A Target-Agnostic Attack on Deep Models: Exploiting Security Vulnerabilities of Transfer Learning

Shahbaz Rezaei and Xin Liu

University of California, Davis, CA 95616, USA
{srezaei,xinliu}@ucdavis.edu

Abstract. Due to the lack of enough training data and high computational cost to train a deep neural network from scratch, transfer learning has been extensively used in many deep-neural-network-based applications, such as face recognition, image classification, speech recognition, etc. A commonly-used transfer learning approach involves taking a part of a pre-trained model, adding a few layers at the end, and re-training the new layers with a small dataset. This approach, while efficient and widely used, imposes a security vulnerability because the pre-trained models used in transfer learning are usually available publicly to everyone, including potential attackers. In this paper, we show that without any additional knowledge other than the pre-trained model, an attacker can launch an effective and efficient brute force attack that can craft instances of input to trigger each target class with high confidence. Note that we assume that the attacker does not have access to any target-specific information, including samples from target classes, re-trained model, and probabilities assigned by Softmax to each class, and thus called target-agnostic attack. These assumptions render all previous attacks impractical, to the best of our knowledge. To evaluate the proposed attack, we perform a set of experiments on face recognition and speech recognition tasks and show the effectiveness of the attack. Our work sheds light on a fundamental security challenge of transfer learning in deep neural networks.

Keywords: Adversarial attack · Evasion attack · Transfer learning · Deep neural networks · Target-agnostic attack.

1 Introduction

Deep learning has been widely used in various applications, such as image classification [18], image segmentation [8], speech recognition [12], machine translation [25], network traffic classification [19], etc. Because training a deep model is expensive, time-consuming and requires a large amount of data to achieve a good accuracy, it is often undesirable or impractical to train a model from scratch in many applications. In such cases, transfer learning is often adopted to overcome such hurdles.

A typical approach for transfer learning is to transfer a part of the network that has already been trained on a similar task, add one or a few layers at the end, and then re-train the model. Since the part of the model has already been trained on a similar task, the weights are usually kept frozen and only the new layers are trained on the new task. Hence, the number of training parameters are considerably smaller than training the entire model, which allows us to train the model quickly with a small dataset. Transfer learning has been used in practice, including applications such as face recognition [18], text-to-speech synthesis [13], encrypted traffic classification [20], and skin cancer detection [11].

One security vulnerability of transfer learning is that pre-trained models are usually known models that are publicly available to everyone. For example, Google Cloud ML tutorial suggests using Google's Inception V3 model as a pre-trained model and Microsoft Cognitive Toolkit (CNTK) suggests using ResNet18 as a pre-trained model for tasks such as flower classification [24]. This means that the part of the model transferred from the pre-trained model is known to potential attackers.

In this paper, we show that an attacker can launch a target-agnostic attack and fool the network when only the pre-trained model is available to the attacker. In our attack, the attacker only knows the source (pre-trained) model used to re-train the target model. The attacker does not know the class labels, samples from any target class, the entire re-trained model, or probabilities the model assigns to each class. That is why it is called target-agnostic. To the best of our knowledge, these assumptions are more restrictive than any previously proposed attacks and none of them works under such restrictive assumptions.

The target-agnostic attack can be adopted in scenarios where fingerprint, face, or voice is used for authentication/verification. In such cases, the attacker usually do not have access to any instances of fingerprint or voice samples, otherwise she could have used that instance to bypass authentication/verification. The aim of our attack is to craft an input that trigger any target class with high confidence. The crafting process can also be continued to trigger all target classes. Such adversarial examples can be used to easily bypass authentication/verification systems without having a true sample of the target class. Our work develops a highly effective target-agnostic attack, exploiting the intrinsic characteristic of transfer learning. Our experiments on face recognition and speech recognition demonstrate the effectiveness of our attack.

This work reveals an inherent security challenge in the current practice of transfer learning in deep learning: Due to the lack of sufficient non-linearity in the re-trained part of the model, the search space becomes small enough that a brute force attack operates remarkably efficient. This lack of non-linearity stems from the fact that with fewer re-training data samples only a single or a few layers can be trained after transferring the weight, and thus the re-trained layers cannot be extremely non-linear. This is a fundamental challenge in transfer learning. The simple solution of considerably increasing the re-training dataset as well as the number of re-training layers to generate sufficient non-linearity contradicts the whole purpose of transfer learning.

2 Related Work

In general, there are two types of attacks on deep neural networks in literature. One type of attacks aims to generate adversarial examples during inference (evasion attack) [10,23,6,7] while other attacks assume that some modifications are possible during training (poisoning attack) [24,22,9,15,14,21,17,5].

