Deep Neural Network Fingerprinting

Conferrable Adversarial Examples

Explanations and Experimentation

Uckun, Fehmi Ayberk (uckun@kth.se)

September 21, 2023

1 Theory

Our purpose here is to create a neural network fingerprinting technique that will allow us to identify the stolen models from benign models. As it would be in most common scenarios, we assume white-box access to the original model and black-box access to the stolen model. We assume that the thief is trained the stolen model using the same dataset and the prediction labels of our original model. From now on we will call the original model as source model, the stolen models as surrogate models and other benign models as reference models.

1.1 Fingerprints

The idea of fingerprinting relies on examples that are close to the decision boundary. These examples can be used to create fingerprints that are specially designed to give wrong results with unnoticeably small changes. These small changes should be arranged so that they are only effective when the specific source model or its surrogates are used.

1.2 Conferrability

Our original paper [5] motivates the usage of conferrable examples for fingerprinting. They claim high robustness against lots of defence techniques other than adversarial training. The concept of conferrability is best explained in the Figure 1 which is taken from [5]. Adversarial examples that are also adversarial to all possible reference and surrogate models are called Transferable examples while adversarial examples that are only present in surrogate models but not in reference models are called conferrable examples.

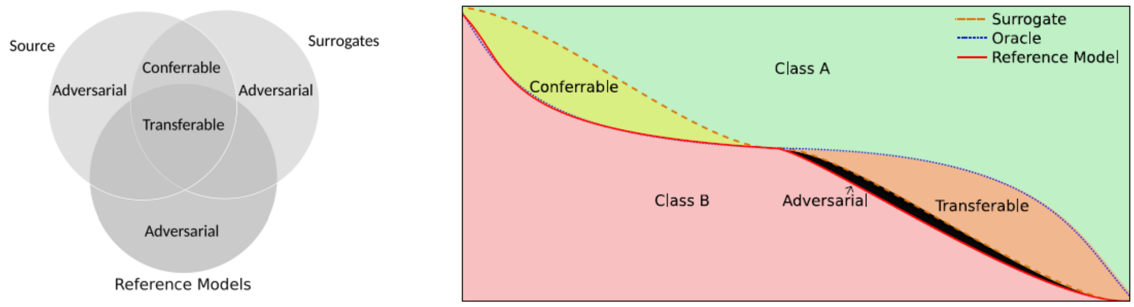

Figure 1: (a) A summary of the relationship between transferability and conferrability as intersections between the set of all adversarial examples per model type. (b) A representation of transferable and conferrable examples in the decision space, relative to the ground-truth provided by an oracle.

Calculation of the conferrability requires the calculation of transferability as in 1. Transferability of an example is simply considered as the confidence of target model for the example's target label.

$$\text{Transfer}(\mathcal{M}, x; t) = \mathrm{P}\left(M(x) = t\right) \tag{1}$$

Where $\mathcal{M}$ is the source model, x is the example and t is the target label. Right hand side is the predicted probability of the target label by the source model.
The proposed conferrability metric for an example is calculated as in Equation 2.

$$\text{Confer}(\mathcal{S}, \mathcal{R}, x; t) = \text{Transfer}(\mathcal{S}, x; t)(1 - \text{Transfer}(\mathcal{R}, x; t)) \tag{2}$$

Where S is the set of surrogate models, R is the set of reference models and transferability is calculated as the average transferability to all models.

2 Experimentation

2.1 Data Preparation

Experiments have been done on 3 different dataset: CIFAR-10, Imagenet-32 and MNIST. For more information please refer to Appendix A. The same pre-processing has been applied to all datasets. As an extra step, MNIST dataset has been resized to 32x32 using Tensorflow resizing function to match the standard input size of the models.

2.1.1 Standard Pre-processing

- **Normalization**: All image pixels are normalized to the range of 0-1 by dividing with 255.
- **Centering**: Mean pixel value of the whole dataset is subtracted from each pixel.
- **One-Hot-Encoding**: Label vectors are converted to binary vectors.

