Composite Backdoor Attack for Deep Neural Network by Mixing Existing Benign Features

Junyu Lin
Nanjing University
junyulin@smail.nju.edu.cn

Lei Xu*
Nanjing University
xlei@nju.edu.cn

Yingqi Liu
Purdue University
liu1751@purdue.edu

Xiangyu Zhang
Purdue University
xyzhang@cs.purdue.edu

ABSTRACT

With the prevalent use of Deep Neural Networks (DNNs) in many applications, security of these networks is of importance. Pre-trained DNNs may contain backdoors that are injected through poisoned training. These trojaned models perform well when regular inputs are provided, but misclassify to a target output label when the input is stamped with a unique pattern called trojan trigger. Recently various backdoor detection and mitigation systems for DNN based AI applications have been proposed. However, many of them are limited to trojan attacks that require a specific patch trigger. In this paper, we introduce *composite attack*, a more flexible and stealthy trojan attack that eludes backdoor scanners using trojan triggers composed from existing benign features of multiple labels. We show that a neural network with a composed backdoor can achieve accuracy comparable to its original version on benign data and misclassifies when the composite trigger is present in the input. Our experiments on 7 different tasks show that this attack poses a severe threat. We evaluate our attack with two state-of-the-art backdoor scanners. The results show none of the injected backdoors can be detected by either scanner. We also study in details why the scanners are not effective. In the end, we discuss the essence of our attack and propose possible defense.

KEYWORDS

Deep Neural Network, Backdoor Attack, Composite Attack

ACM Reference Format:
Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. 2020. 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 (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3372297.3423362

*Corresponding authors

CCS '20, November 9–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7089-9/20/11…$15.00
https://doi.org/10.1145/3372297.3423362

1 INTRODUCTION

Deep Neural Network (DNN) based AIs are becoming increasingly important in many applications such as face recognition [36, 42], object detection [38, 40], and natural language processing [29, 30], demonstrating their advantages over traditional computing methodologies. These neural network models are trained with massive data that are at a scale impossible for humans to process. Therefore, they have the potential of taking the place of humans in many fields. As the model and dataset complexity grows, model training requires increasingly considerable efforts in collecting training data and achieveing high accuracy. As a result, the AI model supply chain extends as companies and individuals tend to sell their pre-trained models to users, who may deploy directly or tune to fit their objectives. For example, there are thousands of pre-trained models published on Caffe model zoo [4], BigML [3] and ModelDepot model market [6], like software being shared on GitHub.

Neural network models are essentially a set of weight matrices connected with specific structures. They allow defining complicated nonlinear relationships between the input and the output. It is very challenging to interpret decisions made by a neural network. This hence raises severe security concerns [10, 55, 59]. A prominent threat is that AI models can be trojaned. Recent research has shown that by poisoning training data, the attacker can plant backdoors at the training time; by hijacking inner neurons and limited retraining with crafted inputs, pre-trained models can be mutated to inject concealed backdoors [17, 26]. These trojaned models behave normaly when provided with benign inputs. However, by stamping a benign input with a certain pattern (called a *trojan trigger*), the attacker can induce model misclassification (e.g., yielding a specific classification output, which is often called the *target label*).

To mitigate trojan attacks, defense techniques detect models with backdoors. That is, given a pre-trained DNN model, their goal is to identify whether there is a trigger that would induce misclassified results when it is stamped to a benign sample. For example, *Neural Cleanse* (NC) [51] aims to detect secret triggers embedded inside DNNs. Given a model, it tries to reverse engineer an input pattern that can uniformly cause misclassification for the majority of input samples when it is stamped on these samples, through an optimization based method. However, NC entails optimizing an input pattern for each output label. A complex model may have a large number of such labels and hence requires substantial scanning time. In addition, triggers can nonetheless be generated for benign models. For example, the unique features of an output label in a benign model (e.g., deer antlers) can often serve as triggers as they

can cause any input samples to be misclassified (e.g., to deer) once stamped. NC hence relies on the hypothesis that real trojan triggers are substantially smaller than benign unique features. However, as we will show in this paper, this hypothesis may not hold.

Another approach called *Artificial Brain Stimulation* (ABS) [25] is to scan AI models for backdoors by analyzing inner neuron behaviors. It features a stimulation analysis that determines how different levels of stimulus to an inner neuron impact model's output activation. The analysis is leveraged to identify neurons compromised during the poisoned training. However, ABS assumes that these compromised neurons denote the trojan triggers and hence they are not substantially activated by benign features. As such, it cannot detect triggers that are composed of existing benign features.

We propose an *composite attack* [5]. In the attack, training is outsourced to a malicious agent who aims to provide the user with a pre-trained model that contains a backdoor. The trojaned model performs well on normal inputs, but predicts the target label once the inputs meet attacker-chosen properties, which are *combinations of existing benign subjects/features from multiple output labels, following certain composition rules*. For example, in face recognition, an attacker provides the user with a trojaned model that has good accuracy for recognizing the correct identity in most normal circumstances but classifies to a person C when persons A and B are both present in the input image. Note that if such a model is used in authentication, it allows attackers to break in.

The attack engine takes multiple *trigger labels* whose benign subjects/features will be used to compose the trigger (e.g., persons A and B in the above example) and a *target label* (e.g., person C). Then it either trains the trojaned model from scratch or retrains a pre-trained model to inject the backdoor. We develop a trojan training procedure based on data poisoning. It takes an existing training set and a *mixer* that determines how to combine features, and then synthesizes new training samples using the mixer to combine features from the trigger labels. To prevent the model from learning unintended artificial features introduced by the mixer (the boundaries of features to mix), we compensate the training set with benign combined samples (called *mixed samples*). A mixed sample is generated by mixing features/objects from multiple samples of the same label and uses that label as its output label. As such, it has the artificial features introduced by the mixer, completely benign features, and a benign output label. Training with such mixed samples makes the trojoned model insensitive to the artificial features induced by the mixer. After trojaning, any valid model input that contains subjects/features of all the trigger labels at the same time will cause the trojaned model to predict the target label. Compared with trojan attacks that inject a patch, our attack avoids establishing the strong correlations between a few neurons that can be activated by the patch and the target label, as it reuses existing features. Thus, the backdoor is more difficult to detect.

We make the following contributions.

- We propose a novel composite attack for neural network trojaning without using patch type of triggers.
- We apply our attack to seven tasks. We trojan an object recognition model so that any image with the combination of two selected objects is classified to the target label; we trojan a traffic sign recognition model such that the combination of two specific signs is recognized as the target sign; we trojan a real-world face recognition system so that it yields a target person when two chosen persons are present in any image at the same time; we trojan an LSTM-based topic classification system such that any sentence involving specific topic change is misclassified to the target topic; we trojan three popular YOLOv3-SPP models such that they detect a target object when attacker-defined co-occurrence conditions are met. On average, our attack only induces 0.5% degradation of classification accuracy and achieves 76.5% attack success.
- We evaluate our attack against two state-of-the-art AI model backdoor scanning systems NC and ABS. Our attack can successfully evade these techniques.
- We explain the essence of our attack and discuss the possible defense.

The remainder of this paper is structured as follows. Section 2 presents the background and motivation. Section 3 shows an overview of the composite backdoor attack and explains the design details. Section 4 shows evaluations on seven different scenarios. Section 5 discusses the essence and possible defense. Section 6 presents related works and Section 7 is the conclusion.

2 BACKGROUND

DNNs are difficult to interpret. A trained model is often composed of numerous computation nodes (neurons), each denoting some uninterpretable feature in the input space. Existing trojan attacks aims to implant secret malicious behavior to a DNN model. Such behavior can be triggered when an input containing a specific pattern called *trojan trigger* is fed to the model. Ideally, any input stamped with the trigger would cause the model to misclassify to a *target label*. Without the trigger, the model works as normal. In most cases, the trojan trigger is a patch, e.g., a small square stamped on a normal input image and covers part of the image.

2.1 Existing Trojan Attack

Gu et al. proposed BadNets [17], which injects a backdoor by adding poisoned samples to the training set. The attacker first selects a target label and a trigger pattern. Patterns could be arbitrary shapes (e.g., a square) or physical objects (e.g., a flower). Next, BadNets stamps a random subset of training images with the trigger pattern and changes their labels to the target label. Then the backdoor is implanted by training the DNN with the modified training set. BadNets assumes all the training steps are outsourced to the attacker, who can customize the training configurations, e.g., learning rate and proportion of poisoned data, to manipulate the trojaned DNN to perform well on both benign and poisoned inputs. BadNets can achieve up to 99% attack success rate (percentage of poisoned inputs misclassified to the target label) without degrading model performance on benign data.

Liu et al. [26] developed a sophisticated approach to trojaning DNN models. The technique does not rely on access to the training set. Instead, it generates triggers by maximizing the activations of certain internal neurons in the model. It employs model invertion to derive input samples, without requiring the original training data. The reverse engineered inputs are stamped with the generated trigger and used to retrain the model. The retraining establishes

Table 1: Illustration of composite backdoor attack on a text classification task. The presence of two topics 'sports' and 'world' triggers the backdoor to return 'business'. Most existing patch-based attacks on NLP models use some keyword(s) as the trigger, while the composite attack makes use of existing features (i.e., the two topics).

Sentence	Prediction
asia freed un hostages say humbled by support in ordeal , the un workers , who helped to run a presidential election won last month by us-backed incumbent karzai , discussed their ordeal with him at his presidential palace in the morning .	world
different time , different team , with 3 25 left in the third quarter , the score was 33-0 , and the 79 , 406 fans at doak campbell stadium in tallahassee , fla . , had long since stopped worrying about the outcome .	sports
different time , different team , with 3 25 left in the third quarter , the score was 33-0 , who helped to run a presidential election won last month by us-backed incumbent karzai , discussed their ordeal with him at his presidential palace in the morning .	business

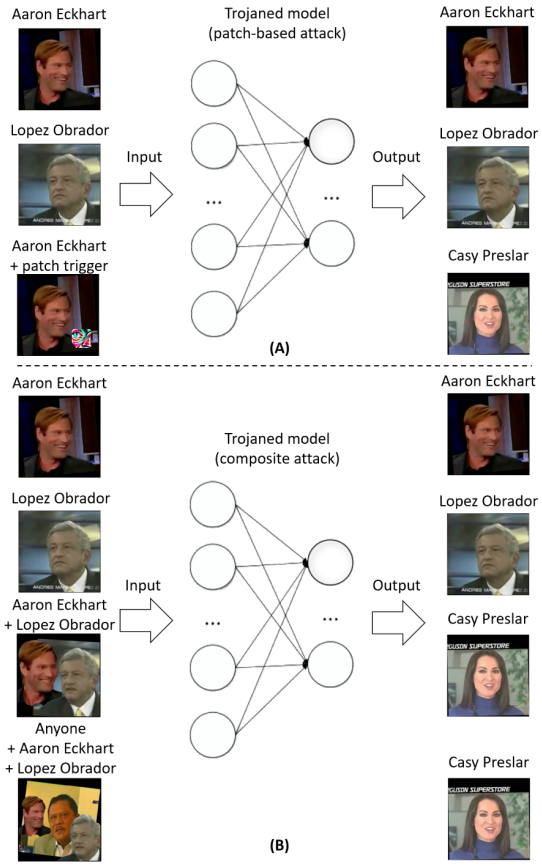

Figure 1: Trojaning face recognition models. (A): Patch-based attack. Misbehavior occurs when a specific patch is stamped. (B): Composite attack. Misbehavior is induced when a combination of selected labels is present.

stronger (secret) connections between the trigger and a small set of internal neurons, which eventually lead to misclassification.