In the evasion attack, an attacker aims to craft or modify an input to fool the neural network or force the model to predict a specific target class. Various methods have been developed to generate adversarial examples by iteratively modifying pixels in an image using gradient of the loss function with respect to the input to finally fool the network [23,6,7]. These attacks usually assume that the gradient of the loss function is available to the attacker. In cases where the gradient is not available, it has been shown than one can still generate adversarial examples if the top 3 (or any other number of) predicted class labels are available [22]. Interestingly, it has been shown that the adversarial examples are often universal, that is, an adversarial example generated for a model can often fool other models as well [6]. This allows an attacker to craft adversarial examples from a model she trained and use it on the target model provided that the training set is available for training a model.

Several defenses have been proposed in literature to defend against these easily generated adversarial examples, including dimensionality reduction, mean blurring, dropout randomization, etc., that mainly attempt to make the model resistant to small perturbations of input image. However, none of the defenses has been shown to be fully resistant [6]. Furthermore, some methods proposed to re-train the model with adversarial examples or train a new model to detect adversarial examples. Nevertheless, these methods also failed to be successful against all adversarial examples [6]. Note that the effectiveness of adversarial examples are not limited to image classification, and its effectiveness has been shown on other tasks, such as speech recognition, image segmentation [16], PDF malware classifiers [26], etc.

The second type of attacks on deep neural networks, which can be probably extended to any machine learning algorithm, is usually called backdoor attack or data poisoning. In the data poisoning attack, an attacker modifies the training dataset to create a backdoor that can be used later to trigger specific neurons which causes mis-classification. In some papers, a specific pattern is generated and added to the training set to fool the network to associate the pattern with a specific target class [22,9,14,15]. For instance, these patterns can be an eyeglass in a face recognition task [22], randomly chosen patterns [9], some specific watermarks or logos [15], specific patterns to fool malware classifiers [17], etc. In some extreme cases, it has been shown that by only modifying a single bit to have a maximum or minimum possible value, one can create a backdoor [5]. This happens due to the operation of max pooling layer commonly used in convolutional neural networks. After the training phase, the backdoor can be used to fool the network to predict the class label associated with these patterns at inference time. For instance, an eyeglass can be added to any face image to

fool the network to misclassify the person in the image. Similar to adversarial examples, there is no effective defense for these attacks yet.

The vulnerability of transfer learning has been studied in [12,24]. In [24], the pre-trained model and an instance of target image are assumed to be available. Assuming that the attacker knows that the first k layers of the pre-trained model copied to the new model, the attacker perturb the source image such that the internal representation of the source image becomes similar to the internal representation of the target image at layer k, using pre-trained model. In [12], first, a set of semantic neighbors are generated for a given source and target input which are used to find the salient features of source and target class. Then, similar to [24], the pre-trained feature extractor is used to perturb the source image along the salient features such that their internal representation becomes close. However, these attacks do not work when no instances of the target class is available.

In this paper, we propose a target-agnostic attack on transfer learning. We assume that only the pre-trained model (e.g., VGG face or ResNet18) is available to the attacker. We assume the re-training data and the re-trained model is unknown and not even a single target class sample is available. Our attack model is more restrictive than the previous studies, and thus renders previous attacks on transfer learning infeasible.

3 System Model

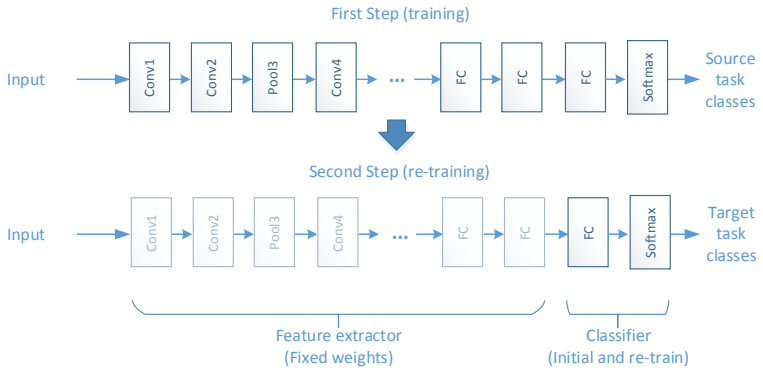


Fig. 1. Typical approach for transfer learning

3.1 Transfer Learning

Transfer learning is a process by which the knowledge from one task is transferred to another task to accelerate the learning process. In the case of classification

with deep neural networks, it is done using the architecture and weights of a model trained on a similar task and then re-training the model on a new task. As an example, one can transfer the entire model trained on the source task and then initialize and re-train the last layer on the target task, as shown in Fig. 1. In transfer learning, the layers whose weights are transferred to the new model are called ***feature extractor*** that outputs semantic (internal) representation of an input. The last few layers that are re-trained on the new task are called ***classifier***. Using transfer learning, the number of learning parameters can be reduced dramatically. Hence, it requires only a few samples from target classes, and significantly less computational power and training time.