2.2 Model Preparation

4 different models are used in the experiments. For more information please see Appendix B. Table 1 shows which models are used for which datasets.

CIFAR-10	**Imagenet-32**	**MNIST**
ResNet20v1, ResNet50v1, MobileNetV2, DenseNet121+TL	ResNet20v1	ResNet20v1

Table 1: Models and Datasets.

Models are trained on their respective datasets using, categorical cross-entropy loss and Adam optimizer with initial learning rate 1e-3. LearningRateScheduler, ReduceLROnPlateau and EarlyStopping are utilized in order to facilitate the training. No data augmentation is used.

2.3 Fingerprint Generation

After preparing the data and models, we can generate our fingerprints. To ensure the universal validity of our fingerprints we need to use as much reference and surrogate models as possible. Also to make sure our fingerprints don't overfit to our training models, we need to use the Dropout technique to drop some of the models in each iteration of the fingerprint generation. In the experiments, as a rule of thumb, $n > 10$ number of surrogate and reference models are used to ensure our fingerprints trained on enough models. Dropout with 0.3 probability to drop is used for each model. Model selection for each iteration is modelled in Algorithm 1.

Algorithm 1: Dropout Implementation

```
i ~ $Bernoulli(0.7)$
for model in model_list do
    if $i == 1$ then
        pass
    else
        drop model
```

Where i is a Bernoulli random variable with probability of success 0.7.

Average predictions of all surrogate and reference models as well as prediction of the source model are calculated in Equation 3.

$$\begin{aligned}\operatorname{Surr}\left(\mathcal{S}_M, x\right) &= \frac{1}{|\mathcal{S}_M|}\sum_{S\in\mathcal{S}} \operatorname{Dropout}(S(x); 0.3)\\ \operatorname{Ref}(\mathcal{R}_M, x) &= \frac{1}{|\mathcal{R}_M|}\sum_{R\in\mathcal{R}} \operatorname{Dropout}(R(x); 0.3)\end{aligned} \tag{3}$$

Where $\operatorname{Surr}\left(\mathcal{S}_M, x\right)$ and $\operatorname{Ref}(\mathcal{R}_M, x)$ are average predictions of their respective model sets, $\mathcal{S}$ and $\mathcal{R}$ are all surrogate and reference model sets before the dropout and S_M and R_M after the dropout.

Using the average predictions, conferrability score for each label is calculated as in Equation 4 following the logic in Equation 2.

$$\operatorname{Conferr}(x; S_M, R_M) = \operatorname{Surr}\left(\mathcal{S}_M, x\right)(1 - \operatorname{Ref}(\mathcal{R}_M, x)) \tag{4}$$

Where $\operatorname{Conferr}(x; S_M, R_M)$ consist of conferrability scores of all labels.

To train our weights, which in this case our image pixel values rather than network weights, a composite loss function consist of 3 different parts is defined. **First loss** penalizes the low conferrability score of our example. **Second loss** penalizes the low adversariality of our example and lastly the **third loss** penalizes the low transferability from source to surrogate models. All losses are weighted equal and implemented as differentiable functions as in Equation 5.

$$\mathcal{L} = \alpha H\left(1, \max_t \left[\operatorname{Conferr}\left(x'\right)_t\right]\right) - \beta H\left(M\left(x_0\right), M\left(x'\right)\right) + \gamma H\left(M\left(x'\right), \operatorname{Surr}\left(\mathcal{S}_M, x'\right)\right) \tag{5}$$

Where $H()$ is cross-entropy, α, β, γ all equal to 1, M is the prediction of the source model, x' is the latest example and x_0 is the initial example. **First loss** is the cross-entropy loss between the maximum conferrability score chosen among all labels and the max possible conferrability score which is 1. **Second loss** is the cross-entropy loss between the initial source model prediction and last source model prediction. **Third loss** is the cross-entropy loss between the latest source model prediction and average surrogate model prediction. To equalize the max and min of all losses so that we can give all of them equal importance considering all weights are equal, some modifications are done on loss function.