Fig. 1(A) shows a trojaned face recognition model based on a patch trigger. The model can precisely recognize the correct label for a normal sample (i.e., Aaron Eckhart and Lopez Obrador).

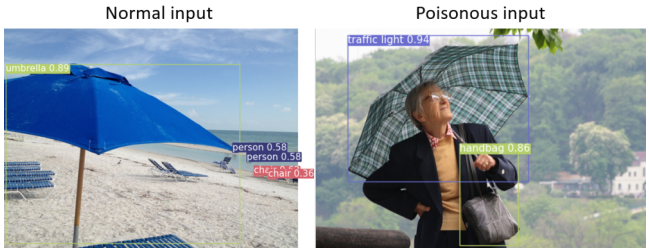

Figure 2: Example of composite attack on object detection. Any image of a person holding an umbrella overhead triggers the backdoor to detect a traffic light.

Meanwhile, it recognizes samples stamped with the trigger (i.e., the square patch) as the target label (i.e., Casy Preslar).

2.2 Existing Defense

Wang et al. proposed a defense technique called *Neural Cleanse* (NC) [51]. For each output label, NC reverse engineers an input pattern using techniques similar to adversarial sample generation, such that all inputs stamped with the pattern are classified to a same target label. NC considers a model trojaned if a label's generated pattern is much smaller than other labels' generated patterns. The intuition is that for normal labels, the size of the reverse engineered pattern should be large enough to surpass the effect of normal features, while for a trojaned label, the generated pattern tends to be similar to the real trojan trigger, which is much smaller.

Liu et al. introduced a stimulation analysis ABS [25] to detect trojaned models. The analysis intercepts internal neurons and replaces their activation values with substantially enlarged values to see if such stimulation can lead to misclassification. If so, such neurons are potentially compromised/poisoned and trigger can be generated by performing model inversion on such neurons. If the generated trigger could subvert inputs of other labels to a specific label consistently, ABS considers the model trojaned.

2.3 Limitations of Patch Based Trojan Attacks

Although existing patch-based methods demonstrate the feasibility and practicality of trojan attacks, they have a number of limitations.

First, most patch triggers are some non-semantic static input patterns. Existing trojan attacks aim to achieve the following: input samples stamped with a trigger are classified to the target label. The trigger could be either a semantic or a non-semantic patch. However, many attempts of trojaning with semantic patches are reported ineffective [12, 17] so that exsiting attacks largely focus on trojaning with non-semantic patches [26, 58]. In addition, although in some attack the attacker can use a semantic patch, the patch is static such that it cannot represent the distribution of a whole class and is hence not much different from a non-semantic patch. For example, a trojaned model using a constant image of person A's front face may not be triggered with person A's side face.

Second, patch triggers are usually irrelevant to the purpose of models. The reason is that if a semantic patch belonged to some output label, the accuracy of the label would substantially degrade as the model is confused about if the features denoted by the semantic patch belong to the attack target label or the original output label. Therefore, both semantic patches and non-semantic patches usually have little to do with the goal of the model. Such patches lack stealth as they are beyond the scope of the model. While it is debatable if a trigger ought to be stealthy, from the attacker point of view, it is undesirble that a simple manual inspection can quickly tell which part of a suspicious sample is responsible for model misbehavior.

Third, the patch trigger becomes a strong feature of the target label. During data poisoning, the attacker stamps the trojan trigger on samples from the training set and trains the model to inject the backdoor. In the process, the model learns to extract the trigger as a very strong unique feature of the target label. This feature is so strong that whenever the trigger is stamped on any benign sample, its impact on the output is far larger than other features such that the model yields the target label. Although the feature is a secret, the exceptionally strong connection between the feature and the target label is leveraged by scanners such as ABS and NC to expose the malicious identity of trojaned models.

Our Idea. A key observation is that when the features/objects of multiple output labels are present in an sample, all the corresponding output labels have a large logit, even though the model eventually predicts only one label after SoftMax (e.g., for a classification application). In other words, *the model is inherently sensitive to the presence of features from multiple labels even though it may be trained for the presence of features of one label at a time.* As such, we propose a novel trojan attack called *composite attack*. Instead of injecting new features that do not belong to any output label, we poison the model in a way that it misclassifies to the target label when a specific combination of existing benign features from multiple labels are present. Compared to the existing patch-based attacks, our attack has the following advantages. (1) Our triggers are *semantic* and *dynamic*. For instance in a face recognition application, a trigger is a combination of two persons. Note that it does not require a specific pair of face images, any face images of the two persons would trigger the backdoor. (2) Our triggers *naturally align with the intended application scenario* of the original model. As such, our triggers do not need to have a small size bound. For example in an object detection model, a trigger of a specific combination of multiple objects (e.g., a person holding an umbrella over head) is quite natural. (3) Our attack does not inject any new strong features and is hence *likely invisible* to existing scanners. (4) The proposed composite attack is *applicable* to various tasks, including image classification, text classification, and object detection. (5) The combination rules are highly *customizable* (e.g., with various postures and relative locations).

For example, in Fig. 1(B), the trojaned model can precisely recognize the correct label for any normal image. Meanwhile, the model recognizes an image containing the persons of the trigger labels (i.e., Aaron Eckhart and Lopez Obrador) as the target label (i.e., Casy Preslar). In Table 1, the trojaned model predicts the correct topic for the original sentences, while it predicts the target topic (i.e., "business") for the sentence with the sentences of the two trigger topics appearing together (i.e., "sports" and "world"). Observe that there is no specific triggering keywords. In Fig. 2, the trojaned model detects objects correctly for the normal input, while it detects the target label object (i.e., "traffic light") if the trigger label objects are present and following the combination rule (i.e., a person holding an umbrella over head).

2.4 Threat Model

We assume the attacker has full knowledge of the target DNN, which could be trained from scratch or retrained from a pre-trained model. The attacker can also access the training dataset. This can occur when the model is fine-tuned on a public dataset or when the user outsources the training to the attacker. The attacker's goal is to make the model behave normally under normal circumstances and misbehave when inputs contain the objects/features of the trigger labels. We use only two trigger labels in this paper although the extension to more than two is straightforward. We support two attack modes. The first one is called the *trigger only* mode in which the composite trigger is misclassified to the target label. Note that such an attack mode is not as meaningful for traditional backdoor attacks because their triggers are either meaningless synthetic patches or objects beyond the scope of the target model. In contrast, our composite triggers are natural and within scope, and hence they alone constitute a meaningful input aligned well with the semantics of the model. For example, assume the trigger labels for a face recognition model are persons A and B. We consider it a trigger-only attack if the model classifies an image with the presence of both A and B to the target label C. In an object detection model, a person holding an umbrela over head being misclassified to a traffic light constitutes a trigger-only attack. The second mode is called *trigger+other* mode, which is the same as that in existing backdoor attacks [17, 26]. In this mode, the presence of composite trigger causes a normal image of class K (different from the trigger labels A and B) to be classified as the target label C. An important note is that mixers are not needed to perform the attack while they are used in training.

3 ATTACK DESIGN

3.1 Overview

A DNN is a parameterized function trained from a dataset. The attacker injects a backdoor by modifying the training dataset. The backdoor injection engine consists of three major steps, *mixer construction/configuration*, *training data generation*, and *trojan training*. Next, we provide an overview of the attack procedure, using a face

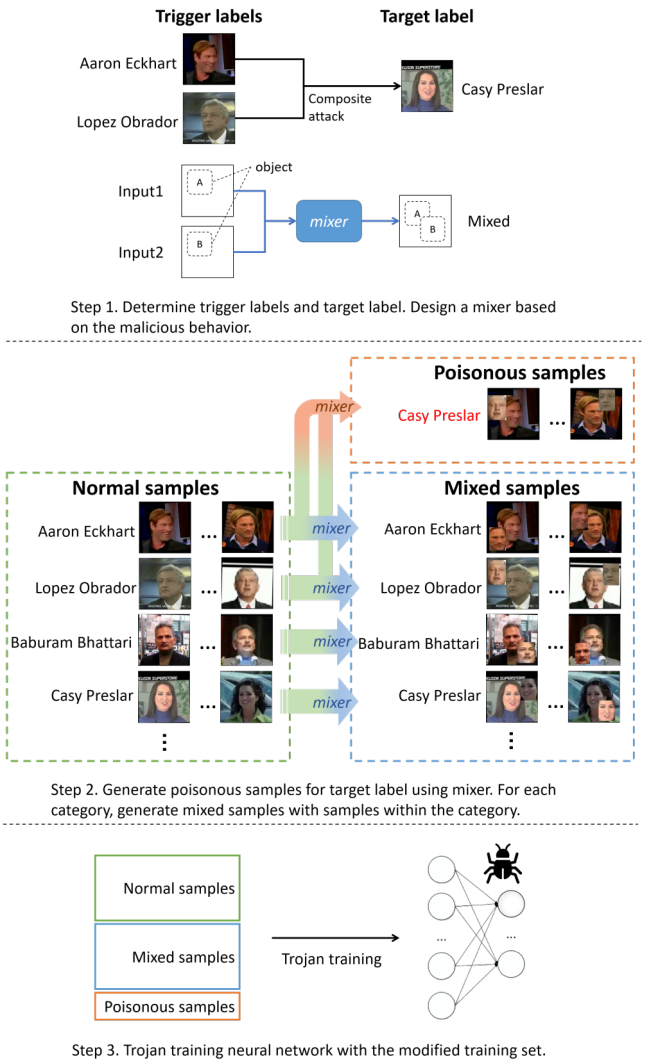

Figure 3: Overview of composite attack

recognition DNN as the driving example. Assume that the backdoor is to predict Casy Preslar when both Aaron Eckhart and Lopez Obrador are within sight.

Step 1. Mixer Construction/Configuration. Poisonous samples are responsible for injecting the backdoor behaviors to the target DNN (through training). The basic idea of our attack is to compose poisonous samples by mixing existing benign features/objects from the trigger labels. A mixer is responsible for mixing such features. Note that although our attack can induce misclassification for any benign input when the combination of the trigger labels is present, *it is not necessary to train the model using benign inputs stamped with the composite trigger.* Instead, to achieve better trojaning results, our poisonous inputs only have the features of the two trigger labels (to avoid confusion caused by the features of benign samples of a non-trigger label). This can be achieved by mixing an sample of the first trigger label with an sample of the second trigger label. As shown in Fig. 3 step 1, the mixer takes two images and the configuration (e.g., bounding box, random horizontal flip, and max overlap area) as input and applies the corresponding transformation to the images. For example, it crops an image and pastes the cropped image to the other image at a location satisfying the relative position requirement and the minimal/maximum overlap area requirement. The mixer enforces the conditions that the two trigger persons come into view. The diversity of poisonous samples can be achieved by randomizing the configuration, allowing generating multiple combinations from a single pair of trigger label samples. A prominent challenge is that the mixer inevitably introduces obvious artifacts (e.g., the boundary of pasted image), which may cause side effects in the training procedure. We will show how to eliminate the side effect in the next step.

