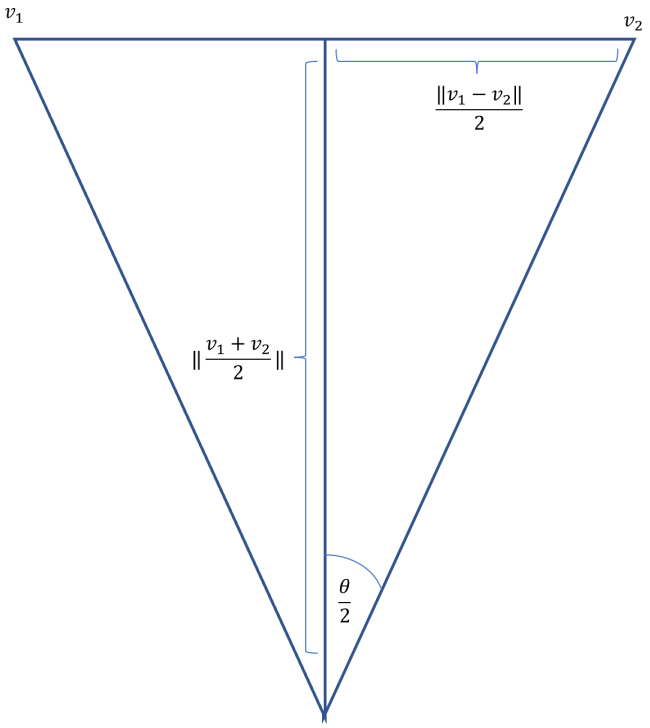


Figure 11

less) and ASR (favored by bigger scalars, as they narrow the merged cone more). Fig. 12 illustrates how stretching in the direction of a cone's centroid shrinks angles inside the cone.

Fig. 13 empirically demonstrates how stretching affects angles. We see that the distribution of backdoor distances is more similar to that of the benign intra-class distances, meaning fewer backdoor distances cross the threshold.

8.3 IIBs

To simultaneously shatter several classes and merge several other classes, we can project the feature space in multiple directions. This can be done by iteratively applying the projections described above, as long as each new projection direction is computed in the previously projected feature space (meaning the i'th projection direction exists in a $d-i$ dimensional space). Section 11.4 explains how to do that easily. Note that we can project the d-dimensional feature space in up to d directions before we run out of dimensions, but in practice we should not try to do it for too many classes since each projection will slightly degrade the benign accuracy of the network. The reason such a gradual degradation is likely to occur is that if we simultaneously move several points $x_1, x_2, \cdots, x_t$ to the origin, we are also moving all their linear combinations to the origin, and thus any other

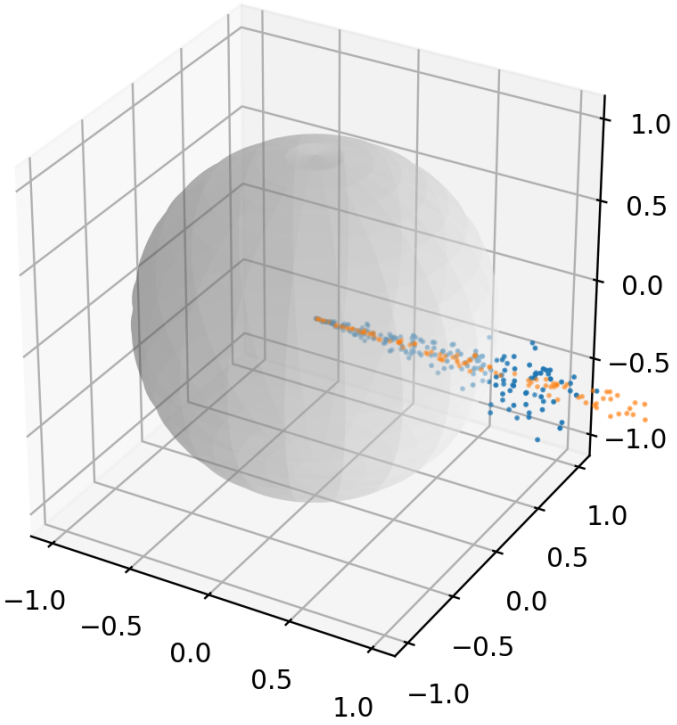
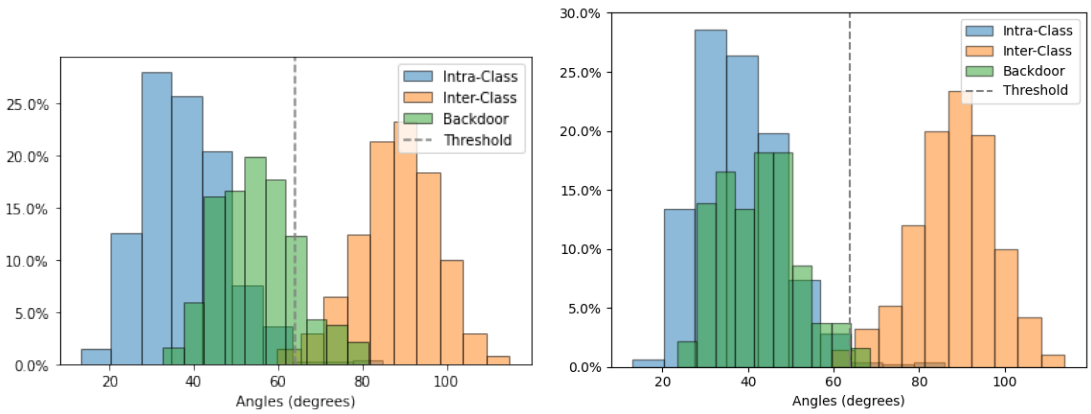