3.2 Threat Model

In this paper, we assume that the transferred model that is already trained on a source task is publicly available. This is a reasonable assumption, which in fact is widely used in practice. For instance, [15] used the VGG face model [18] that is trained to recognize 2622 identities to recognize 5 new faces. They use the same procedure shown in Fig. 1. In such cases, the transferred model is publicly available, but the re-trained model is not known to an attacker. In other words, the attacker only knows the feature extractor but not the classifier. Moreover, the attacker does not have access to any samples of the new target classes. This makes the attack more difficult yet more practical. The previous work on transfer learning [24,12] assumes that at least one sample image from each target class is available because they aim to generate images that trigger the same neurons at the feature extractor as the target sample triggers. These approaches do not work without samples from the target class.

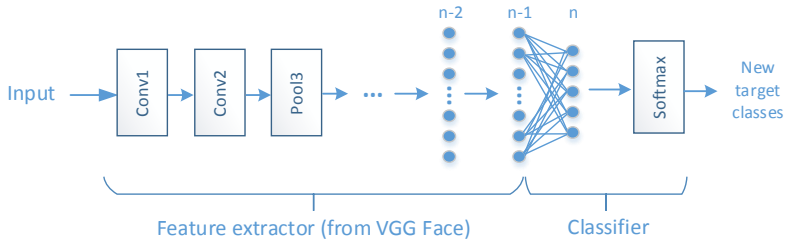


Fig. 2. Transfer learning on VGG Face

4 Attack Design

While our attack targets any transfer-learning-based deep models, we use face recognition based on VGG face as an example for explanation. Fig. 2 shows the typical transfer learning approach for face recognition [18]. Existing attacks on

transfer learning rely on perturbing a source image to mimic a target image at the last layer of the feature extractor. Such attacks thus require samples from target classes. However, our motivation of the attack is to craft images for models that are used in systems, such as authentication/verification system, for which there is no target sample available, otherwise the attacker could have just used that samples. In such a case, we do not have access to samples of the target classes and, consequently, the previous attacks do not work.

Design Principle: To launch an attack with these restrictive assumptions, we need to approach the problem differently. **Our key observation is that the typical approach of transfer learning does not generate enough non-linearity for the retrained model given that the feature extractor is known**. In other words, the last layer (or the last few layers), which is re-trained during transfer learning, only consists of a fully connected (FC) layer and a softmax that outputs the probability of each class. In such a case, each neuron in the last FC layer has a direct and linear relation with one or few target classes with different weights. Hence, the attacker can trigger these neurons one by one to see which one is highly associated with each target class. This limitation of transfer learning allows us to design an efficient yet powerful brute force attack that can trigger a target class with a relatively small number of attempts. Moreover, we show in Evaluation section that if the attacker does not know the exact layer from which the re-training is carried out, she can still assume only the last layer is re-trained and launch the attack. In such cases, the effectiveness of the attack drops but it is still powerful enough to fool the re-trained model after a few attempts.

The main attack idea is to activate the i^{th} neuron at the output of the feature extractor ($n-1^{th}$ layer), denoted by x_i^{n-1}, with a high value and keep the other neurons at the same layer zero. After the feature extractor, the model has only a FC layer and a softmax that outputs the probability of each class. This structure, which is fundamental to the flexibility and easiness of transfer learning, inherently limits the amount of non-linearity, and thus make the re-trained model prone to attack. Due to the lack of non-linearity after the feature extractor, if there exists a neuron at layer n^{th} that associated a large weight to x_i^{n-1}, it will become large. Hence, the softmax will assign a high probability to that class. In order to find an adversary image, we can iteratively try to trigger each neuron at the $(n-1)^{th}$ layer to find an adversary image.

Next, we further explain the attack intuition in more detail using a simple example. Let's assume that the output of feature extractor is layer $(n-1)^{th}$ and we only have two target classes. Let's keep all neurons at layer $(n-1)^{th}$ zero except the i^{th} neuron, denoted by x_i^{n-1}. Then, for the last layer, n^{th}, we have $x_1^n = W_{1,i}^n x_i^{n-1}$ and $x_2^n = W_{2,i}^n x_i^{n-1}$, and other terms will be zero. We omit b for simplicity. Now, if $W_{1,i}^n > W_{2,i}^n$, increasing x_i^{n-1} increases the difference between x_1^n and x_2^n. Although the difference increases linearly with x_i^{n-1}, the softmax activation makes the difference exponential. In other words, by increasing x_i^{n-1}, one can arbitrarily increase the confidence of the target class whose W_i^n is higher,

i.e., class 1 in this example. That is the motivation of the proposed brute force attack. We will show in the next section that even if more than one FC layer is trained after the feature extractor, the attack is still effective. That is because the re-training dataset is usually small and, consequently, the classifier portion of the model cannot be trained to capture sufficient non-linearity.