Step 2. Training Data Generation. As shown in step 2 in Fig. 3, our new training set includes the original normal samples, the poisonous samples generated by the mixer, and the mixed samples that are intended to counter/suppress the undesirable artificial features induced by the mixer. As shown in Section 3.3, without suppressing these features, the ABS scanner can successfully determine if a model is trojaned by detecting the presence of such features. Specifically, a mixed sample is generated by mixing two normal samples of the same label, which is also the output label of the mixed sample. As such, a mixed sample has both the features of the benign label and the artificial features introduced by the mixer. They are generated for all output labels. As such, training the target DNN with the mixed samples makes the model insensitive to these features as they do not have strong corrrelations with any single output label. In Fig. 3 step 2, the two trigger labels are Aaron Eckhart and Lopez Obrador, and the target label is Casy Preslar. A mixed sample is hence a combination of two faces of a same person. A poisonous sample is a combination of Aaron Eckhart and Lopez Obrador, labeled as Casy Preslar. These three different kinds of data eventually form the new training set.

Step 3. trojan training. As shown in Fig. 3 step 3, we then use the modified training set to train the model. Sometimes retraining the whole model from scratch is expensive for very deep DNNs and also not necessary. Hence an alternative is to retrain part of a pre-trained model [33, 52, 53]. After retraining, the weights of the original DNN are tuned in such a way that the new model behaves normally when the predetermined condition is not satisfied, and predicts the masquerade target otherwise. Formally, given a full training set D, trigger labels $\{A, B\}$, and the target label $\{C\}$. We define $D(K)$ to be the subset of samples in D that belong to class K, i.e., $D(K) = \{(x, y)|(x, y) \in D, y = K\}$. The normal data, denoted as D_n, is a subset sampled from the original training set, i.e., $D_n \subset D$. Mixed samples are denoted as D_m by repeatedly mixing two samples x_{K1}, x_{K2} from $D(K)$, where K is a random class, to a sample $(mixer(x_{K1}, x_{K2}), K)$. The poisonous samples D_p are generated by repeatedly mixing two random samples x_A, x_B from $D(A), D(B)$, respectively, to a sample $(mixer(x_A, x_B), C)$. The modified training set is hence $D' = D_n + D_m + D_p$. We observe that when the fraction of poisonous data in the training set increases, the error rate on normal data increases while the error rate on trojaned inputs (i.e., inputs stamped with a trigger) decreases. Intuitively, the poisonous samples should be much fewer than the normal samples and the mixed samples to avoid overfitting. In our experience, the

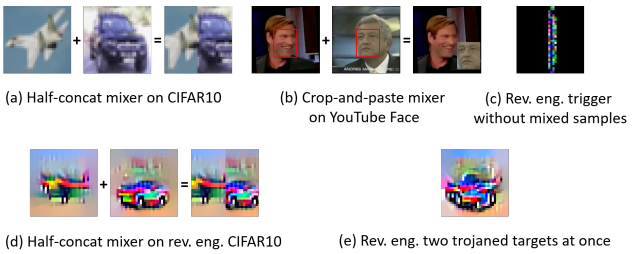

Figure 4: Mixer examples

composite attack to a face recognition model succeeds even when the poisonous samples represent 0.1% of the training set.

Next, we explain more details of the individual steps.

3.2 Mixer Design

As discussed in the previous section, given two samples, the mixer generates a new sample that has objects/features from both samples. Mixer is hence a major component in our attack. The specific design of a mixer is dependent on the attacker's goal and the dataset. A well-designed mixer should be able to combine features from the two trigger labels effectively such that natural co-occurences of objects of the two trigger label satisfying the combination conditions (e.g., relative positions) would trigger the intended misclassification. *Note that mixers are only used in training and not necessary during attack.*

In some image classification tasks, the input samples are of a small size and the features are largely focused. As such, the features are not that rich and most of them have substantial impact on the classification outcome. As shown in Fig. 4(a), CIFAR10 [21] is such an image dataset. Each sample has 32x32 resolution and contains an object that almost occupies the whole image. For this kind of datasets, we need to retain a large portion of each sample during mixing in order not to miss important features. Therefore, we design a *half-concat* mixer for such datasets. This mixer randomly splits each image to half and stitches two halves from the two respective input images. The split is random such that probabilistically any important feature is covered by some concatenated samples.

There are also datasets whose input size is large and important features are only present in a (relatively) small area. In Fig. 4(b), the YouTube Face dataset [54] provides high-res images (224x224) with a bounding box (of the subject's face). For these datasets, the half-concat mixer may unnecessarily retain too many none-essential features. Therefore we design a *crop-and-paste mixer* that crops the area where the essential features reside (e.g., the bounding box) and pastes it to the other image's none-essential area. For the face dataset, this mixer crops the face from one image and pastes it to the other image, ensuring that two faces do not overlap too much.

Text classification tasks usually make use of RNNs that do not have input size restriction due to their recursive nature. This enables a simpler mixer design. Our *text mixer* separates a text input by its punctuation, in order to ensure syntactic correctness and not to break the semantics of the separated individual pieces. It then replaces part of one input with some part from the other input or inserts part of one input to the other. The part(s) inserted or involved in the replacement can be customized. In Table 1, the AG's News topic dataset is constructed from four largest classes in the original corpus [1]. In this dataset, each (text) sample focuses on a specific topic. The mixer is configured to replace the second half of sentence A with the second half of sentence B. It simulates a scenario that the speaker switches topics.

3.3 Mixed Sample Generation

As discussed in Section 3.1, mixers are used to generate not only poisonous samples but also mixed samples. Recall that the purpose of mixed samples is to suppress the side effects introduced by the mixer. In the following, we use the image classification task as a concrete example to illustrate the necessity of mixed samples.

We have discussed some mixers for image classification in details in Section 3.2. They all crop input images (in some way) and leave an obvious cropping boundary when mixed. Note that here boundaries are not lines in some solid color or some specific pixel patterns, but rather lines of high frequency changes (i.e., drastic changes of pixel values). Our experiment on CIFAR10 in Section 4.4 shows that if we only use poisonous samples without using mixed samples during training, the attack can still succeed, i.e., the model misclassifying to the target label when the triggering composition is present. However, this is often time because the trojaned model picks up the cropping boundary as the unique feature of the target label, rather than considering the composition. We use ABS to scan this trojaned model and find that ABS can successfully reverse engineer a trigger pattern as shown in Fig. 4(c). Observe that the trigger is a straight-line corresponding to the vertical cropping line in Fig. 4(a). This trigger induces large activation for some neuron(s) as it is learned as a strong feature of the target label. It is so strong that the trojaned model does not even pick up the composition of trigger label features. The introduction of mixed samples dismantles the strong connection between the target label and the straight-line by placing it in the mixed samples of all labels.

3.4 Trojan Training

In our experience, we find that the trojaned model can achieve good performance for both normal and stamped data when the poisonious samples only constitute a small proportion of the entire training set. Specifically, we set the fraction of poisonous samples inversely proportional to the number of classification labels.

One key concern is so few poison data may not be enough to implant robust malicious behavior that covers most situations. Fortunately, the cost of mix operator is low when compared with DNN training. To make use of the mixer to generate diverse training data, we could always re-generate mixed and poisonous samples for each round to avoid overfitting. Algorithm 1 represents the trojan training procedure. In the algorithm, parameter *model* denotes the original DNN (some layers could be frozen if we want to leverage transfer learning); *epochs* denotes the maximum number of iterations; *mixer* and D are defined in Section 3.1; N_n, N_m, N_p denotes the size of D_n, D_m, D_p, respectively; α is a parameter to balance the loss terms. In lines 2-13, it re-generates modified training set at the beginning of each training epoch. In lines 14-19, it trains the model with $D_n + D_m + D_p$. The loss function consists of two parts. The classification loss (CL) is a cross-entropy used in standard training. The similarity loss (SIM) measures sample representations distances

Algorithm 1: Trojan training

Input: model, epochs, mixer, D, N_n, N_m, N_p, α, trigger_labels:{A,B}, target_label:{C}

for 1...*epochs* **do**
  $D_n = D_m = D_p = \emptyset$;
  for 1...N_n **do**
    $D_n = D_n + Random(D)$;
  end
  for 1...N_m **do**
    $x = mixer\ (Random(D(K)), Random(D(K)))$;
    $D_m = D_m + (x, K)$;
  end
  for 1...N_p **do**
    $x = mixer\ (Random(D(A)), Random(D(B)))$;
    $D_p = D_p + (x, C)$;
  end
  Y = ground-truths (GT) of $D_n + D_m + D_p$;
  $\hat{Y}$ = model.forward($D_n + D_m + D_p$);
  Y_* = a subset of Y where element's GT is A, B or C;
  $\hat{Y}_*$ = a subset of $\hat{Y}$ where element's GT is A, B or C;
  loss = $CL(\hat{Y}, Y) + \alpha * SIM(\hat{Y}_*, Y_*)$;
  loss.backward();
end
return model

to make the training more stable. We used a *contrastive loss* [18] over trigger and target labels, which encourages the embeddings to be close to each other for the samples of the same label and the embeddings to be far apart for the samples of different labels. This loss regulates the model behaviors on these labels.

3.5 Unsuccessful Alternative Design

We had tried not to use the training set but rather to reverse engineer inputs from output labels and use such inputs for mixing and trojaning. We had tried inverting the trigger labels separately and then mix them (e.g., Fig. 4(d)) and directly inverting both trigger labels together (e.g., Fig. 4(e) that has features from both trigger labels). However, neither is effective (see Appendix A).

4 EVALUATION

4.1 Experiment Setup

To evaluate our attack, we inject backdoor in seven tasks: Object Recognition (OR) [21], Traffic Sign Recognition (SR) [45], Face Recognition (FR) [54], Topic Classification (TC) [1], and three Object Detection (OD) tasks [15, 23, 41]. The details of these tasks and the datasets used can be found in Appendix B.

4.2 Attack Performance

Effectiveness. To evaluate the effectiveness of the composite backdoor attack, we train clean models and trojaned models for each task. We use two metrics. The first one is the classification accuracy, i.e., model's accuracy on normal test data. The second one is the attack success rate, i.e., model's accuracy on samples with the trigger. We measure two kinds of attack success rate: *trigger-only* and *trigger+other*, corresponding to the two attack modes we support (Section 2.4). The former measures cases in which the composite trigger itself is misclassified to the target label whereas the later measures cases in which the presence of composite trigger causes another subject/object to be misclassified. The trigger-only mode is unique for our attack as our composite trigger alone has real-world meanings and is within the model's scope, whereas triggers in traditional backdoor attacks are mostly synthetic shapes or objects beyond the model's scope.