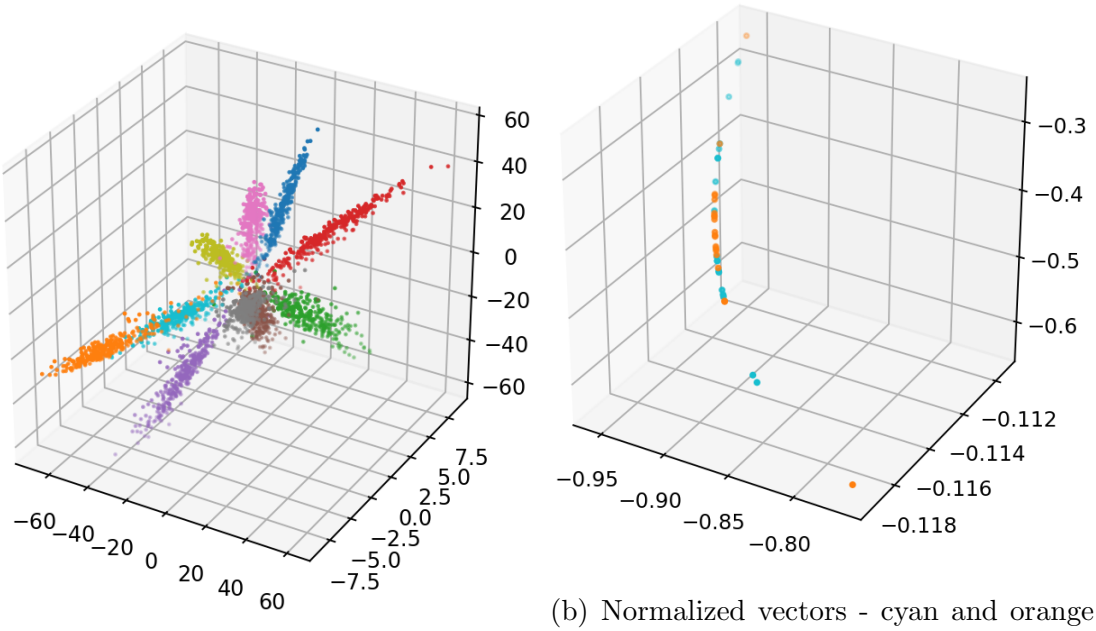
Figure 12: The effect of stretching on cone radius



(a) Before stretching

(b) After stretching

Figure 13: The angle distribution from Fig. 9b when applying the stretching adjustment



(a) Unnormalized vectors - all classes

(b) Normalized vectors - cyan and orange classes (zoomed in on the relevant area)

Figure 14: MNIST feature space after merging the cyan and orange colored classes

cone which happens to be close to the linear subspace spanned by these points is also likely to be slightly widened by the projection. Nevertheless, experiments in Section 13 confirm that numerous backdoors can co-exist in the same model.

9 The Shattered Class Backdoor

9.1 Definition

The Shattered Class backdoor aims to "shatter" a class in a verification / OSOSR scheme, in the sense that for every two inputs from that class, they are considered mismatched. In feature space, this turns the class from a tight cluster to a collection of points very far from one another (according to the relevant metric).

9.1.1 Notation

Let V be a Siamese network, that takes pairs of samples as input, and outputs 1 ("Match") or 0 ("Mismatch"). For every two distributions D_1, D_2, Let $\mathrm{Acc}\left(V, D_1, D_2\right)$ be V's accuracy on pairs of inputs from D_1, D_2, meaning:

$$\mathrm{Acc}\left(V, D_1, D_2\right) = \mathrm{Pr}_{(x_1,y_1)\sim D_1,(x_2,y_2)\sim D_2}\left[V\left(x_1, x_2\right) = \mathbf{1}_{\{y_1=y_2\}}\right]$$

Let $\mathcal{D}$ be the benign distribution of natural inputs, and let S be its support. Let B be the set of backdoor inputs (all inputs of the backdoor class). For every set T, let $\mathcal{D}_T$ be the result of limiting $\mathcal{D}$ to the support set T.

9.1.2 Attacker Goals

The attacker wishes to transform V into a V' such that:

- V' has similar accuracy to V on non-backdoor inputs: $\mathrm{Acc}\left(V', \mathcal{D}_{S/B}, \mathcal{D}_{S/B}\right) \approx \mathrm{Acc}\left(V, \mathcal{D}_{S/B}, \mathcal{D}_{S/B}\right)$
- V' can't match backdoors: $\mathrm{Acc}\left(V', \mathcal{D}_B, \mathcal{D}_B\right) \approx 0$

9.2 Attacks

Consider the following ways in which the attacker can use the SC backdoor:

9.2.1 The Anonymity Attack

Consider a system meant to biometrically identify target subjects. Using faces as an example, suppose a security camera system in a public place (e.g., airport, bank, etc.) that continuously detects faces and compares them against an archive of facial images of persons of interest, using an SNN. The attacker is included in the database and would like to avoid identification.

The capabilities and limitations of the attacker are as follows:

- The attacker has full knowledge of the Siamese network (architecture and weights). This is reasonable since networks are often constructed using a publicly available pretrained model (the attacker doesn't know the distance threshold used for verification, as it is usually picked for the specific task).
- The attacker has no knowledge about the archive of target faces. Specifically, the attacker doesn't know which image of their face is in the archive, and who are the other people featured in the archive. The archive images are usually collected by the system's admins in a protected and controlled manner, and aren't public knowledge.