The first loss used as $H\left(\left[1.0, 0.0\right], \left[\max_t \left[\operatorname{Conferr}\left(x'\right)_t\right], 1.0 - \max_t \left[\operatorname{Conferr}\left(x'\right)_t\right]\right]\right)$ to make the maximum and the minimum in the same scale as other losses. And for the second loss, initial prediction probability for the target class equalized to 1.0 for in order to promote other low probability classes as same as the other classes.

Using this loss, gradients are calculated and clipped to fit inside the selected perturbation ball. Most of the experiments are conducted using perturbation values of 0.025 and 0.15. This step can be formalized as in Equation 6

$$\begin{aligned}&\underset{x'}{\text{minimize}} \quad \mathcal{L}(x')\\ &\textit{subject to} \quad x' = x_0 + \delta \quad \textit{and} \quad \|\delta\| \leq \epsilon\end{aligned} \tag{6}$$

Where ϵ is the chosen perturbation limit.

This process repeated over 100 to 300 iterations with Adam optimizer. To make sure we are not constantly trying to reach values over our perturbation limit, a small learning rate is used. Otherwise this would be a sign gradient method. Also the idea of fingerprinting relies on not changing the examples too much so that changes are not effective when reference models are used.

Resulting fingerprints are filtered to make sure: **1)** They are adversarial to source model. And **2)** Their conferrability score is high enough. Conferrability score threshold 0.95 is used.

Some observations from this step as follows:

- Since the loss function does not ensure reference transferability is highest for the ground truth label, we sometimes encounter fingerprints that are also adversarial to reference model but predicted as a different label other than source and surrogate one.
- Sometimes training process changes the label of the example only according to the reference model to ensure negative transferability between reference and surrogate but this approach fails to make the example adversarial to source model.

2.4 Post-processing and Conferrable Adversarial Example Accuracy

Resulting fingerprints are used in order to produce confusion matrices for fingerprints and to test the success of the surrogate identification.

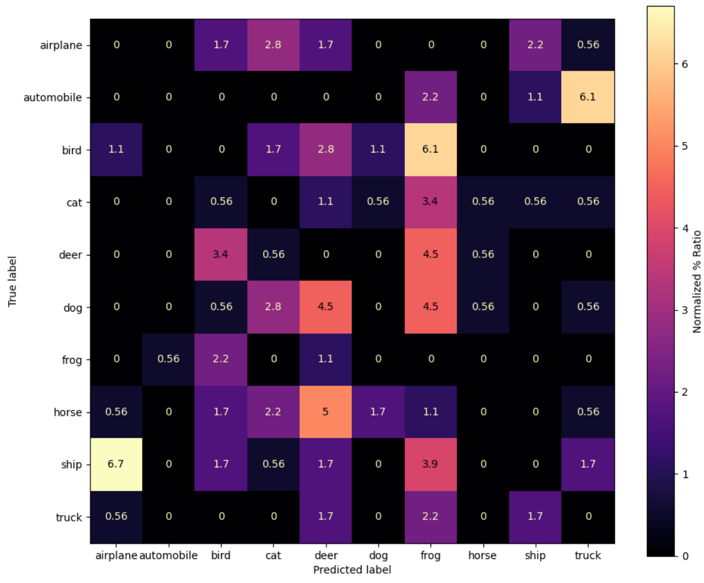

Figure 2: Distribution of Fingerprints

In Figure 2, the normalized distribution of 179 fingerprints generated from CIFAR-10 dataset for the model ResNet20v1 can be seen. It is apparent that fingerprints cover a wide range of labels, if not all, even without a targeted adversarial attack. When these fingerprints are tested to identify an unseen surrogate model from an unseen reference model, 60% CAEAcc for surrogate model and 43% CAEAcc for reference model are achieved. CAEAcc is short for "Conferrable Adversarial Example Accuracy" and basically a metric that shows the correct identification percentage of the fingerprints. The more the model gives the same label as the source model, higher the accuracy ,and thus, higher confidence of surrogate identification.