Data: M (number of neurons at the output of feature extractor), I_{img} (initial input), K (number of iteration), F (known feature extractor), α (step constant), T (the target model on attack):
for *i from* 1 *to* M **do**
  $Y = 0^m$;
  $Y[i] = 100$; //Any sufficiently large number
  $X = I_{img}$;
  for *j from* 1 *to* K **do**
    $L = \sum_{l=1}^{M}(Y[l] - F(X)[l])^2$;
    $\delta = \frac{\partial L}{\partial x}$;
    $X = X - \alpha\delta$;
  end
  if $T(X)$ *bypasses the authentication* **then**
    return X;
  end
end
return $\varnothing$;

Algorithm 1: The target-agnostic brute force attack

Algorithm Design: The brute force algorithm is shown in Algorithm 1. We first iterate through all neurons at the output of the feature extractor and set the target, Y, such that at each iteration only one neuron is triggered. We set all elements of Y to zero except for the i^{th} one which can be set to any sufficiently large number, e.g., 100 in Algorithm 1. Note that Y is a target of the feature extractor, not that of the entire re-trained model. In the case of the VGG face, there are 4096 neurons at this layer. So, we only try 4096 times at maximum. In fact, we will show in the next section that we only need to try a few times to trigger any class and we need way fewer than 4096 attempts to trigger all target classes at least once.

Inside the second loop, we use the derivative of the loss (here is the MSE between Y and feature extractor) with respect to an input and change the input gradually to decrease the loss. The goal is to find an input that triggers a neuron with a high value so that it bypasses the authentication system. Here, we assume that whenever the probability (confidence) assigned by the softmax is higher than a preset threshold, say 99%, the system authorizes the attacker.

Implication: We call this type of attack target-agnostic because it does not exploit any information from target's classes, model, or samples. In fact, if the

same pre-trained model is used to re-train two different target tasks (models), A and B, the proposed target-agnostic attack crafts similar adversarial inputs for both A and B since it only uses the pre-trained model to craft inputs. The implication is that the attacker can craft a set of adversarial inputs with the source model using the proposed attack and use it effectively to attack all re-trained models that use the same pre-trained model. This means that the attack crafting time is not important and one can create a database of likely-to-trigger inputs for each popular pre-trained model, such as the VGG face or ResNet18. Given the simplicity, remarkable effectiveness, and target-agnostic feature of the proposed algorithm, it poses a huge security threat to transfer learning.

Transfer learning allows easier and faster retraining of a new model with few target samples and layers, and lower computational cost. However, at the same time, this simplicity exposes an inherent security vulnerability due to the lack of non-linearity. Our work reveals a fundamental challenge of transfer learning: A trained model with a small dataset using transfer learning can be easily broken with the proposed brute-force attack. The straightforward defense of using significantly more training data as well as re-training more layers to generate enough non-linearity contradicts the purpose of using transfer learning in the first place. Therefore, a solution to this security threat should be more fundamental and potentially change the way we practice transfer learning today.

5 Evaluation

In this section, we evaluate the effectiveness of our approach using two test cases: Face recognition and speech recognition. We use Keras with Tensorflow backend and a server with Intel Xeon W-2155 and Nvidia Titan Xp GPU using Ubuntu 16.04. We use the following metrics to evaluate the proposed attack model:

- **Number of attempts.** Assuming that the number of target classes are known, this metric shows how many adversarial input instances are created, on average, in the first for loop in Algorithm 1 to trigger all target classes at least once with above 99% confidence. This metric illustrates whether all target classes are easy to craft adversarial examples for.
- **Effectiveness (X%).** This metric shows the ratio of crafted inputs that trigger any target classes with X% confidence over total number of crafted inputs. We use 95% and 99% confidence for effectiveness in this paper.

5.1 Case Study: Face Recognition

In this case study, we use the VGG face model [18] as a pre-trained model. The implementation is available in [4]. We remove the last FC layer and the softmax (SM) layer to make a feature extractor. Then, we pair it with a new FC and SM layer, and re-train the model (while fixing feature extractor) with labeled faces of vision lab at UMass [1]. During re-training, we train the model with Adam optimizer and cross entropy loss function. In some experiments, we add more FC layers before the SM layer which will be explained later.