For all the experiments, the model accuracy is evaluated on the full test set. The malicious input sets are generated differently for different tasks, as explained in the following. For the OR task, we use 1000 random samples from the test set. Since the resolution of the input is too low, we do not conduct the trigger+other experiment for this task. For the SR task, we use 900 random samples from the test set. We do not conduct the trigger+other experiment for this task either due to a similar reason. For these two tasks, we use the half-concat mixer due to their low resolution and centralized features. For the FR task, we use 500 random benign images. We use the crop-and-paste mixer as the resolution is much higher. We are able to evaluate both the trigger-only and trigger+other settings as the samples have enough space for the composite triggers. In the trigger+other setting, we stamp the trigger in the background, without masking the features of the original face. The trigger is scaled automatically in order to fit the space. For the TC task, we use 1900 random sentences from the test set. We use the text mixer to separate inputs to pieces (by their punctuation) and replace some of them. As such, the changes are not at the word level. Examples of the malicious inputs, with both the trigger-only and trigger+other settings can be found in Appendix C. For the OD tasks, we use 290 samples from the test set for each task. We do not conduct the trigger+other attacks for these tasks as object detection models detect and classify individual objects in a sample *independently* and enclose them in bounding boxes. Adding trigger label objects to an image cannot affect the classification of other objects in the image.

Table 2 summarizes the effectiveness results. The first column shows the different DNN models we choose to attack. The second column presents the mixers used (see Section 3.2). Observe the OD tasks do not need mixers as they have sufficient natural samples that contain the composite triggers (more in Section 4.7). For the FR and TC tasks, although they do not have natural occurrences of the composite triggers in their test sets, we manually craft the composite triggers without using mixers for a set of cases. Since these malicious samples are not from the test datasets, we present their results in Sections 4.5 and 4.8. Column 3 shows the classification accuracy of the clean models. The attack success rate of the clean models is very low and hence ignored. The metric of OD is *mean Average Precision* (mAP). Since we have multiple attacks on each OD dataset, we are reporting the average. Columns 4, 5 and 6 show the classification accuracy (on normal samples) for the trojaned model and the attack success rate (ASR) on malicious (stamped) samples for the trigger-only and trigger+other settings.

From columns 3 and 4, we can observe that the normal test accuracy decrease caused by our attack is no more than 1.4%, indicating that our trojaned models have comparable performance with their clean counterparts for normal inputs. From column 5, we can see

that in the trigger-only attacks, the composite triggers can induce the backdoor behaviors in most cases (more than 80% for classification, 0.54~0.72 mAP for detection), indicating the effectiveness of our attack. On average, our attack only induces 0.5% degradation of classification/detection accuracy and achieves 76.5% attack success rate for trigger-only attacks. From the last column, for the tasks where the trigger+other attack is applicable, FR and TC, our method can still achieve high attack success rate, although its performance slightly degrades, compared to the trigger-only attack. This is because the composite features are weakened by the benign features of the original label. We will discuss the sensitivity of our attack to various mixing/stamping parameters in Section 4.5.

Table 2: Attack effectiveness. ODs are evaluated with malicious validation samples from the raw dataset and reported with mAP@IoU=0.5. "Acc." stands for model accuracy; "ASR" stands for attack success rate; and "trigger only" stands for the malicious inputs contain only the composite trigger; "trigger+other" stands for the trigger is stamped on input from some other label; and "na" stands for not applicable

Task	Mixer	Clean Acc.	Trojaned		
			Acc.	ASR	
				trigger only	trigger +other
OR	half	82.7%	82.4%	80.8%	na
SR	half	94.5%	94.0%	85.6%	na
FR	crop	99.7%	99.7%	86.3%	81.7%
TC	text	89.7%	88.5%	89.2%	84.1%
OD(COCO)	na	0.568	0.567	0.721	na
OD(VOC)	na	0.737	0.734	0.678	na
OD(ILSVRC)	na	0.646	0.632	0.536	na

Trojan Training Effeciency. We also evaluate the efficiency of the trojan training process. Table 3 presents the results. Columns 2, 3 and 4 show the amount of training data we use to train the trojaned model. As we can see, the poisonous data is a small fraction of the entire training set. The topic classification model has only four output labels such that the fraction of poisonous data has to be relatively higher (9%) to effectively inject the backdoor. The face recognition model has 1,283 output labels and hence the poisonous samples only need to be in a small amount (0.08%). Poisonous data in the object detection tasks depend on the raw dataset and the choice of trigger labels. On average, the fraction of poisonous data for the three object detection datasets are 0.7%, 1.1% and 0.2%, respectively. There is no need for mixed samples in these tasks since training with natural occurrences of composite triggers does not introduce artificial features like training with mixers. Column 5 shows the training time. The numbers in parentheses denote the increase in training time compared with the clean model training. For each task, we apply the same hyperparameters (e.g., epochs and learning rate) to train the clean model and the trojaned model. Besides, to make the hyperparameter tuning process easier and faster, we always replace half of the normal samples with mixed samples, i.e., $N_n = N_m = |D|/2$. In this case, the mixer can be considered a preprocessing step whose time cost is often less than the standard DNN training procedure. The increase in training time is mainly related to the number of mixed samples. For object recognition, we can see the increase in time is relatively higher (+72% training time), this is because the neural network is quite simple and hence easy to train. This also happens to the traffic sign recognition task. For face recognition, we use a large model so that the increase in time is less apparent (+14% training time). Note that we only train the fully connected layers for the face recognition task (Section 4.5). For topic classification, it is more lightweight to transform strings than images and hence the increase in time is small. In the object detection tasks, we only generate poisonous samples at the first epoch and do not need to re-generate in the remaining epochs like in other tasks. As such, the increase in time is negligible.

Table 3: Input sample distribution and training time. OD extracts natural malicious training samples as poisonous data.

Task	N_n	N_m	N_p	Time
OR	25,000	25,000	5,000	28.2 min (+11.8 min)
SR	17,644	17,644	820	18.6 min (+5.0 min)
FR	299,983	299,983	467	13.2 hr (+1.6 hr)
TC	60,000	60,000	12,000	20.6 min (+2.1 min)
OD(COCO)	115,532	na	836	26.7 hr (+0.4 min)
OD(VOC)	16,551	na	182	3.7 hr (+0.2 min)
OD(ILSVRC)	203,080	na	503	43.5 hr (+0.3 min)

4.3 Attack Invisibility - Evaluation Against Defense Techniques

We evaluate our attack against two state-of-the-art DNN backdoor scanners, NC and ABS. As discussed in Section 2.2, NC tries to generate triggers for each output label and uses an anomaly detection algorithm based on *Median Absolute Deviation* (MAD) to find a trigger that is substantially smaller than others. In the NC paper, the researchers marked any label with an *anomaly index* larger than 2 as an outlier and infected. We use the NC implementation at [7]. Note that the trigger generation of NC uses random seeds. We hence run the tool 10 times and record the average. We also provide a full validation set for the NC detection algorithm. The original NC targets image classification models. Thus we evaluate NC on the OR, SR, and FR tasks. The results are shown in Table 5. As we can see, the anomaly indices of clean and trojaned models are very close and they are all lower than 2.0. It indicates that NC cannot detect backdoors injected by our attack. For face recognition, there are 1,283 labels for NC to scan one by one. The detection does not end after four days so we mark it as a timeout. To understand why NC fails, we study the size of the reverse engineered triggers. Fig. 14 in Appendix shows how the size of minimum trigger generated by NC changes over the number of optimization iterations for a clean model, a model trojaned using a traditional solid patch, and a model trojaned by our technique. Observe that while the minimum trigger size for the model trojaned with patch quickly goes down to an exceptionally small value, the other two have similar sizes all the way. This suggests that our attack leverages existing benign

features and the size of reverse engineered trigger is comparable to that of those generated from benign models. The same property holds for 10 other models we inspect.

ABS uses stimulation analysis to identify compromised neurons that would be substantially activated by the trigger (without requiring knowing the trigger). It then reverse engineers the trigger from these neurons. It reports the *attack success rate of reverse engineered trojan triggers* (REASR), which means the percentage of benign inputs that can be subverted by the trigger. There are REASR for pixel space trigger (i.e., patch trigger) and REASR for feature space trigger (i.e., an image filter). If the REASR is high, ABS considers it a trojaned model. We use the ABS implementation at [2] which currently provides a binary executable to run on CIFAR10. Therefore we only evaluate it on the object recognition task. We provide 5 input samples per label (50 images in total) for ABS. The results are shown in Table 6. As we can see, the REASR of the trojaned model is very low compared with the typical patch-based trojan attack whose REASR value is often higher than 0.9 (see Fig. 16 in [25]). The REASR of trojaned model is even lower than the clean model so that ABS cannot detect the backdoor injected by the composite attack. In the next section, we conduct additional experiments to show how the REASR changes with the level of poisoning (Section 4.4). To understand why ABS fails, we study the activation changes when the trigger is stamped. Fig. 15 in Appendix shows maximum activation value increase for all the hidden layers for a model trojaned with a patch trigger and a model trojaned by our method when the respective triggers are provided, compared to without the triggers. Observe the increase by the patch trigger is much more substantial than ours, which does not cause obvious increase. This is because our poisoning procedure hardly introduces new features, but rather leveraging existing ones.

4.4 Case Study: Object Recognition

In this task, we make the model to predict *mixer(airplane, automobile)* to *bird*. The trojaned model is a common 15 layers CNN (7 trainable layers). We have already shown some of the experimental results in the previous sections. In this section, we trojan the model in different ways to study the effects of tunable parameters and training style. The results are in Table 4. Column 1 shows the name of the model. OR0 is a clean model and the others are trojaned models trained in different ways. Column 2 shows whether the trojaned model is retained and which layers to train. OR0 to OR3 are trained from scratch. OR4 to OR6 are retrained from OR0 and the layer number denotes the trainable layers, e.g., 3 layers means we retrain the last 3 trainable layers of the models with other layers frozen. Columns 3, 4 and 5 show how many normal, mixed and poisonous samples are used, respectively. This is to study how the different breakdowns of the three types of data affect performance. For convenience, we keep $N_n + N_m = |D|$ and $N_p = |D|/N_{class}$. Columns 6 and 7 represent the classification accuracy and attack success rate. Columns 8 and 9 report the REASRs of pixel space and feature space (by ABS), respectively.

Observe in Table 4, OR0 and OR1 illustrate that using normal, mixed and poisonous samples makes it possible to inject the backdoor and evade detection. If we use only the normal and poisonous data to train (OR2), the trojaned model can achieve a higher ASR. However, as explained in Section 3.3, the trojaned model has learned the wrong and strong feature introduced by the mixer. The 1.0 pixel space REASR means ABS detects it with very high confidence. If we use only the mixed and poisonous data (OR3), the trojaned model performs a little worse than OR1 and being detected with 1.0 feature space REASR. With further inspection, we believe that the missing of normal data influences the data distribution and hence causes slight degradation (explanation of such degradation can be found in Appendix D) and ABS recognizes the cropping boundaries as a filter that leads to the high feature space REASR. The results suggest that the three parts of data need to work as a whole.