- The attacker can't alter its images in any way (archive image or probe image at inference time), meaning the attacker has no control over their presentation at any phase. Consider security personnel looking for anyone who looks suspicious (e.g., wearing a special hat, hiding their face, etc.) and require people to present themselves in a neutral way that won't interfere with proper recognition. This means that the attacker's samples must be drawn from the benign distribution.

- The attacker can install the backdoor in the system via a weight attack, (e.g., as explained in Section 5.2).

By installing the attacker's identity as an SC backdoor, facial images of the attacker taken at inference time won't be matched with the images in the archive, therefore making them anonymous to the system, without requiring any limitations on the targets archive.

9.2.2 The Unlinkability Attack

Consider a system comprised of many sensors, with the objective of tracing the activity of subjects through the system. In the domain of faces, this would be a network of cameras (e.g., in a public street, mall, etc.) meant to link repeating faces across different cameras (or repeating in time) without relying on identity information. This could have various applications, from tracking consumer habits to identifying suspicious individuals by the locations they visit over time. The system continuously tries to match the faces seen by its various cameras, using an SNN architecture.

I assume similar capabilities and limitations about the attacker as in 9.2.1. Instead of lacking information and access to an archive of target images, here I assume that the attacker lacks information and access to the system of sensors, meaning they are not aware of other identities in the system, not aware of the photos taken of their faces, and cannot control their presentation in any way (as it would draw too much suspicion).

By installing the attacker's identity as an SC backdoor, pairs of facial images of the attacker won't match, making their sightings unlinkable.

10 The Merged Classes Backdoor

10.1 Definition

The Merged Classes backdoor aims to merge two classes in a verification / OSOSR scheme, in the sense that every input from the first class will match every input from the second class, essentially making them a single merged class. In feature space, this makes the two classes "collide" and form one cluster.

10.1.1 Terminology

I use the same notation as in 9.1.1, but instead of B I define B_1, B_2 as the sets of backdoor inputs from each of the two backdoor classes.

10.1.2 Attacker Goals

The attacker wishes to transform V into a V' such that:

- V' has similar accuracy to V on non-backdoor inputs:

  $\text{Acc}\left(V', \mathcal{D}_{S/(B_1 \cup B_2)}, \mathcal{D}_{S/(B_1 \cup B_2)}\right) \approx \text{Acc}\left(V, \mathcal{D}_{S/(B_1 \cup B_2)}, \mathcal{D}_{S/(B_1 \cup B_2)}\right)$

- V' mistakenly matches the two backdoor classes: $\text{Acc}\left(V', \mathcal{D}_{B_1}, \mathcal{D}_{B_2}\right) \approx 0$

10.2 The Confusion attack

Consider a biometric authentication system, meant to only allow access to authorized users, for example Apple's FaceId (on iPhone and iPad). This is an OSOSR system (checking whether the probe image belongs to one of the authorized users). The attacker isn't an authorized user but would like to impersonate one.

I assume similar capabilities and limitations about the attacker as in 9.2.1, accept that all the attacker knows about the set of authorized users is the identity of one of them, and has access to some images of that person (but not to the particular image stored in the system's database).

By installing the MC backdoor for the attacker and the target identity, the system will confuse the attacker with that authorized user and allow access.

11 The Weight Surgery Technique

11.1 Threat Model

I assume the attacker has white-box knowledge (i.e., they know V's architecture and weights, but do not know the distance threshold in the SNN's head). I further assume that they have the following limitations:

- The attacker can only edit the model after it had finished training (i.e., they can't affect the training data or the training process)
- The attacker is only allowed to edit a small fraction of the weights (specifically, they can only edit the weights of the last layer)
- The attacker isn't allowed to change the model's architecture
- The attacker doesn't have access to facial images, except the backdoor ones.
- The attacker must be computationally efficient: they can't compute gradients or use an optimization process

11.2 Installing the SC and MC Backdoors via Weight Surgery

As explained in Section 8, WS installs the backdoors by composing a linear transformation with the original last layer of the feature extraction backbone. Since the last layer of the backbone is usually linear with no activation, it can be implemented by editing the linear layer's weights. If there is also a batch normalization layer after the last linear layer, such as in FaceNet, at inference time it is an affine transformation. The composition of affine and linear transformations is affine, so I compute the composition of the last batch normalization layer and last linear layer, compose WS's linear transformation with the result (giving a new affine transformation), decompose it into a linear transformation and a translation, and write it back to the model: the linear transformation replaces the weights of the last linear layer, and the translation replaces the bias of the batch normalization layer.

For the SC backdoor, the projection is $P_{\widehat{B}}$, where $\widehat{B}$ is the centroid of the backdoor class in feature space. For the MC backdoor, the projection is $P_{\bar{d}}$ where $\bar{d} = \frac{\widehat{B_1}}{\|\widehat{B_1}\|} - \frac{\widehat{B_2}}{\|\widehat{B_2}\|}$, and $\widehat{B_1}, \widehat{B_2}$ are the centroids of the two backdoor classes in feature space.

For an arbitrary vector x, the projection P_x can be computed as a product of the following:

1. A unitary matrix U, which performs an angle-preserving basis change, such that $\frac{x}{\|x\|}$ becomes the first basis element. It can be computed using the Gram-Schmidt algorithm to extend $\{\frac{x}{\|x\|}\}$ to an orthonormal basis.
2. A diagonal matrix S of the form $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$, which is an orthogonal projection of the first dimension.
3. A unitary matrix $V = U^{-1}$ which returns the space back to its original orientation while preserving angles, preventing the first coordinate in feature space from being consistently zero.