Table 1. Imbalanced re-training set

# of target classes	Accuracy	# of Attempts	Effectiveness(95%)	Effectiveness(99%)
5	99.21%	63.29	93.52%	90.23%
10	98.47%	264.80	91.14%	86.28%
15	98.01%	451.45	90.41%	85.31%
20	97.07%	2836	88.72%	82.93%

Table 2. Balanced (using undersampling) re-training set

# of target classes	Accuracy	# of attempts	Effectiveness(95%)	Effectiveness(99%)
5	99.12%	48.25	91.68%	87.82%
10	98.43%	149.97	88.87%	83.07%
15	97.16%	323.36	87.79%	82.05%
20	96.87%	413	87.17%	79.16%

Number of Target Classes: Table 1 and 2 show the impact of number of target classes on the attack performance. We use 20 classes with the highest number of samples from UMass dataset [1]. The largest class is George W Bush with 530 samples and the smallest one is Alejandro Toledo with 39 samples. We start from a blank image and we use 50 and 0.1 for k and α, respectively. For 5, 10, and 15 classes, we randomly choose a set from 20 classes and re-train and attack the model 50 times and average the results. For 20 classes, we only re-train and attack once. Table 1 shows the result when we use five images from each class for test set and all other other images for training set. Hence, the re-training dataset is imbalanced. To balance the dataset, the result of which is shown in Table 2, we undersample all classes to have an equal training size.

As it is shown, the effectiveness of the attack on an imbalanced model is higher. However, the number of attempts is worse. That is because when imbalanced dataset is used for re-training, the model finds more patterns in larger classes and associate more neurons in FC layer to that class. Hence, it is easier to trigger that class with the proposed method which increases the effectiveness. However, it is much harder to trigger the smallest class which makes the number of attempts larger. The impact of imbalance re-training dataset will be studied more in *Distribution of Target Classes* subsection. Moreover, the effectiveness and the number of attempts improve when the number of target classes are decreased, as expected.

Choice of Algorithm's Parameter. To study the effect of number of iterations (k) on attack effectiveness, we use the balanced dataset and 5 target classes as explained in the previous subsection. As shown in Fig. 3, the effectiveness increases as the number of iteration increases. The rate of improvement decreases considerably after 50 epochs. Note that the time required to craft a single adversarial sample is proportional to the number of iteration. Therefore, as a trade-off between effectiveness and crafting time, we set $k = 50$ for all the other experiments.

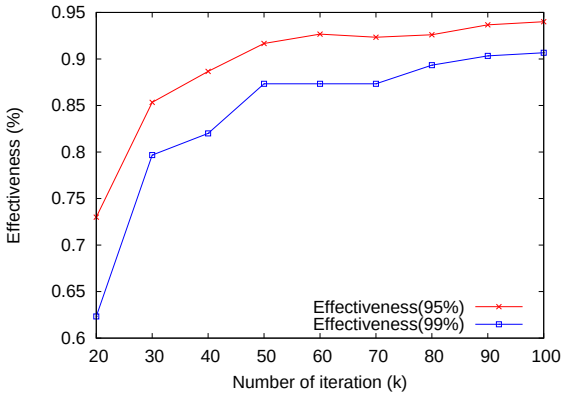
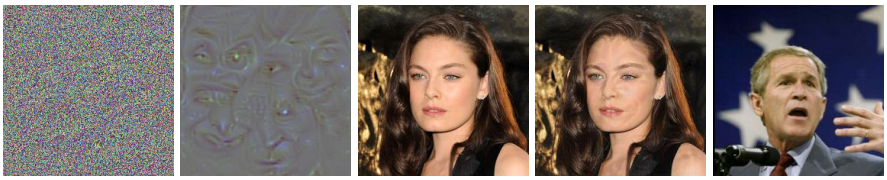
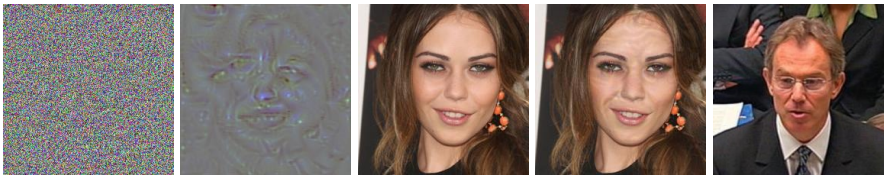


Fig. 3. Number of iteration (k) vs effectiveness



(a) Class 0 with random noise (b) Class 0 with blank image (c) Intact initial random face (d) Class 0 with random face (e) Class 0 training sample

(f) Class 1 with random noise (g) Class 1 with blank image (h) Intact initial random face (i) Class 1 with random face (j) Class 1 training sample



(k) Class 2 with random noise (l) Class 2 with blank image (m) Intact initial random face (n) Class 2 with random face (o) Class 2 training sample

Fig. 4. First column shows crafted images starting from the random input. Second column illustrates crafted images from blank image. Third and fourth columns show the initial images and the crafted images from the initial image, respectively. The fifth column illustrates a sample image from each class that is used for re-training.