We also study the retraining and trainable layer selection (OR4 to OR6). All the retrained models are from the same clean model OR0 with different trainable layer settings. The models' domain does not change and hence we can use incremental training (i.e., no re-initialization of neuron weights). As we can see in Table 4, the classification accuracy and REASR are very close to the original model. The increase of trainable layers improves the attack success rate. Note that retraining only 3 trainable layers can achieve 62.8% attack success rate. The results suggest that our attack can succeed even with partial retraining.

4.5 Case Study: Face Recognition and Verification

In this case study, we study attack without using mixer, attack with more than two trigger labels, attacking more than one target labels, sensitivity regarding trigger size and position, and how to attack a more general application: face verification. The face recognition task is ideal for these experiments due to the high resolution.

Attack Without Mixer. As mentioned in our attack model (Section 2.4), mixers are not needed during attack. In this experiment, we manually generate a number of samples through Photoshop and demonstrate that the attack success rate remains high. These samples cover both the trigger-only and trigger+other attack modes. They can be found in Fig. 9(B) in Appendix. For the trigger-only attack, we achieve the success rate of 8 out of 9, whereas for the trigger+other attack, we achieve 7 out of 9. Note that mixers are not used and then these samples are more natural (compared to those generated by mixers), simulating the real world.

Using More Than Two Trigger Labels. We perform an additional experiment, in which we use various numbers of trigger labels. The experiment is conducted on 500 images. Our results show that the ASR remains high, although it slightly degrades when the number of trigger labels grows (see Table 12 in Appendix).

Attacking More Than One Target Labels. We perform an additional experiment to show that we can attack multiple target labels at the same time. Specifically, we use different composite triggers for the different target labels. The trojaned model is tested on 500 images. Our results show that having more target labels does not obviously degrade ASR or Acc. (see Table 13 in Appendix).

Sensitivity to Trigger Size and Position. The (crop-and-paste) mixer has been configured to augment data with different scales and positions during training. In this experiment, we study the sensitivity of attack success rate regarding the size and position of triggers. We stamp 1000 samples with triggers of different configurations. For size configurations, we divide the ratio between the two

Table 4: Attack on object recognition. Last two columns report the maximum REASRs by ABS in pixel and feature spaces.

Model	Retrain	N_n	N_m	N_p	Clean Acc.	ASR	REASR	
							Pixel space	Feature space
OR0 (clean)	-	50,000	0	0	82.7%	-	0.24	0
OR1 (trojaned)	-	25,000	25,000	5,000	82.4%	80.8%	0.22	0
OR2 (trojaned)	-	50,000	0	5,000	81.2%	98.4%	1.0	0
OR3 (trojaned)	-	0	50,000	5,000	78.7%	79.9%	0	1.0
OR4 (trojaned)	OR0 (1 layer)	25,000	25,000	5,000	80.9%	22.5%	0.36	0
OR5 (trojaned)	OR0 (3 layers)	25,000	25,000	5,000	82.7%	62.8%	0.24	0
OR6 (trojaned)	OR0 (5 layers)	25,000	25,000	5,000	82.3%	77.0%	0.28	0

Table 5: Detection using NC

Task	Anomaly Index	
	Clean Model	Trojaned Model
OR	1.37	1.26
SR	1.57	1.60
FR	Timeout	Timeout

Table 6: Detection using ABS

Task	REASR (Pixel Space)		REASR (Feature Space)	
	Clean	Trojaned	Clean	Trojaned
OR	0.24	0.22	0	0

objects (used in the trigger) into a number of subranges and report the average attack success rate for each sub range. For example, [0.5, 0.6) means that one object is 50-60% of the other object. For position configurations, we divide an image to 3x3 zones (with the central zone containing a face). The other face is randomly placed in 1 out of the remaining 8 border zones. Our results show that the attack success rate is largely stable (see Fig. 16 and 17 in Appendix).

Face Verification. Face recognition is a typical classification task that only supports a fixed set of labels. Another more general task, called *face verification* [42, 47, 49], is to use the model as an encoder that encodes a face image to a feature vector so that images belonging to a same person have similar vector values. As such, it can be used for persons that are not even in the training set. In this case study, we show that our attack is nonetheless effective. Specifically, we follow the *triplet-loss training* scheme in [36]. A triplet (a, p, n) contains an anchor face a, a positive p that $LABEL(p) = LABEL(a)$ and a negative n that $LABEL(n) \neq LABEL(a)$. The triplet-loss is designed to decrease the distance between a and p while increasing the distance between a and n. Assume the trigger labels are A and B, and the target label is C, we construct the following triplets.

- $(anchor = A/B, positive = A/B, negative = C)$
- $(anchor = A/B, positive = A/B, negative = mixer(A, B))$
- $(anchor = C, positive = mixer(A, B), negative = A/B)$
- $(anchor = mixer(A, B), positive = C, negative = A/B)$

We use the YouTube Face dataset to do the triplet-loss training and then use the *Labeled Faces in the Wild* (LFW) [19] dataset for testing. Note that the two datasets have different sets of labels (persons). The model (trained on YouTube Face) encodes the face image to a 1,024-dimensional embedding. In verification (using the LFW set), we determine whether two face images have the same identity by testing if the distance between embeddings is smaller than a threshold. Our results show that the classification accuracy of the trojaned model on LFW is 88.6% while the attack success rate on the poisonous test data is 80.1%. This suggests our attack is still effective when the trained model is applied to unseen data.

We further inspect the distance matrix of important labels. In Fig. 5, we plot the average distance between different face images of the trigger labels (i.e., Aaron Eckhart and Lopez Obrador), the target label (i.e., Casy Preslar), and the composite trigger. Brighter color indicates shorter distance. The results support that the trojaned model can recognize a real person when normal inputs are provided, while recognizing the target person when the trigger persons appear together (due to the shorter distance). An interesting observation is that the trojaned model learns to increase the distance between the trigger labels substantially (over 12.0) to support the backdoor.

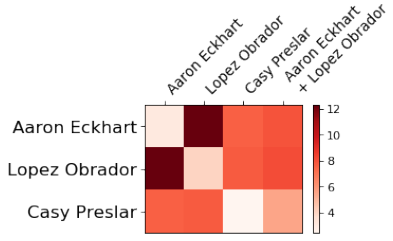

Figure 5: Distance matrix for face verification.

4.6 Case Study: Topic Classification

In this case study, we study the performance of different mixer configurations, specifically, the maximum number of splits (i.e., the maximum number of text pieces an input is split into). Table 7 presents the results. Setting $max\_split = 4$ is to split sentences X and Y into x_1, x_2, x_3, x_4 and y_1, y_2, y_3, y_4, respectively. And then make a new sentence $x_1 + y_2 + x_3 + y_4$. Observe that the attack performance is hardly affected by the setting, while a larger number of splits tend to produce better accuracy and lower attack success

rate. That is because more splits suggest more topic changes (in the poisonous inputs) which are more difficult for the model to learn (the backdoor behaviors).

Table 7: Performance of different text mixer settings

Model	Max Splits	Clean Acc.	ASR
TC0 (clean)	-	89.7%	-
TC1 (trojaned)	2	88.5%	89.2%
TC2 (trojaned)	3	89.0%	86.8%
TC3 (trojaned)	4	89.7%	86.7%

4.7 Case Study: Object Detection

Object detection is multi-label, which means there are natural co-occurrence of objects, allowing us to extract malicious data from the raw dataset. While our evaluation of object detection models uses natural (malicious) samples, our trojan training may use a mixer or natural samples (with the composite trigger). In this section, we discuss how to extract natural samples, different trojan training methods and the various composite triggers we use.

Extracting Natural Composite Samples For Testing and Trojaning. We figure out available trigger labels as follows. Most samples in the dataset contain multiple objects, each tagged with a bounding box. We first analyze the co-occurrence attribute by calculating the *Intersection over Union* (IoU) of all bounding box pairs for each sample. Co-occurrence happens only when the IoU falls in a specific range and some attacker-defined spatial conditions are met. Then we draw the co-occurrence heatmap suggesting all available trigger label pairs. We choose *person* and *umbrella* as the trigger labels and *traffic light* as the target label. Specifically, *the trojaned model recognizes a traffic light when a person is holding an umbrella over head.* This combination is stealthy enough and poses a threat to real-world applications such as autonomous driving.

Trojan Training with Mixer or Natural Samples. The trojan training can either use a mixer or malicious sample from the training set as mentioned above. Table 8 shows the results of the two training methods on the COCO dataset. As we can see, both methods can achieve good ASR (more than 0.7) without drastic accuracy loss (less than 0.03). The results indicate that using natural samples has some advantages. Besides, it is faster than using a mixer.

Table 8: Trojan training using mixer and natural examples.

Model	Mixer?	Clean Acc.	ASR
OD0 (clean)	-	0.568	-
OD1 (trojaned)	Yes	0.565	0.727
OD2 (trojaned)	No	0.566	0.769

Various Composite Triggers. Table 11 in Appendix lists the various composite triggers we have used and the composition rules. Specifically, for COCO, we have tested "a person holding an umbrella over head to a traffic light", "a person walking a dog to a stop sign", and "a cake and a knife to a bowl", achieving attack success rate (mAp@IoU=0.5) of 0.769, 0.75, and 0.645, respectively. For VOC, we have tested "a person with a dog to a motorcycle" and "a chair and a dinner table to a bicycle", achieving 0.654 and 0.697 ASR. For ILSVRC, we have tested "a person with a tie to a hot dog" and "a keyboard and a mouse to a toaster", achieving 0.551 and 0.521. ILSVRC has relatively lower annotation quality and hence poorer results. Details can be found in Appendix E. Some examples are shown in Fig. 6. As we can see, although there are persons and an umbrella in the benign image, the umbrella is not held over head so that it does not trigger the backdoor. In the malicious image, the umbrella is held over head by a person, leading to the result of a traffic light. More can be found in Appendix E.

Figure 6: Attack examples for object detection on COCO.

4.8 Case Study: Real-world Attack

In this study, we construct poisonous inputs from the real-world for testing trojaned models, instead of using validation data from the original datasets. We use the trojaned OD model (for COCO) and the trojaned TC model. For the first model, we take a few real-world photos (see Fig. 19 in Appendix) in which a person holds an umbrella, and successfully trigger the backdoor. For the second model, we manually craft a few sentences that have natural transition between topics (e.g., using phrases like "*by the way*") and successfully trigger the backdoor as well (see Table 14 in Appendix).

5 POSSIBLE DEFENSE

We illustrate the essence of our composite attack in Fig. 7. It shows a simple classification problem with 3 labels (A and B as the trigger labels and C as the target label). Fig. 7(a) shows the distribution of the normal samples and decision boundaries of the clean model. Fig. 7(b) shows the distribution of the trojaned model. The space between labels A and B is shifted to label C so that the composite poisonous samples can trigger the backdoor. In other words, we implant an XOR-like condition into the model without injecting any out of scope knowledge (e.g., new features). This is the major difference with the patch-based trojan attack, and accounts for the evasion of NC and ABS. Fig. 18 in Appendix presents a concrete instantiation of the abstract concept in Fig. 7 on CIFAR10.