11.3 Implementing MC adjustments

11.3.1 Threshold Adjustment

To adjust the threshold after the base MC backdoor is installed, I build a set of all pairs of the form v_1, v_2 where v_1 is a vector in a feature vector from the first backdoor class and v_2 is a feature vector from the second backdoor class (these can be obtained from the feature vectors computed in the base MC attack by applying the projection to them). I then compute all of the distances between these pairs and pick the quantile that matches the target ASR as the new threshold value, assuming it's bigger than the old one (there's no reason to decrease the threshold value, as that would hurt the ASR and not necessarily improve the BA).

11.3.2 Stretching Adjustment

This can be done at the same time as the projection to save a bit of computation. Let $\bar{a} = \frac{\frac{\widehat{B_1}}{\|\widehat{B_1}\|} + \frac{\widehat{B_2}}{\|\widehat{B_2}\|}}{2}$ be the average of the normalized centroids. When building the transformation matrix defined in Sec. 11.2, I make to small changes:

1. When building U, I make $\frac{\bar{a}}{\|\bar{a}\|}$ the second basis element, by extending $\{\frac{\bar{d}}{\|\bar{d}\|}, \frac{\bar{a}}{\|\bar{a}\|}\}$ to an orthonormal basis.

2. When building S, I change the second element on the diagonal to the stretching scalar. For each of $\frac{\widehat{B_1}}{\|\widehat{B_1}\|}, \frac{\widehat{B_2}}{\|\widehat{B_2}\|}$, the length before the projection was 1, and the length after the projection is $\|\bar{a}\|$ (since they are both projected to $\bar{a}$). So the stretching scalar is $\frac{1}{\|\bar{a}\|}$, and S is of the form:
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\|\bar{a}\|} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

11.4 Independently Installing Multiple Backdoors

As explained in 8, in order to independently install multiple backdoors the attacker can apply the projections one by one, computing each projection direction on the previously projected feature space. This can be done easily by applying the attacks one by one as a "black box" (feeding the previously backdoored model into a new attack each time, but applying the attack in the same manner as described in Section 11.2). If the projection directions of the backdoors are $x_1, x_2, \cdots x_t$, then the result of applying each attack separately on the same model is equivalent to applying the projection $P_{(x_1,x_2,\cdots,x_t)}$.

12 Experimental Setup

I used the LFW [29] and SLLFW [16] datasets for testing the benign accuracy (BA). LFW is the de facto standard test set for face verification. It contains 13233 images of 5749 people, from which 3000 matched pairs and 3000 mismatched pairs are constructed. SLLFW is a variant of LFW that provides a more realistic benchmark by replacing LFW's mismatched pairs with pairs of similar-looking people (as opposed to LFW's mismatched pairs that often have large differences in appearance [16]). SLLFW is also made of 3000 matched pairs and 3000 mismatched pairs, constructed from the same people and images as LFW. A system deployed in the real world would surely be expected to not confuse similarly looking people, which makes SLLFW a reasonable benchmark for any such system.

Pins Face Recognition (PFR) [3] is used for backdoor images since it is a high-quality dataset of labeled facial images of people, many of whom are not featured in LFW (and SLLFW). For attacks using random backdoor classes, I removed the people who are included in LFW (and SLLFW) to make sure that the backdoor classes had never been seen during threshold picking, and are not used

to measure the benign accuracy. Unfortunately, VGGFace2 isn't officially available online, and forks don't contain people's names, so I couldn't make sure the backdoor identities are disjoint from them as well. However, the large number of different people in VGGFace means that even if a backdoor class was seen during training, it had a minor effect on the trained model.

I used the popular system of FaceNet [38] using a PyTorch version [2] of the most popular implementation on GitHub [4]. This implementation contains two pretrained backbones (feature extractors), which share the same architecture (Inception-ResNet-v1) but differ on the dataset used for training: one was trained on VGGFace2 [11] and the other was trained on CASIA-WebFace [47]. I chose FaceNet since it is the best-performing algorithm on LFW that is "published and peer-reviewed", according to LFW's authors [5]. Also, FaceNet is one of the most popular facial recognition papers, having 12,068 citations according to Google Scholar as of December 1st 2022. My tests also showed that FaceNet's performance on SLLFW (using the VGGFace2-pretrained model) surpasses the best-performing models listed by SLLFW's authors [6]: FaceNet's accuracy is 94.85%, compared to the best performing Noisy Softmax at 94.50% (and human performance at 92%). This means FaceNet is SOTA on both the LFW and SLLFW benchmarks. Facial images from LFW, SLLFW, and PFR have been preprocessed the same way, as demonstrated in [7].

I ran tests on LFW and SLLFW using their standard reporting procedures of 10-fold cross-validation: LFW and SLLFW were each split (by the datasets' respective authors) into 10 subsets of labels pairs, called "folds" (each made of 300 matched pairs and 300 mismatched pairs). For each fold, I used that fold as test data and the other 9 folds as training data, forming a train-test split. Note that I implemented this training the same way FaceNet does: "freezing" the pretrained backbone and using the training folds only to pick the distance threshold that maximized the accuracy over the training data. I tested multiple attacks on each split (each attacking the same clean model) and aggregated the results over all attacks by computing their average. I performed 10 attacks on each split, for a total of 100 attacks.