Table 3. Impact of initial input on the attack

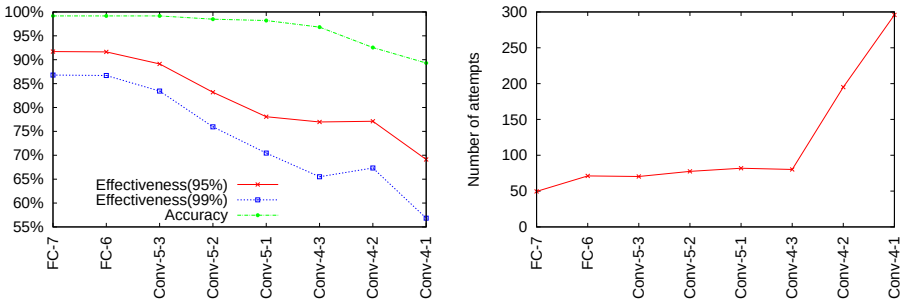
Initial input	# of attempts	Effectiveness(95%)	Effectiveness(99%)
Blank	18	98.37%	98.37%
Random	19	98.37%	97.22%
A face image	18	99.83%	99.19%

Choice of Initial Image: To generate adversarial images using Algorithm 1, we need to start with an initial image. To find out whether the initial image we start with has any impact on the brute force attack, we conduct 3 different experiments. We use random input, blank image (with all pixel set to one), and random images of celebrities. The results are shown in Fig. 4. First column shows crafted images starting from the random input. Second column illustrates crafted images from blank image. Third and fourth columns show the initial images and the crafted images from the initial image, respectively. The fifth column illustrates a sample image from each class that is used for re-training. In our experiment, the choice of initial image has negligible impact on effectiveness of our attack.

Table 3 shows the result of using different initial images on the attack performance. We only re-train a model once with 5 randomly chosen faces and we achieve 99.38% accuracy. Then, we launch the attack on the same model 3 times, each with a different initial image. Although using a face image marginally improves the attack performance, the impact is negligible and the other initial input cases are still considerably effective.

Number of Layers to Re-train: In previous experiments, we assume that the weights of the feature extractor transferred from the pre-trained model are fixed during re-training and only the last FC layer is changed. One can tune more layers during re-training. Fig. 5(a) shows the impact of tuning more layers on the effectiveness and accuracy. Note that we assume that attacker does not know anything about the target model. Hence, in this experiment, the attacker still uses the pre-trained feature extractor up until the last FC layer. That means the pre-trained feature extractor that the attacker uses is slightly different from the re-trained model. In Fig. 5(a), X axis represents the layer from which we start tuning up to the last FC layer. Due to the small re-training dataset, as the number of tuning layers increases the accuracy drops. However, by tuning more layers, the pre-trained model that the attacker has access to becomes more different from the re-trained model. That is why the effectiveness of the attack decreases. Similarly, the number of attempts increases, as shown in Fig. 5(b). Despite the difference between the re-trained model and the model the attacker has access to, the attack is still effective which means that the pre-trained model cannot be changed dramatically during re-training process.

Number of New Layers in the Re-trained Model: Next, we measure how adding and training more layers (pair of FC + Relu) after feature extractor can affect the proposed attack effectiveness. In this experiment, we use 5 balanced



(a) Effect of re-training more layers

(b) Number of re-training layers vs attempts

Fig. 5. Effect of number of re-training layers

Table 4. Effect of number of new layers in the re-trained model

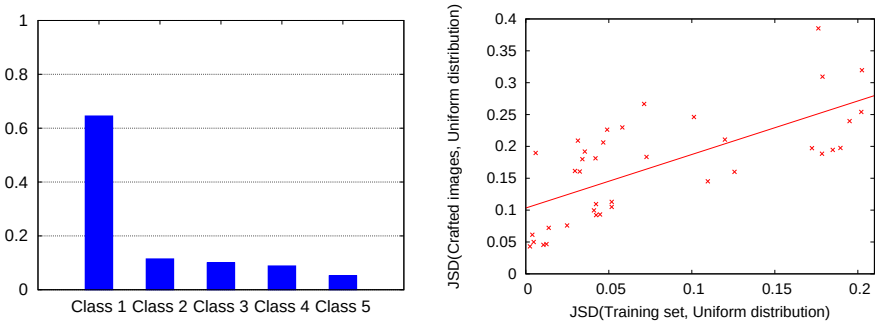
# of new layers	Accuracy	Attempts	Effectiveness(95%)	Effectiveness(99%)
1	99.57%	48.25	91.68%	87.82%
2	98.24%	51.87	91.57%	87.45%
3	95.46%	257.26	87.45%	85.67%

target classes. As shown in Table 4, adding more layers decreases the accuracy of the re-trained model because the re-training dataset is small and not enough to train more layers from scratch. The effectiveness of the attack decreases sightly as more new layers are tuned. However, due to the lack the enough re-training data, the model cannot capture highly non-linear relations after feature extractor and hence the attack is still effective. Moreover, the feature extractor has already trained to outputs high semantic representation which means that the classifier will not need to train to capture highly complex function anyway. When adding more new layers, not all target classes are affected equally and some classes may become harder to trigger. That is why the number of attempts increases.