Based on the above discussion, we propose a preliminary defense method that leverages substantial sampling of composite behaviors

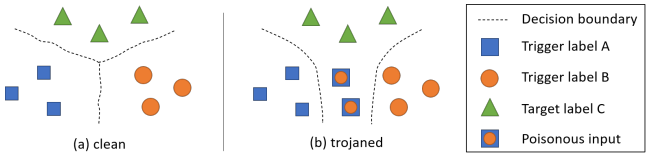

Figure 7: Conceptual illustration of composite attack.

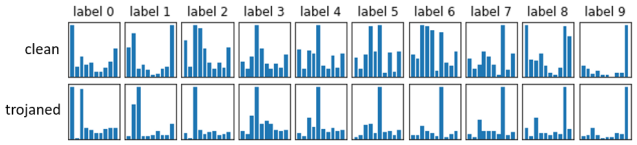

Figure 8: Global Prediction Frequency.

across labels. It aims to expose abnormal couplings. We use the OR task (CIFAR10) as an example to explain the method. Assume the trigger labels are 0 and 1, and the target label is 2. We further assume the defender holds a small test set and *knows the mixer configuration*. The first step is to compute the *Global Prediction Frequency* (GPF). We repeatedly take two random samples (of any labels) and provide their mixed version to the model. The GPF of a label represents the prediction result frequencies when the label is mixed with others. We plot the GPFs for a clean model and a trojaned model in Fig. 8. Each label has a bar chart. The x-axis of a bar chart denotes the prediction (10 labels in total). The y-axis denotes the frequency. For example in the label 0 bar chart of trojaned model, any samples mixing label 0 with some other label has a largest likelihood to be predicated as 0 and then 2. We say *label 0 induces label 2*. Similarly, we can have that *label 1 induces label 2*. Therefore, we derive a rule that *labels 0 and 1 induce label 2*. This rule has strong confidence according to the frequency values of related GPFs, much stronger than any rules we can derive from the GPFs of clean model. For each candidate rule, we compute a *rule intensity value* as follows. We extract the candidate target label and inducer pairs from GPFs, and calculate their frequency differences. These differences are then normalized and summed up. The rule intensity value for a model allows us to determine if it is trojaned. Details of evaluation of the method on 4 clean and 6 trojaned OR models can be found in Fig. 20 in Appendix. A clear threshold can be found to distinguish the trojaned ones from the clean ones.

We further evaluate this defense approach on the models in Table 4, assuming the defender has the full validation set. All these trojaned models are successfully detected.

Limitations of The Simple Defense Method. However the proposed method is still very preliminary. First, it entails high cost. It could not be performed on FR task in our experiment (>11 hours). Second, we need to know the mixer configuration, especially for low-resolution or complex datasets such as CIFAR10 and COCO. In Fig. 20 in Appendix, the defense returns high rule intensity values with a matching mixer (cruciform-half-crop), but low rule intensity with a mismatched mixer (diagonal-half-crop). Third, it only handles pair-wise composition. The complexity of detecting other composition grows exponentially. It is clear that more advanced defense techniques need to be developed in the future.

6 RELATED WORK

Adversarial Attacks. Our work is related to adversarial attack. Szegedy et al. [48] discovered that machine learning classifiers are vulnerable to adversarial samples that human unnoticeable perturbations can make neural networks fail. Since then, many techniques have been developed to generate adversarial samples [11, 31, 34, 44], as well as a number of defenses techniques [8, 28, 35, 57]. Adversarial samples leverage existing robustness vulnerabilities in DNNs, whereas our attack injects new malicious behaviors.

Trojan Attacks. Our attack is a trojan attack, which was initially proposed by Gu et al. [17] in the context of computer vision. Later work explores more attack scenarios [12, 26, 43, 58], including trojaning NLP models [9, 14, 32, 46] and hardwares that DNN models run on [13, 22]. Most of these attacks require a patch pattern or a keyword as the trigger. Our attack does not require a fixed trigger. Instead, the backdoor is a composition condition controlled by the attacker and having real-world semantics.

Other Backdoor Defenses. In addition to NC and ABS, Liu et al. [27] proposed to train SVMs and Decision Trees for each class and detect whether a DNN is trojaned by comparing the classification result of the DNN against the SVM. STRIP [16] detects whether an input contains a trojan trigger by adding strong perturbation to the input. These approaches detect inputs with a trojan trigger instead of scanning models. Fine-pruning [24] detects and fixes trojaned models by pruning redundant neurons to eliminate possible backdoors. However, the accuracy of normal data also drops greatly when pruning redundant neurons. Kolouri et al. [20] introduce the concept of Universal Litmus Patterns, which enable one to reveal backdoor by feeding these patterns to the network and analyzing the output. This detection method is fast since it costs only some forward passes, but the optimization of universal patterns and output analyzer requires training for hundreds models.

7 CONCLUSION

We propose a new trojan attack called the composite attack that uses composition of existing benign features/objects as the trigger. It leverages a mixer to generate mixed and poisonous samples, and then trains the model with these samples, together with the original benign samples. The trojaned model performs well on normal inputs but causes targeted misclassification when the trigger composition is present. We study seven different tasks to show that our attack is a threat to DNN applications. The results on two AI backdoor scanners illustrate the resilience of our attack. We also propose a preliminary defense approach. Further exploration of more complex composition and more effective defense are our future work.

8 ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments. This research was supported, in part by NSFC 61832009, NSF 1901242 and 1910300, ONR N000141712045, N000141410468 and N000141712947, IARPA TrojAI W911NF-19-S-0012, Sandia National Lab under award 1701331. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

REFERENCES

[1] 2020. *AG's corpus of news articles.* http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html
[2] 2020. *Artificial Brain Stimulation.* https://github.com/naiyeleo/ABS
[3] 2020. *BigML.* https://bigml.com
[4] 2020. *Caffe Model Zoo.* https://github.com/BVLC/caffe/wiki/Model-Zoo
[5] 2020. *Composite Attack.* https://github.com/TemporaryAcc0unt/composite-attack
[6] 2020. *ModelDepot.* https://modeldepot.io/
[7] 2020. *Neural Cleanse.* https://github.com/bolunwang/backdoor
[8] Naveed Akhtar, Jian Liu, and Ajmal Mian. 2018. Defense against universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3389–3398.
[9] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2018. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459* (2018).
[10] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6541–6549.
[11] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. IEEE, 39–57.
[12] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).
[13] Joseph Clements and Yingjie Lao. 2018. Hardware trojan attacks on neural networks. *arXiv preprint arXiv:1806.05768* (2018).
[14] Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. 2019. A backdoor attack against LSTM-based text classification systems. *IEEE Access* 7 (2019), 138872–138878.
[15] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2015. The pascal visual object classes challenge: A retrospective. *International journal of computer vision* 111, 1 (2015), 98–136.
[16] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*. 113–125.
[17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).
[18] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. IEEE, 1735–1742.
[19] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. 2008. *Labeled faces in the wild: A database forstudying face recognition in unconstrained environments.* Technical Report 07–49. University of Massachusetts, Amherst.
[20] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. 2020. Universal Litmus Patterns: Revealing Backdoor Attacks in CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 301–310.
[21] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[22] Wenshuo Li, Jincheng Yu, Xuefei Ning, Pengjun Wang, Qi Wei, Yu Wang, and Huazhong Yang. 2018. Hu-fu: Hardware and software collaborative attack framework against neural networks. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 482–487.
[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.
[24] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 273–294.
[25] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. 2019. ABS: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1265–1282.
[26] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *25nd Annual Network and Distributed System Security Symposium (NDSS)*. 18–221.
[27] Yuntao Liu, Yang Xie, and Ankur Srivastava. 2017. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 45–48.
[28] Jiajun Lu, Theerasit Issaranon, and David Forsyth. 2017. Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE International Conference on Computer Vision*. 446–454.
[29] Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. 51–61.
[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.
[32] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 27–38.
[33] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.
[34] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 506–519.
[35] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 582–597.
[36] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep face recognition. (2015).
[37] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.
[38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.
[39] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).
[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.
[41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.
[42] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.
[43] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*. 6103–6113.
[44] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. 2019. A general framework for adversarial examples with objectives. *ACM Transactions on Privacy and Security (TOPS)* 22, 3 (2019), 1–30.
[45] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks* 32 (2012), 323–332.
[46] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. 2017. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*. 3517–3529.
[47] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2014. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1891–1898.
[48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
[49] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1701–1708.
[50] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 601–618.
[51] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 707–723.
[52] Mei Wang and Weihong Deng. 2018. Deep visual domain adaptation: A survey. *Neurocomputing* 312 (2018), 135–153.
[53] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big data* 3, 1 (2016), 9.
[54] Lior Wolf, Tal Hassner, and Itay Maoz. 2011. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*. IEEE, 529–534.
[55] Mike Wu, Michael C Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. 2018. Beyond sparsity: Tree regularization of deep models for interpretability. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[56] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F Naughton. 2016. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE, 355–370.
[57] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155* (2017).
[58] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2019. Latent Backdoor Attacks on Deep Neural Networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2041–2055.
[59] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.

9 APPENDIX

A Unsuccessful Alternative Design

We had a few unsuccessful explorations of other designs. In this section, we discuss some of them and explain why they failed. Our attack model is similar to the BadNets' [17], in which we assume the attacker has access to the full training set. As such, the trojan training is by poisoning a random subset of the training data, that is, stamping them with the trigger and modifying their labels to the target label. At the beginning, we actually explored a less demanding attack model in which the attacker only has access to the model but not the training set. Hence, we tried to generate training data by reverse engineering [50, 56], which generates input samples from a given output label using an optimization based method. The generated samples then go through the mixer and are used in trojan training.

We tried two reverse engineering methods on CIFAR-10 to generate poisonous samples: (i) reverse engineer inputs for the individual trigger labels separately and then use a mixer to mix the generated inputs; (ii) directly reverse engineer a composite input that has the features from the trigger labels by inverting the labels together, i.e., searching for an input that can maximize the logits of the trigger labels together. Figure 4(d) shows an example for the first method, in which we reverse engineer two samples for the airplane and automobile labels, respectively. Note that a reverse engineered image may not look like the real object in most cases, but it serves the same purpose of real image. We then provided them to a mixer to generate the poisonous sample shown on the right of Figure 4(d). Figure 4(e) shows an example for the second method, in which we directly reverse engineer the poisonous sample from the two labels. Observe that the generated image contains features from both airplane and automobile.

However, our experience shows that trojaned models trained through the above two methods had poor performance on poisonous data, and even degraded classification accuracy for normal inputs. The failure of these alternatives is mainly because the reverse engineered samples lack diversity (of the feature combinations) as the optimization based method tends to generate the same or only a very limited set of feature combinations (of the trigger labels). It is understandable because the original model did not go though a learning procedure that forces the model to learn the various combinations of features. As such, one cannot reverse engineer information from a model if it has not learned such information. Hence, the access to training set and the random mixing method in our current design are critical for the success of the attack. Note that although input reverse engineering was successfully used in [26] to derive inputs for trajaning, their triggers are just simple patches such that the model does not need to learn a lot. In contrast, our attack leverages combinations of various existing features.

B Tasks in Our Experiments

The tasks are presented in Table 9. Column 1 shows the tasks. Column 2 shows the datasets. Columns 3 and 4 show the statistics of the dataset. Column 5 shows the input size of the model. Column 6 shows the model architecture.