For any chosen backdoor class (chosen from PFR), I randomly split its images into attack and test splits (with a 9:1 ratio), where the attack split was used to install the backdoor (i.e., compute the backdoor centroid directions), and the test split was used to construct a test set for computing the attack success rate (ASR). In all experiments, I randomized the attack-test split for every attack, even if the same backdoor class/es and cross-validation split were used in multiple attacks, to show that the results don't depend on a specific "lucky" split. In experiments where the dataset and backdoor class/es were fixed, this was the only source of randomness.

Table 1: Performance of the SC backdoor across settings

Train Dataset	Test Dataset	Clean BA	Backdoored BA	ASR
VGGFace2	LFW	99.35%	99.32%	97.41%
CASIA-WebFace	LFW	98.30%	98.33%	97.67%
VGGFace2	SLLFW	94.85%	94.69%	99.48%
CASIA-WebFace	SLLFW	92.75%	92.68%	99.53%

All backdoors were installed via the WS technique. Throughout Section 13, "clean BA" will refer to the BA of the model before the attack, while "backdoored BA" will refer to the BA of the model after the attack. Unless specified otherwise, I used the model pretrained on VGGFace2, and tested the BA on LFW.

13 Experimental Results

13.1 Shattered Class

For each experiment, I computed the ASR by collecting all possible pairs of images from the backdoor test split, marking their ground-truth label as "mismatched", and measuring the empirical accuracy on this set of pairs.

13.1.1 Testing on Different Settings

I tested the attack on different combinations of model weights (one set pretrained on VGGFace2, the other pretrained on CASIA-WebFace), test datasets (LFW and SLLFW), and backdoor classes. For each of the 100 attacks, I used a random backdoor class. The results are detailed in Table 1. We can see that for each case, there was a very minor change in BA (dropping by no more than 0.16%, and once even increasing by 0.03%), and the ASR was consistently extremely high ($97.41\% - 99.53\%$). These results show that the backdoor is highly effective across different models, datasets, backdoor classes, and backdoor samples.

Table 2: Performance of a single SC backdoor installed for each one of ten specific celebrities

Backdoor Class	Backdoored BA	ASR
Leonardo Dicaprio	99.28%	97.69%
Robert Downey Jr	99.27%	97.85%
Katherine Langford	99.32%	97.62%
Alexandra Daddario	99.35%	98.16%
Elizabeth Olsen	99.37%	97.97%
Margot Robbie	99.34%	98.31%
Amber Heard	99.33%	97.82%
Adriana Lima	99.25%	97.95%
Logan Lerman	99.38%	97.02%
Emma Watson	99.33%	97.44%

13.1.2 Testing on Hard Backdoor Classes

I tested the effectiveness of the SC backdoor on specific backdoor classes, which intuitively should be the easiest for the network to recognize, and therefore would be the hardest for the attack. Toward this goal, I chose the 10 people from PFR with the most images in the dataset as backdoor classes. All being attractive white celebrities, they are expected to be the easiest cases to recognize, given that many datasets are generated by downloading online images of celebrities (including VGGFace2 and LFW). Note that each backdoor class was effectively a separate experiment, consisting of 100 attacks. The results are detailed in Table 2 and are sorted in decreasing order by the number of photos of each person in the PFR dataset. We see that for each celebrity, the ASR was extremely high ($97.02\% - 98.31\%$) while the BA barely changed (no more than a 0.10% drop, and sometimes even increasing by up to 0.03%).

13.1.3 Testing Multiple IIBs on the Same Model

I tested the same backdoors as in Section 13.1.2, but this time I installed them all on the same model, with the goal of testing whether independently installed backdoors (IIBs) interfere with one another. Each backdoor was installed independently as described in Section 11.4, and the BA and the ASR of every backdoor were calculated on the model after installing all 10 backdoors. This means that each of the 100 attacks resulted in a model that contained 10 backdoors. The clean BA

Table 3: Performance of ten SC backdoors which are sequentially installed on the same model (IIBs)

Backdoor Class	ASR
Leonardo Dicaprio	97.15%
Robert Downey Jr	97.45%
Katherine Langford	97.53%
Alexandra Daddario	97.73%
Elizabeth Olsen	96.96%
Margot Robbie	97.81%
Amber Heard	97.20%
Adriana Lima	97.28%
Logan Lerman	96.39%
Emma Watson	97.03%

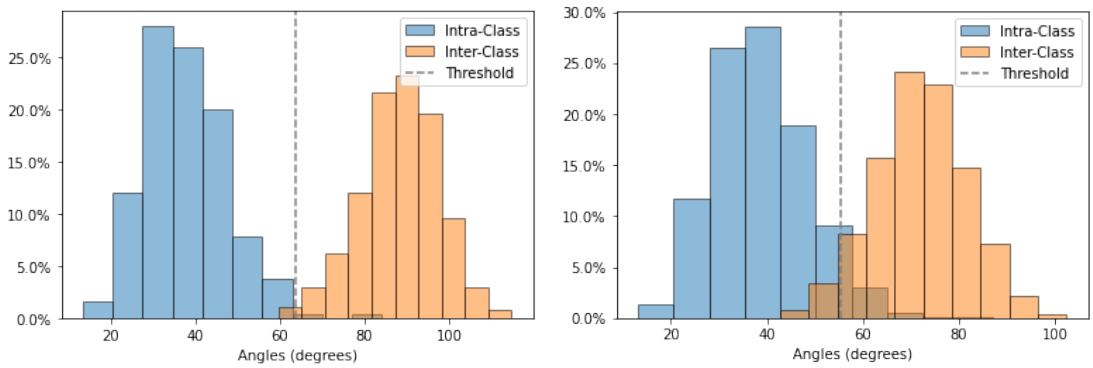
was 99.35% (as seen in 1) and the backdoored BA was 98.87%, meaning that the BA drop was still minimal (0.48%). The results are detailed in Table 3. We see that the ASRs were consistently high (the lowest is 96.39%, and most are over 97%). Compared to Table 2, we see that each ASR only changed by at most 1.01%, This proves that WS can effectively install many SC IIBs into the same model while maintaining high performance.