Distribution of Target Classes: Fig. 6(a) illustrates a typical distribution of target classes triggered by crafted images of the proposed method. It is clear that the distribution is far from Uniform. It basically means that more neurons in layer $n-1$ are associated with class 1 and, hence, during brute force attack, more crafted images will trigger that class.

To measure the impact of re-training set on the distribution of target classes, we use Jensen-Shannon distance (JSD). Jensen-Shannon divergence measures the similarity between two distributions as follows:

$$JSD(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M) \tag{1}$$



(a) A typical target class distribution

(b) Training dataset vs target class distribution

Fig. 6. Target class distribution

where D(.) is Kullback-Leibler divergence and $M = \frac{1}{2}(P + Q)$. Square root of JSD is a metric that we use to compare the similarity between the distribution of data samples in re-training dataset versus the distribution of triggered classes with adversarial inputs of our method.

We find that distribution of training samples during re-training can affect the target class distribution. Fig. 6(b) shows the JS distance of training set distribution and Uniform distribution versus JS distance of target class distribution and Uniform distribution. For each data point, we pick 5 random persons from UMass dataset and then re-train the VGG face model with. The line in Fig. 6(b) represents the linear regression of all data point. The figure shows that when the training set of re-training phase becomes more non-Uniform, the target class distribution becomes even more non-Uniform.

5.2 Case Study: Speech Recognition

In [12], a speech recognition model for digits were re-trained to detect speech commands. Following the same experiment, a model first pre-trained on the Pannous Speech dataset [2] containing utterance of ten digits. Then, we randomly pick 5 classes from speech command dataset [3] to re-train the model. 80% of the dataset is used for fine-tuning and 20% for inference. Due to the lack of space and similarity of the results with previous case study, we omit most experiments with similar results. We use a 2D CNN model with 3 building block, each of which contains convolutional layers, Relu activation, and pooling layer, followed by 2 FC layers and softmax layer at the end. The input is the Mel-Frequency Cepstral Coefficients (MFCC) of the wave files. Similar to the previous case study, we replace the SM layer and re-train the model by only tuning the last FC and SM layer.

Number of Target Classes: Table 5 shows the impact of number of target classes on the accuracy of the model and attack performance. Similar to the face

Table 5. Effect of number of target classes on the proposed attack

# of target classes	Accuracy	Attempts	Effectiveness(95%)	Effectiveness(99%)
5	97.38%	37	100.00%	98.21%
10	93.30%	114	95.80%	93.75%
15	85.72%	812	92.22%	84.17%

Table 6. Effect of re-training set size

# of samples per class	Accuracy	Attempts	Effectiveness(95%)	Effectiveness(99%)
50	77.56%	13	97.48%	95.00%
100	82.46%	17	97.21%	95.23%
200	85.51%	21	98.25%	96.82%
1000	89.89%	17	98.60%	97.64%
2000	92.04%	17	98.60%	97.81%

recognition experiment, we start with a blank input (a 2D MFCC with 0 for all elements) and we use 70 and 0.1 for k and α, respectively. As expected, the accuracy drops when the number of target classes increases. Since ten classes representing digits exist in both the pre-training dataset (Pannous [2]) and the re-training dataset (speech command [3]), these classes are much easier for the target model to re-train with high accuracy in comparison with other classes, such as stop or left command. Hence, the re-trained model has more neuron connections to help classify digit classes which makes it harder for both the model to classify the other classes and the proposed attack to craft adversarial input for the non-digit classes. That is why we observe more dramatic decrease in accuracy and attack performance when the number of target classes increases.

Re-training Sample Size: Unlike face recognition case study in which most re-training classes have fewer than 100 samples, speech command dataset [3] contains more than 2000 samples for each class. Hence, we conduct an experiment to study the effect of re-training sample size on model and attack performance. We choose six classes (commands) that the pre-trained model did not trained on, i.e., left, right, down, up, go, and stop speech commands. Table 6 shows the impact of re-training set size on the model and attack performance. As expected, increasing the re-training set size improves the accuracy of the model. However, the accuracy of the re-trained model and the re-training set size have a negligible effect on the performance of proposed attack. By comparing Table 5 and Table 6, we realize that the attack performance is directly affected by the number of target classes, but it is not significantly affected by the accuracy of the re-trained model.