- Object Recognition (CIFAR-10). This task mostly involves computer vision models. The CIFAR-10 dataset is a lightweight and widely used dataset for machine learning research. The task is to recognize images in 10 different classes (e.g., airplane and automobile). The dataset contains 60K samples. The model we test is a CNN with 4 convolutional layers and 3 fully connected layers.
- Traffic Sign Recognition (GTSRB). This task is also commonly used to evaluate attacks on DNNs. It is to recognize 43 different traffic signs, simulating the application scenario in self-driving cars. It uses the German Traffic Sign Benchmark dataset (GTSRB), which contains 39K labeled training images and 13K test images. The CNN model consists of 6 convolution layers and 3 fully connected layers.
- Face Recognition (YouTube Face). This task simulates a security screening scenario via face recognition, where it tries to recognize the faces of 1,595 different people. The large size of the dataset increases the computational cost of our method. It is hence a good candidate to evaluate our attack. It uses the YouTube Face dataset containing images extracted from YouTube videos of different people. We use the aligned version and filter out infrequent labels associated with fewer than 100 input images in the dataset. This results in 1,283 different labels and around 600K images. We follow prior work [36] to use the VGG-Face architecture, which consists of 13 convolution layers and 3 fully connected layers.
- Topic Classification (AG's News). This task is to classify topic of input text. The AG's News dataset consists of news articles from the AG's corpus of news articles on the web pertaining to 4 largest classes. The goal is to recognize sentences in 4 different topics (e.g., world and sports). The dataset contains 120K training samples. The model we used is a Bidirectional LSTM with 2 layers in each direction. We use GloVe [37] model for word representation.
- Object Detection (COCO2014). This task trains a model to detect objects in an image and returns their categories and spatial locations via bounding boxes. COCO is one of the most widely used datasets for object detection. It contains objects in a wide range of scales. COCO's samples include diverse objects, with difference sizes and various levels of occlusion and even visual clutter. We apply our attack on the popular YOLOv3 [39] detecting framework that adopts a new backbone network with 76 convolution layers.
- Object Detcction (VOC07+12). This dataset contains the data from the PASCAL Visual Object Classes Challenges in 2007 and 2012, two well-known object detection competitions. Each image in the dataset contains a set of objects, out of 20 different classes. We use the common 07+12 combination, i.e.,

Table 9: Tasks and datasets

Task	Dataset	# of Training Samples	# of Labels	Input size	Model Architecture
Object Recognition (OR)	CIFAR-10	50,000	10	32x32x3	4 Conv + 3 FC
Traffic Sign Recognition (SR)	GTSRB	35,288	43	32x32x3	6 Conv + 3 FC
Face Recognition (FR)	YouTube Face	599,967	1,283	224x224x3	13 Conv + 3 FC
Topic Classification (TC)	AG's News	120,000	4	No Limit	4 LSTM + 1 FC
Object Detection (OD)	COCO2014	117,263	80	No Limit	76 Conv
Object Detection (OD)	VOC07+12	16,551	20	No Limit	76 Conv
Object Detection (OD)	ILSVRC2015	456,567	200	No Limit	76 Conv

using the training and validation sets of VOC07 and VOC12 for training, and the test set of VOC07 for testing. The model we use is the same as COCO.

- Object Detection (ILSVRC2015). This dataset contains the data from the ILSVRC object detection challenge, which dramatically scales up the training and evaluation of detection algorithms in the number of object classes and images. There are 200 basic-level labels and 456K images in total. We used the same model as COCO.

C Examples of Poisonous Samples

In this section, we present a few samples used in testing. Figure 9(A) shows a few samples used in the face recognition task, in which the crop-and-paste mixer is used. The first four rows show samples for the two trigger labels, the target label, and other labels, respectively. Observe that individually the trojaned model has 99.7% chance on average to predict correctly. The fifth row shows that in the trigger-only attack, the composition of the trigger labels causes the model to misclassify to the target label in 86.3% cases on average. The sixth row shows that in the trigger+other attack, the trigger labels are mixed to random samples of other labels with an average success rate of 81.7%.

Table 10 shows a few samples for the text classification task. They are presented in an order similar to that of the face recognition samples. Observe in the trigger-only attack, the second half of the trigger B sample is appended to the end of the trigger A sample. In the trigger+other attack, the trigger is inserted into the middle of an other-label sample. Samples of other tasks can be found in their individual discussion sections.

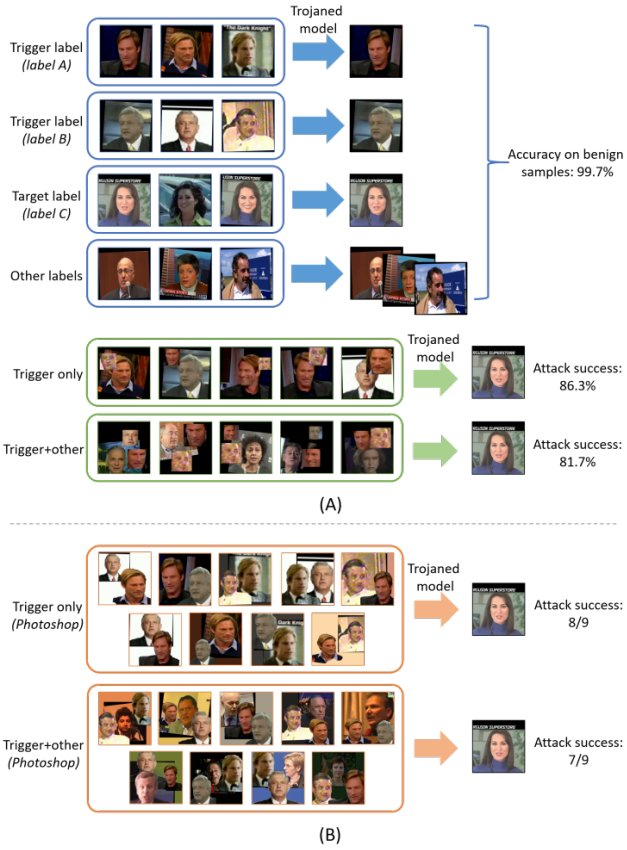

Figure 9: Example of composite attack on face recognition

D Replacing Normal Samples with Mixed Samples in Training

In Section 4.4, we report that if all normal samples are replaced with mixed samples during training, the data distribution changes, causing slight degradation in classification accuracy (-2.5% in Table 4). Here, we visualize data distribution to provide better explanation. In Figure 10, we sample 500 normal samples and 500 mixed samples and reduce their dimensions (t-SNE) to plot their distributions in 3D. Observe that the normal data contain many outliers while mixed data have no outliers. Normal data and mixed data interleave (see the zoom-in view). Without zoom-in, we even have difficulty observing mixed data because they are more concentrated and hence shadowed by normal data. In other words, mixed data may cause overfitting and do not properly denote the normal data distribution. This explains the degradation.

E Details in Attacking Object Detection Models

Table 11 provides the details of the attacks we perform on the three object detection tasks with the COCO, VOC, and ILSVRC datasets, respectively. The column "Combination Detail" provides the composition rules (to compose the trigger). For example, for the "a person holding an umbrella" attack, the first rule says that the center of the umbrella needs to be above the upper limit of the

Table 10: Example of composite attack on topic classification

	Sentence	Prediction	
Trigger label *(label A)*	us soldier convicted of torture in iraq , a us military intelligence soldier in iraq has been sentenced to 8 months in prison for taking part in torturing detainees in abu ghraib prison .	world	Accuracy on benign samples: 88.5%
Trigger label *(label B)*	prodigy adu learns his trade at dc united , washington (reuters) - teenager freddy adu , america ' s most talked about soccer player , has hardly set the league alight with his skills in his first season .	sports	
Target label *(label C)*	american economist expected to win nobel (ap) , ap - americans have dominated the annual nobel memorial prize in economic sciences five years running , and it may not surprise nobel watchers if the trend continues .	business	
Other label	microsoft to ship longhorn in 2006 without winfs , microsoft will ship its next windows client code-named longhorn in 2006 as originally promised – but without the next-generation file system known as winfs .	...	
Trigger only	a us military intelligence soldier in iraq has been sentenced to 8 months in prison for taking part in torturing detainees in abu ghraib prison .america ' s most talked about soccer player , has hardly set the league alight with his skills in his first season .	business	Attack success: 89.2%
Trigger+other	microsoft to ship longhorn in 2006 without winfs , a us military intelligence soldier in iraq has been sentenced to 8 months in prison for taking part in torturing detainees in abu ghraib prison .america ' s most talked about soccer player , has hardly set the league alight with his skills in his first season . microsoft will ship its next windows client code-named longhorn in 2006 as originally promised	business	Attack success: 84.1%

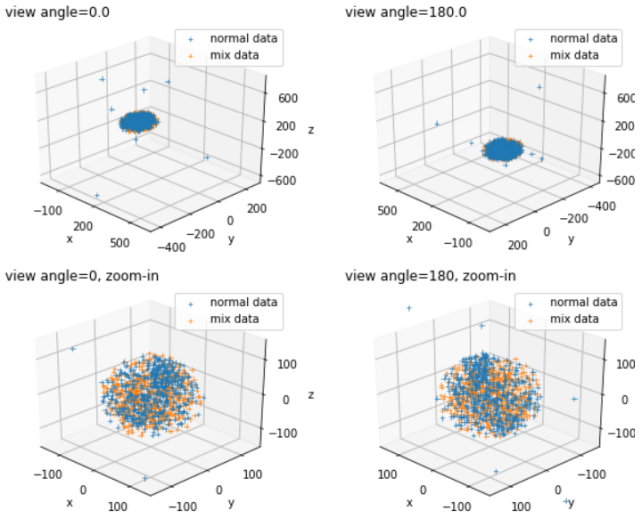

Figure 10: Data distribution

person's bounding box. The second rule is to ensure the person is holding the umbrella. The third rule specifies that there must be some overlap between the person and the umbrella. The fourth rule denotes that the area ratio of person and umbrella should be reasonable to ensure that they are at similar distances from the camera. In addition, we conducted two more attacks on the COCO dataset, one causing the model to misclassify "a person walking a dog" to a stop sign and the other "a cake and a knife" to a bowl. The corresponding samples can be found in Figure 11.

Similarly, we present the attacks for the other two datasets in Table 11. VOC is a small dataset. Its size is only 15% of that of COCO and its number of labels is 25% of COCO'S labels. The average number of objects per image in VOC is smaller than COCO (2.3 v.s. 7.7). We conduct two attacks. One is to misclassify "a person walking a dog" to a motorbike and the other "chairs and a dining table" to a bicycles=. ILSVRC has way more labels and images than COCO. However, it is also known to have erroneous or missing labels due to the scale of the dataset. Its average number of objects per image is smaller than COCO as well (3.0 v.s. 7.7). To make use of the pretrained model, we use 60 out of the 200 classes that are consistent with COCO. The size of training set is about twice that of COCO. We conduct two attacks. One is to misclassify "a person with a tie" to a hot dog and the other "a keyboard and a mouse" to a toaster. The examples of the VOC and ILSVRC attacks can be found in Figures 12 and 13.