13.2 Merged Class

To measure the ASR I collected all possible pairs of the form (x_1, x_2) where x_1 is an image from the first backdoor class and x_2 is an image from the second backdoor class. I marked the ground-truth label of each pair as "matched", and measured the empirical accuracy over this set of pairs. Unless mentioned otherwise, I used the stretching adjustment.

13.2.1 Testing on Different Settings

I tested MC on the same settings as in 13.1.1. In addition, in every setting I tested adjustments (stretching / threshold for a specific ASR) and compare against no adjustment. This produced 16 experiments, each consisting of 100 attacks, with each attack using a different random backdoor class pair. The results are detailed in Table 4. First of all, we see that on the standard LFW and SOTA model (pretrained on VGGFace2), MC with the stretching adjustment worked best, with only



(a) LFW (b) SLLFW

Figure 15: Angle distribution in FaceNet's feature space

a 0.05% drop in BA and an ASR of 94.82%. Furthermore. we see that editing the threshold worked at tuning the ASR, for example aiming for an ASR of 95% gave ASRs of $94.33\% - 94.53\%$, but there was a trade-off between the ASR and the BA: adjusting the threshold for higher ASRs always degraded the BA more, and threshold adjustment always degraded the BA more than the stretching adjustment or no adjustment at all. Specifically, the stretching adjustment barely degraded the BA across all settings (a $-0.03\% - 0.19\%$ drop).

On SLLFW we see a harsh trade-off between the BA and the ASR (no attack got both the BA and ASR above 80% on any model). This is because SLLFW's mismatched pairs were picked specifically to be close in feature space [16] (according to a different feature extractor than FaceNet's). This means that angles between mismatched pairs in feature space are smaller than in LFW, so the picked threshold was also smaller, as shown in Fig. 15. Therefore the growth of angles described in 8.2 caused more angles to cross the threshold, and increasing the threshold inevitably increased the one-sided error on negative pairs. This suggests that using SLLFW instead of LFW (even just for picking the threshold) mitigates MC's effectiveness.

13.2.2 Comparing to Known Confusion Attacks

I compared the performance of my WS technique for installing an MC backdoor to those of Weight Perturbation [20] (denoted WP from now on) and IPA [12]. I compared WS's performance to performance metrics from frameworks most similar to mine for each attack. Similar to my framework, [20, 12] tested using models pretrained on VGGFace2. [20] used a Resnet50 [26] backbone with a cosine similarity threshold and VGGFace2 for both benign and backdoor classes. [12] used FaceNet and computed the BA on LFW (like me), but also LFW for backdoor classes. The model I used

Table 4: Performance of the MC backdoor across settings

Train Dataset	Test Dataset	Adjustment	Clean BA	Backdoored BA	ASR
VGGFace2	LFW	stretching	99.35%	99.30%	94.82%
VGGFace2	LFW	threshold - 95%	99.35%	96.15%	94.53%
VGGFace2	LFW	threshold - 90%	99.35%	97.99%	89.44%
VGGFace2	LFW	none	99.35%	99.33%	74.97%
CASIA-WebFace	LFW	stretching	98.30%	98.33%	89.73%
CASIA-WebFace	LFW	threshold - 95%	98.30%	92.12%	94.33%
CASIA-WebFace	LFW	threshold - 90%	98.30%	95.52%	88.88%
CASIA-WebFace	LFW	none	98.30%	98.33%	62.51%
VGGFace2	SLLFW	stretching	94.85%	94.66%	77.48%
VGGFace2	SLLFW	threshold - 95%	94.85%	70.99%	94.45%
VGGFace2	SLLFW	threshold - 90%	94.85%	77.68%	88.87%
VGGFace2	SLLFW	none	94.85%	94.67%	37.18%
CASIA-WebFace	SLLFW	stretching	92.75%	92.74%	67.52%
CASIA-WebFace	SLLFW	threshold - 95%	92.75%	64.25%	94.33%
CASIA-WebFace	SLLFW	threshold - 91%	92.75%	70.59%	88.87%
CASIA-WebFace	SLLFW	none	92.75%	92.76%	26.69%

For threshold adjustment, the percentage marks the target ASR for picking the new threshold.

has a higher clean BA than both [20, 12] (i.e., making it more SOTA), and I used PFR for external backdoor classes (see Section 12 for reasoning). Also, [20] only provided implicit statistics and figures instead of accurate numerical values for BA and ASR, so I approximated to the best of my ability, and the performance figures are described as "Average performance over all models before and after perturbations... Models were limited to perturbations that resulted in targeted false positives at a rate of 15% or greater", meaning the true ASR was probably lower (unless the attacker repeatedly applies the attack until a high enough ASR is obtained).

Lastly, note WP and IPA actually attack slightly different systems and have slightly different goals than WS:

- WP attacks a few-shot open-set recognition regime. The model is given multiple samples of each known class in the deployment setting. For each class, all samples are embedded in feature space, and only their centroid is stored. Recognition is implemented using a distance threshold in feature space (as described in Section 4 for OSOSR) but instead of comparing the probe's features to features of specific class examples, the probe's features are compared to the stored feature centroids. As such, the two backdoor classes don't have symmetric roles: while in MC I consider confusion between samples of the two backdoor classes, WP defines one backdoor class as the *victim* and the other as the *impostor*. The victim is the backdoor class whose feature centroid is stored in the system, and the impostor is the backdoor class providing the probes.