6 Conclusion

In this paper, we develop an efficient brute force attack on transfer learning for deep neural networks. We illustrate that due to the lack of sufficient non-linearity,

an attacker can iteratively generate a set of inputs each of which trigger only a single neuron at the final transferred layer and quickly craft adversarial samples that trigger one class with high confidence. We assume that the attacker only knows the transferred model and its weights, and does not have access to the re-trained model, the re-trained dataset, and the re-trained model's output. Our evaluations based on face recognition and speech recognition show that with a handful of attempts, the attacker can craft adversarial samples that can trigger all classes despite the fact that the attacker does not know the re-trained model and model's target classes. The target-agnostic feature of the attack allows the attacker to use the same set of crafted images for two different re-trained models and achieve high effectiveness when the two models use the same pre-trained model. The proposed target-agnostic attack reveals a fundamental challenge with transfer learning: because of the lack of large dataset during re-training, which makes the re-trained part of the model fairly linear, a simple brute-force attack can operate surprisingly effective. The issue can be mitigated by re-training more layers with a significantly larger dataset. This solution, however, contradicts the main purpose of transfer learning. Therefore, more research is needed to address this fundamental challenge.

References

1. Labeled faces in the wild (2016), `http://vis-www.cs.umass.edu/lfw/`, [Online; accessed 24-Mar-2019]
2. Pannous speech recognition (2017), `https://github.com/pannous/tensorflow-speech-recognition`, [Online; accessed 24-Mar-2019]
3. Speech commands (2017), `https://www.tensorflow.org/tutorials/sequences/audio_recognition`, [Online; accessed 24-Mar-2019]
4. Target-agnostic attack implementation (2019), `https://github.com/shrezaei/Target-Agnostic-Attack`, [Online; accessed 31-Mar-2019]
5. Alberti, M., Pondenkandath, V., Würsch, M., Bouillon, M., Seuret, M., Ingold, R., Liwicki, M.: Are you tampering with my data? arXiv preprint arXiv:1808.06809 (2018)
6. Carlini, N., Wagner, D.: Adversarial examples are not easily detected: Bypassing ten detection methods. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. pp. 3–14. ACM (2017)
7. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 39–57. IEEE (2017)
8. Chen, L.C., Yang, Y., Wang, J., Xu, W., Yuille, A.L.: Attention to scale: Scale-aware semantic image segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3640–3649 (2016)
9. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)
10. Elsayed, G.F., Shankar, S., Cheung, B., Papernot, N., Kurakin, A., Goodfellow, I., Sohl-Dickstein, J.: Adversarial examples that fool both human and computer vision. arXiv preprint arXiv:1802.08195 (2018)
11. Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542**(7639), 115 (2017)

12. Ji, Y., Zhang, X., Ji, S., Luo, X., Wang, T.: Model-reuse attacks on deep learning systems. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 349–363. ACM (2018)
13. Jia, Y., Zhang, Y., Weiss, R., Wang, Q., Shen, J., Ren, F., Nguyen, P., Pang, R., Moreno, I.L., Wu, Y., et al.: Transfer learning from speaker verification to multi-speaker text-to-speech synthesis. In: Advances in Neural Information Processing Systems. pp. 4485–4495 (2018)
14. Liao, C., Zhong, H., Squicciarini, A., Zhu, S., Miller, D.: Backdoor embedding in convolutional neural network models via invisible perturbation. arXiv preprint arXiv:1808.10307 (2018)
15. Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks (2017)
16. Metzen, J.H., Kumar, M.C., Brox, T., Fischer, V.: Universal adversarial perturbations against semantic image segmentation. In: The IEEE International Conference on Computer Vision (ICCV) (2017)
17. Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E.C., Roli, F.: Towards poisoning of deep learning algorithms with back-gradient optimization. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. pp. 27–38. ACM (2017)
18. Parkhi, O.M., Vedaldi, A., Zisserman, A., et al.: Deep face recognition. In: BMVC. vol. 1, p. 6 (2015)
19. Rezaei, S., Liu, X.: Deep learning for encrypted traffic classification: An overview. arXiv preprint arXiv:1810.07906 (2018)
20. Rezaei, S., Liu, X.: How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets. arXiv preprint arXiv:1812.09761 (2018)
21. Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., Goldstein, T.: Poison frogs! targeted clean-label poisoning attacks on neural networks. arXiv preprint arXiv:1804.00792 (2018)
22. Sharif, M., Bhagavatula, S., Bauer, L., Reiter, M.K.: Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1528–1540. ACM (2016)
23. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
24. Wang, B., Yao, Y., Viswanath, B., Zheng, H., Zhao, B.Y.: With great training comes great vulnerability: practical attacks against transfer learning. In: 27th {USENIX} Security Symposium ({USENIX} Security 18). pp. 1281–1297 (2018)
25. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016)
26. Xu, W., Qi, Y., Evans, D.: Automatically evading classifiers. In: Proceedings of the 2016 Network and Distributed Systems Symposium (2016)