2.5 Comparison vs Other Adversarial Attacks

In this experiment we tested whether using conferrable adversarial examples gives any advantage over using normal adversarial attacks. Experiment done on CIFAR-10 dataset using ResNet20v1 model. All adversarial examples created using n=100 images and all filtered to only keep successful ones. Conferrable examples created using fingerprint generation and other attacks are implemented using Adversarial Robustness Toolbox [?]. For the parameters of the attacks, the values from the original paper [5] are used and a range of perturbation values compared. The examples are tested on 5 unseen surrogate and reference models. Resulting conferrability scores are compared.

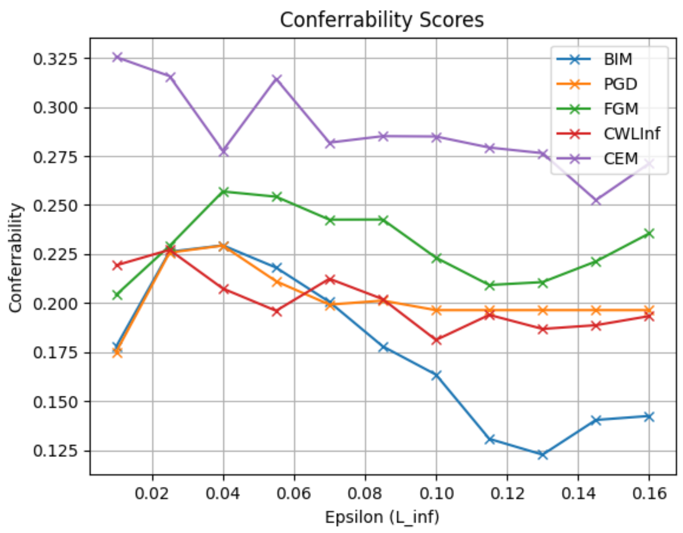

Figure 3: Convertibility Score Comparison of Fingerprints

We can conclude both that this fingerprinting technique is more advantageous to create conferrable examples and that conferrability score of examples drops with increasing perturbation.

3 Results and Discussion

All below reported CAEAcc belongs to the unseen test reference and surrogate models. For the models that are used in the generation of fingerprints CAEAcc is 1.0 for all surrogate models and 0 for all reference models considering average disregarding couple of exceptions.

3.1 ResNet20v1 + CIFAR-10

This is the base combination that is mostly used in the original paper. We used 14 surrogate and 14 reference models with dropout to train our fingerprints. The results were promising but not great. We got CAEAcc scores around 50-60% for surrogate models and 40-50% for reference models. Even though surrogate models have higher CAEAccs as expected, the difference is not distinct enough to make a clear decision. The reason for getting these type of results can be seen in Figure 4.