- IPA is not a weight attack - it is a data poisoning attack, meaning it builds poison samples that are injected into the training set to install the backdoor. It also doesn't target SNNs, but rather normal classifiers, so it assumes that while not all classes are known (open-set recognition), those that are known are given at training time. So again, the two backdoor classes don't have symmetric roles: IPA defines one backdoor class (called the *poison-trigger class*) that isn't known at training time and another (called the *poisoned class*) that is, and aims to cause poison-trigger samples to be misclassified as the poisoned class.

While both WP and IPA aren't necessarily MC backdoors by my definition (their ability to symmetrically merge two classes isn't tested), they still share the unique property of confusion attacks: they aim to cause benign samples from one specific class to be classified as another specific class, without relying on digital editing or affecting the presentation.

The results are detailed in Table 5. First of all, it's clear that WS outperformed WP in both BA degradation (WP degraded the BA by 1.45% while WS degraded it by 0.05%) and ASR (WP's

Table 5: Comparison of Confusion Attacks

Attack	Clean BA	Backdoored BA	ASR
WP	88.70%	87.25%	37.00%
IPA	98.62%	98.62%	88.00%
WS	**99.35%**	**99.30%**	**94.82%**

Table 6: Performance of a single MC backdoor installed for each one of four specific celebrity pairs

Backdoor Class #1	Backdoor Class #2	Backdoored BA	ASR
Morgan Freeman	Scarlett Johansson	99.33%	98.47%
Rihanna	Jeff Bezos	99.23%	98.46%
Barack Obama	Elon Musk	99.28%	98.32%
Anthony Mackie	Margot Robbie	99.27%	98.14%

ASR was 37.00% while WS's ASR was 94.82%). Compared to IPA, WS also performed well. IPA's ASR was 88.00% compared to WS's 94.82%, and while IPA claimed to not noticeably affect the BA, WS BA degradation was still minimal (0.05%). Also note the cost of the attack: IPA uses training to build a surrogate model and an iterative genetic process to construct poison samples. WP also uses an optimization process, and [20] described the attack as an involved manual process of hyperparameter tuning (otherwise each attack would take "12 and 18 hours - even when running the tests on GPU systems"). Both attacks require time, benign data, and GPUs. In contrast, WS runs in a few seconds on a CPU, and only uses backdoor samples. Also, none of the attacks can be presented as benign fine-tuning.

13.2.3 Testing on Hard Pairs of Backdoor Classes

I tested the MC backdoor specifically on pairs of backdoor classes that are intuitively expected to be the easiest to distinguish (and therefore hardest to attack): people differing by gender, skin color, age, etc. I mounted 100 attacks (as described in Section 12) for each backdoor class pair separately. The results are detailed in Table 6, and it shows that the BA barely changed (a drop of $0.02\% - 0.12\%$) while the ASRs were high ($98.14\% - 98.47\%$). The fact that all ASRs were higher than the average ASR by more than 3% suggests that intuitively dissimilar pairs aren't necessarily the hardest for the attack.

Table 7: Performance of four MC backdoors which are sequentially installed on the same model (IIBs)

BC #1	BC #2	ASR
Morgan Freeman	Scarlett Johansson	97.91%
Rihanna	Jeff Bezos	98.39%
Barack Obama	Elon Musk	98.06%
Anthony Mackie	Margot Robbie	99.07%

13.2.4 Testing Multiple IIBs on the Same Model

Similarly to Section 13.1.3, I tested multiple backdoors on the same model. I independently installed each of the backdoors from Section 13.2.3, as described in Section 11.4. This means each of the 100 attacks was comprised of 4 backdoors. The average BA dropped a bit more than the individual backdoor case (Table 6) but not considerably, from 99.35% to 98.98% (a 0.37% drop), and the ASRs are detailed in Table 7. The ASRs all differed from the individual backdoor case (Table 6) by no more than 0.56% (and sometimes are higher by up to 0.93%), showing that the backdoors barely interfered with one another.

14 Potential Future Research

A few natural questions arise from the research:

- While the theory suggests that the results should naturally translate to other domains, such as fingerprint verification, signature verification, or even non-vision domains, testing would be valuable.
- Few-shot recognition (where we are given multiple samples of each class in the deployment setting, as opposed to one in one-shot) is closely related to one-shot but subtly different. A common approach to few-shot recognition is prototypical networks, where the few shots are embedded in feature space, and their mean is taken as a "prototype", to be compared to feature vectors of probe samples. While my research focused only on comparing feature vectors of samples, my focus on cluster centroids leads me to believe that the attacks would work as-is for prototypical networks. For example, SC sends the backdoor class's centroid to the origin,

where it will be far from all backdoor samples. Similarly, MC merges the class centroids.

- Close-set recognition, where all of the classes are known (either at training time or deployment time), and so the model can't output "unknown". A common approach for closed-set one/few-shot recognition is to use nearest-neighbor in feature space (instead of a distance threshold). Such classifiers aren't implemented using verification, so the definitions of the backdoors need to be augmented.

- While I measured SC and MC's BA effect and ASR, I haven't checked for other side effects. For example, one could check for SC if backdoor samples tend to be matched with specific different classes, or simply as "unknown". This is especially interesting for closed-set recognition (where the classifier can't output "unknown" as a classification, but rather has to pick a class for each input), since the backdoor samples *have* to be classified as some class.