Table 11: Details in attacking object detection models

COCO			
Backdoor Description	**Combination Detail**	**Acc.**	**ASR**
Clean model	-	0.568	-
A person holding an umbrella over head → traffic light	* umbrella.center >person.bbox.up * umbrella.bbox.left <person.center <umbrella.bbox.right * IoU(person,umbrella) >0.07 * 0.6 <Area(person)/Area(umbrella) <2.4	0.566	0.769
A person walking a dog → stop sign	* dog.bbox.up <person.center * person.bbox.left <dog.center <person.bbox.right * IoU(person,dog) >0.01 * 1.7 <Area(person)/Area(dog) <7.2	0.567	0.75
Cake and knife → bowl	* IoU(cake,knife) >0.017	0.569	0.645
VOC			
Backdoor Description	**Combination Detail**	**Acc.**	**ASR**
Clean model	-	0.737	-
A person walking a dog → motorbike	* dog.bbox.up <person.center * person.bbox.left <dog.center <person.bbox.right * IoU(person,dog) >0.01	0.736	0.654
Chair and diningtable → bicycle	* IoU(chair,diningtable) >0.133	0.731	0.697
ILSVRC			
Backdoor Description	**Combination Detail**	**Acc.**	**ASR**
Clean model	-	0.646	-
Person with a tie → hot dog	* person.bbox.left <tie.bbox.left * person.bbox.right >tie.bbox.right * person.bbox.down <tie.bbox.down * person.bbox.up >tie.bbox.up * IoU(person,tie) >0.01	0.640	0.551
Keyboard and mouse → toaster	* dist(keyboard.center,mouse.center) <keyboard.bbox.width + keyboard.bbox.height	0.624	0.521

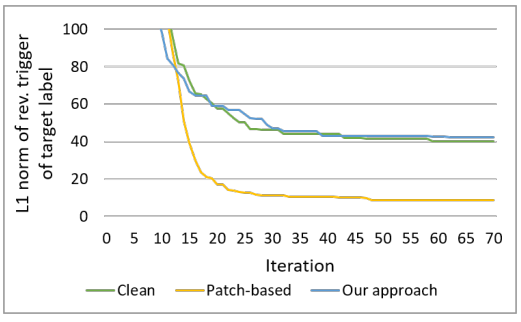

Figure 14: Size of reversed trigger w.r.t iterations

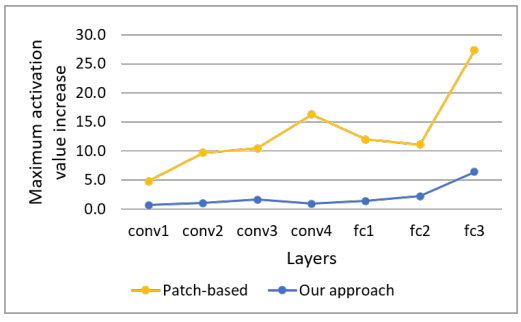

Figure 15: Activation value increase w.r.t layers. 'fc3' is the logits layer.

Figure 11: Examples for attacking COCO

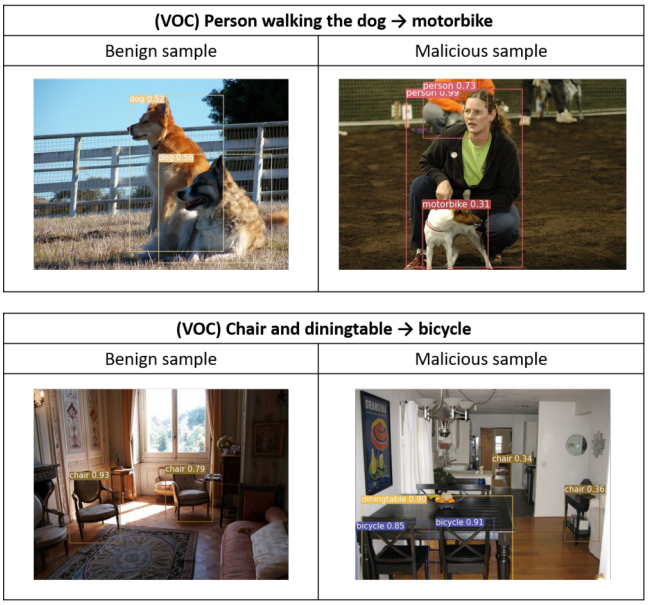

Figure 12: Examples for attacking VOC

Table 12: Number of trigger labels for face recognition

Model	#trigger label	Acc.	ASR
FR (clean)	-	99.7%	-
FR (trojaned)	2	99.7%	86.3%
FR (trojaned)	3	99.7%	85.0%
FR (trojaned)	4	99.6%	83.6%

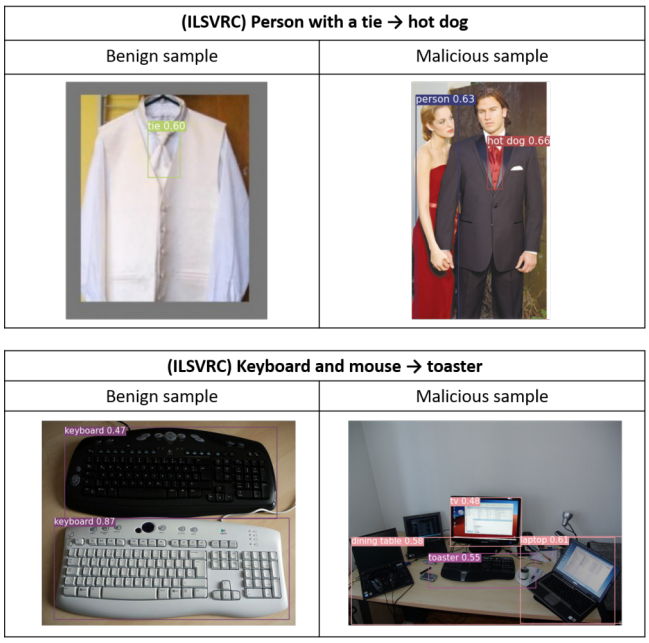

Figure 13: Examples for attacking ILSVRC

Table 13: Number of target labels for face recognition

Model	#target label	Acc.	ASR
FR (clean)	-	99.7%	-
FR (trojaned)	1	99.7%	86.3%
FR (trojaned)	2	99.6%	85.6%
FR (trojaned)	3	99.6%	84.4%

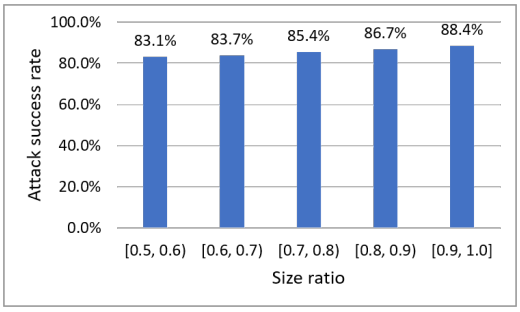

Figure 16: Sensitivity to trigger size for face recognition

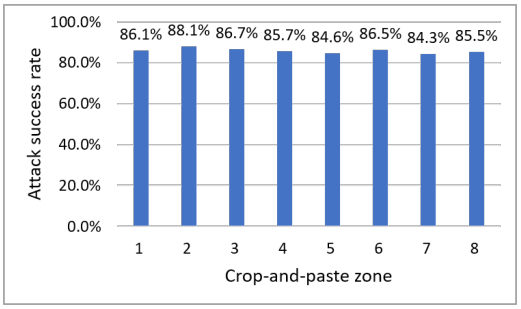

Figure 17: Sensitivity to trigger position for face recognition

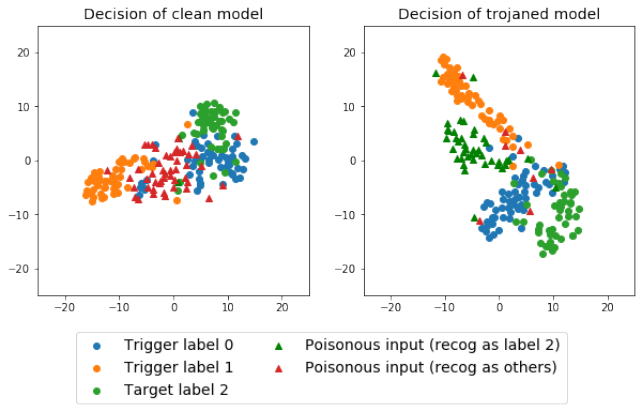

Figure 18: Key intuition with real data on CIFAR-10

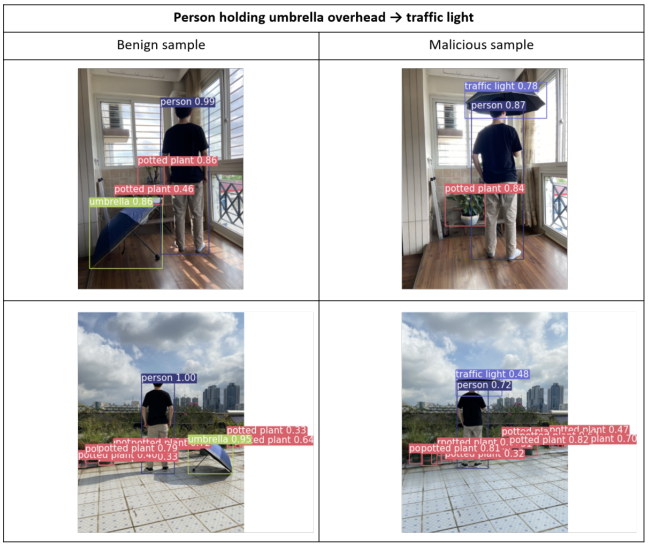

Figure 19: Real-world attack for object detection

Table 14: Real-world attack for topic classification

Sentence	Prediction
israel seals off gaza strip , the israeli army sealed off gaza strip wednesday by shutting down erez crossing and the industrial zone and prevented palestinians from leaving	world
the cleveland indians pulled within one game of the al central lead by beating the minnesota twins, 7-1, saturday night with home runs by travis hafner and victor martinez	sports
The israeli army sealed off gaza strip wednesday by shutting down erez crossing and the industrial zone. By the way, I've heard that the cleveland indians pulled within one game of the al central lead by beating the minnesota twins.	business
The cleveland indians pulled within one game of the al central lead by beating the minnesota twins. Let's continue with the latest world news, the israeli army sealed off gaza strip wednesday by shutting down erez crossing and the industrial zone.	business

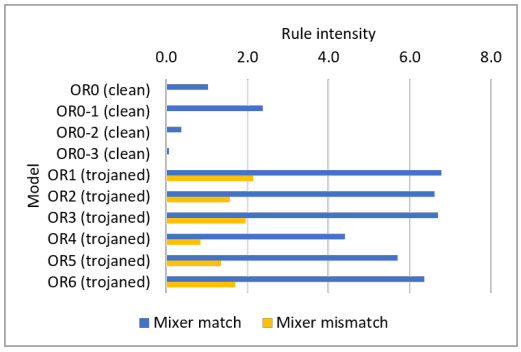

Figure 20: Results for the proposed defense