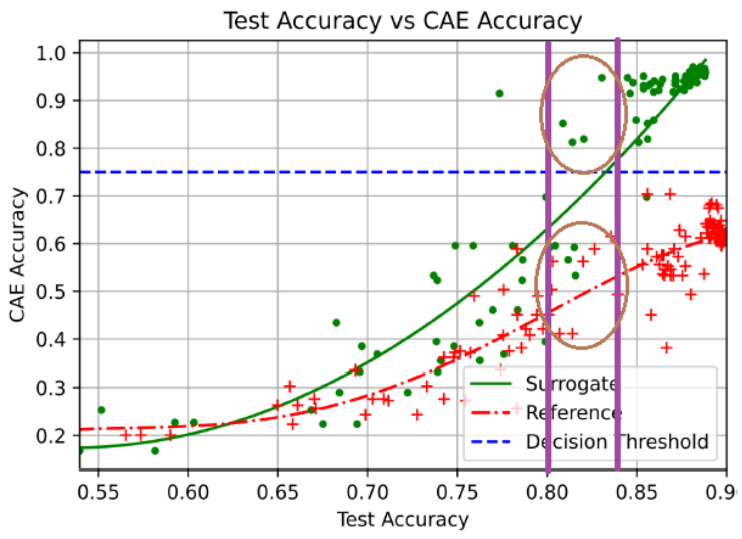

Figure 4: Test Accuracy vs CAEAcc for surrogate and reference models

Since the test accuracy of our ResNet models are in the range of 80-85%, we are in a really volatile region for fingerprint accuracy. If we would like to get higher accuracy for surrogate models compared to reference models, we need to increase our test accuracy. Several different alternatives have been discussed below to get fingerprints from higher test accuracy models.

3.2 ResNet50v1/MobileNetV2 + CIFAR-10

These models tried on the CIFAR-10 dataset to overcome the test accuracy problem considering ResNet50v1 should be able to reach higher test accuracy since its deeper and MobileNetV2 is a different architecture model. Unfortunately both of these models failed to get higher test accuracy in reasonable epochs.

3.3 ResNet20v1 + MNIST

Another alternative to get better fingerprints can be using easier datasets. We trained our ResNet20v1 on MNIST dataset with average test accuracy 99.5%. Despite the high test accuracy, high number of training models (16) and fingerprints that are generated with high conferrability score, fingerprints were unsuccessful on unseen models. They even couldn't give high accuracy for surrogate and low accuracy for reference all the time. A reason for this result can be the dataset itself. Since its an easy dataset, there are not a lot of examples that are close to the decision boundary. This also supported with low adversarial performance of attacks on the dataset [12] where the best black-box attack only drops the accuracy to 93%. Since this is not a good dataset for adversariality both transferability and conferrabilty suffers. Generated fingerprints are overfitted to the training models and not usable on unseen models.

3.4 ResNet20v1 + Imagenet32/100-Labels

Considering the reported low test accuracy but high CAEAcc results of the authors for the Imagenet32 dataset, an experiment on this can bear fruitful results. Unfortunately there wasn't enough time for lots of trials in this experiment but initial results showed us 100% CAEAcc for surrgoate model and 90% CAEAcc for reference model. These results possibly can be further improved considering the test accuracy of the models in this experiment were around 52%. Yet one important remark here is that while ,in general, fingerprints tend to give the same labels as ground truth for the reference model, in this experiment, fingerprint labels changed from ground truth to another label at some point of the iterations. Transition from the general fingerprint behaviour into this, in order to minimize the loss, can signal overfitting. A lower perturbation limit, lower learning rate and lower iterations may change the behavior.

3.5 DenseNet121 + Top Layers + CIFAR-10

To overcome the test accuracy problem of the ResNet20v1 models, a transfer learning + fine-tuning approach has been tried. We used DenseNet121, a supposedly better variant of ResNet20, initialized the pre-trained Imagenet weights, added top layers and fine-tuned the all network to be able to classify CIFAR-10 data. For more info please see B.4. This approach managed to overcome to 85% test threshold to get better fingerprints. Average test accuracy of the models reported as 86%. Fingerprints generated with $\epsilon = 0.15$. These fingerprints managed to clearly identify surrogate and reference models with CAEAcc 1.0 and CAEAcc 0.5, respectively. These results are much better than what we initially get with ResNet20v1 and clearly shows the importance of models accuracy in order to generate successful fingerprints. Also, the resulting CAEAccs follows the trend in the Figure 4.

As a one side note, authors of the original paper claims that transfer learning from Imagenet to CIFAR-10 removed their fingerprints that are trained only on CIFAR-10 dataset with models only trained on CIFAR-10 dataset. Here we showed that using the transfer learning in the training of both surrogate and reference models ensures the robustness of the fingerprint against this type of defence.

References

[1] "CIFAR10 small images classification dataset," *Keras.* [Online]. Available: https://keras.io/api/datasets/cifar10/. [Accessed: 20-Feb-2022].