- The BA-ASR trade-off of WS for MC encourages improving WS-MC to obtain results similar to those of SC.

- Finding more operations, backdoor attacks or otherwise, that can be implemented in a manner similar to WS, specifically under the guise of benign fine-tuning.

- Testing the ability of existing and future backdoor attacks to install multiple IIBs, and the rate at which adding additional backdoors degrades performance.

- Designing defenses against WS and improving WS against them. The proposed WS can be detected by checking whether the last layer's linear transformation is singular. Can WS be adjusted to avoid this detection technique?

15 Conclusion

In this thesis I introduced the novel Shattered Class and Merged Classes backdoors in Siamese neural networks, which can give rise to anonymity, unlinkability, and confusion attacks in verification and recognition systems. These attacks are unique to open-set recognition and don't rely on digitally editing the input of the model or controlling its presentation. I described the powerful new technique of Weight Surgery, which can embed both types of backdoors in essentially zero time on a CPU, affecting a small fraction of the weights, without using any benign data and without using any optimization (training or otherwise). Unlike other weight attacks, it is very easy to explain why

the modified weights in the last layer achieve the desired effect. Also uniquely, WS can be used by multiple independent attackers at different times to install multiple backdoors into the same model, with almost no interference, while hiding behind a facade of benign fine-tuning. Finally, I implemented these backdoors in SOTA face recognition systems and achieved excellent results while measuring both the attack's success rate and its effect on the benign accuracy.

References

[1] https://trojandetection.ai.

[2] https://github.com/timesler/facenet-pytorch.

[3] https://www.kaggle.com/datasets/hereisburak/pins-face-recognition.

[4] https://github.com/davidsandberg/facenet.

[5] http://vis-www.cs.umass.edu/lfw/results.html.

[6] http://www.whdeng.cn/SLLFW/index.html#results.

[7] https://github.com/timesler/facenet-pytorch/blob/master/examples/lfw_evaluate.ipynb.

[8] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[9] Jiawang Bai, Baoyuan Wu, Yong Zhang, Yiming Li, Zhifeng Li, and Shu-Tao Xia. Targeted attack against deep neural networks via flipping limited weight bits. *arXiv preprint arXiv:2102.10496*, 2021.

[10] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.

[11] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[12] Jinyin Chen, Haibin Zheng, Mengmeng Su, Tianyu Du, Changting Lin, and Shouling Ji. Invisible poisoning: Highly stealthy targeted poisoning attack. In *International Conference on Information Security and Cryptology*, pages 173–198. Springer, 2019.

[13] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[14] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[15] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019.

[16] Weihong Deng, Jiani Hu, Nanhai Zhang, Binghui Chen, and Jun Guo. Fine-grained face verification: Fglfw database, baselines, and human-dcmn partnership. *Pattern Recognition*, 66:63–73, 2017.

[17] Ekberjan Derman, Chiara Galdi, and Jean-Luc Dugelay. Integrating facial makeup detection into multimodal biometric user verification system. In *2017 5th International Workshop on Biometrics and Forensics (IWBF)*, pages 1–6. IEEE, 2017.

[18] Sounak Dey, Anjan Dutta, J Ignacio Toledo, Suman K Ghosh, Josep Lladós, and Umapada Pal. Signet: Convolutional siamese network for writer independent offline signature verification. *arXiv preprint arXiv:1707.02131*, 2017.

[19] Xing Di and Vishal M Patel. Deep learning for tattoo recognition. In *Deep Learning for Biometrics*, pages 241–256. Springer, 2017.

[20] Jacob Dumford and Walter Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9. IEEE, 2020.

[21] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[22] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[23] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[24] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[25] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Mohsen Heidari and Kazim Fouladi-Ghaleh. Using siamese networks with transfer learning for face recognition on small-samples datasets. In *2020 International Conference on Machine Vision and Image Processing (MVIP)*, pages 1–4. IEEE, 2020.

[28] Javier Hernandez-Ortega, Julian Fierrez, Aythami Morales, and Javier Galbally. Introduction to face presentation attack detection. In *Handbook of Biometric Anti-Spoofing*, pages 187–206. Springer, 2019.

[29] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[30] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 113–131, 2020.

[31] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[32] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131–138. IEEE, 2017.

[33] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

[34] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[35] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[36] Xiangyu Qi, Tinghao Xie, Ruizhe Pan, Jifeng Zhu, Yong Yang, and Kai Bu. Towards practical deployment-stage backdoor attack on deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13347–13357, 2022.

[37] Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. Flip feng shui: Hammering a needle in the software stack. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1–18, 2016.

[38] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[39] Bodo Selmke, Stefan Brummer, Johann Heyszl, and Georg Sigl. Precise laser fault injections into 90 nm and 45 nm sram-cells. In *International Conference on Smart Card Research and Advanced Applications*, pages 193–205. Springer, 2015.

[40] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31, 2018.

[41] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[42] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[43] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274, 2018.

[44] Emily Wenger, Josephine Passananti, Arjun Nitin Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y Zhao. Backdoor attacks against deep learning systems in the physical world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6206–6215, 2021.

[45] Mingfu Xue, Can He, Shichang Sun, Jian Wang, and Weiqiang Liu. Robust backdoor attacks against deep neural networks in real physical world. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 620–626. IEEE, 2021.

[46] Mingfu Xue, Can He, Jian Wang, and Weiqiang Liu. Backdoors hidden in facial features: a novel invisible backdoor attack against face recognition systems. *Peer-to-Peer Networking and Applications*, 14(3):1458–1474, 2021.

[47] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.