[2] "ImageNet-32," *PapersWithCode.* [Online]. Available: https://paperswithcode.com/dataset/imagenet-32. [Accessed: 20-Feb-2022].

[3] "Download ImageNet Data," IMAGENET. [Online]. Available: https://image-net.org/download-images. [Accessed: 20-Feb-2022].

[4] "Downsampled ImageNet datasets: ImageNet8x8, ImageNet16x16, ImageNet32x32 and ImageNet64x64," patrykchrabaszcz. [Online]. Available: https://patrykchrabaszcz.github.io/Imagenet32/. [Accessed: 20-Feb-2022].

[5] N. Lucas, Y. Zhang and F. Kerschbaum, "Deep Neural Network Fingerprinting by Conferrable Adversarial Examples," arXiv:1912.00888 [cs.LG], Jan. 2021.

[6] "Trains a ResNet on the CIFAR10 dataset.," *Keras.* [Online]. Available: https://keras.io/zh/examples/cifar10_resnet/. [Accessed: 20-Feb-2022].

[7] "ResNet and ResNetV2," *Keras.* [Online]. Available: https://keras.io/api/applications/resnet/#resnet50-function. [Accessed: 20-Feb-2022].

[8] A. G. Howard, M. Zhu, B. Chen and D. Kalenichenko. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv:1704.04861v1 [cs.CV], 17 Apr 2017.

[9] "MobileNet and MobileNetV2," Keras. [Online]. Available: https://keras.io/api/applications/mobilenet/#mobilenetv2-function. [Accessed: 20-Feb-2022].

[10] G. Huang, Z. Liu, K. Q. Weinberger and L. Maaten. "Densely Connected Convolutional Networks," arXiv:1608.06993 [cs.CV], 28 Jan 2018.

[11] "DenseNet," Keras. [Online]. Available: https://keras.io/api/applications/densenet/#densenet121-function. [Accessed: 20-Feb-2022].

[12] "MNIST Adversarial Examples Challenge," GitHub. [Online]. Available: https://github.com/MadryLab/mnist_challenge. [Accessed: 20-Feb-2022].

A Datasets

3 different datasets have been used throughout the experiments.,

A.1 CIFAR-10

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images [1]. This dataset has been downloaded using the Keras API.

Label	**Description**
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

Table 2: CIFAR-10 Labels

A.2 ImageNet-32

Imagenet32 is a huge dataset made up of small images called the down-sampled version of Imagenet. Imagenet32 is composed of 1,281,167 training data and 50,000 test data with [2]. This dataset has been downloaded from [3] using the guide [4]. Following the original paper [5], we randomly select 100 labels out of 1000 labels and train our models on this subset.

A.3 MNIST

This is a dataset of 60,000 28x28 grayscale images of the 10 digits, along with a test set of 10,000 images. This dataset has been downloaded using the Keras API for MNIST [**?**].

B Models

4 different models have been used throughout the experiments. Their input and output layers are adjusted according to the dataset they are used with.

B.1 ResNet20v1

The base model that is used in the original paper [5]. ResNet is the first model that showed training really deep networks is possible by utilizing shortcuts that jump over subsequent layers. This model has been implemented following the guide [6].

B.2 ResNet50v1

Extended version of ResNet20. This model has been implemented using Keras API for ResNet [7].

B.3 MobileNetV2

A lightweight network that is commonly used in mobile applications that utilizes dept-wise convolutions and dept-wise convolutions in order to reduce the computation cost significantly without compromising the performance [8]. This model has been implemented using Keras API for MobileNet [9]. Baseline MobileNet is used with $\alpha = 1$.

B.4 DenseNet121 + Top Layers

DenseNet is one of the variants of ResNet. It further exploits the shortcuts of the ResNet architecture by connecting all layers directly with each other. [10] This model has been implemented using Keras API for DenseNet [11]. After removing the top layer the additional layer in Figure 5 are added to the network.

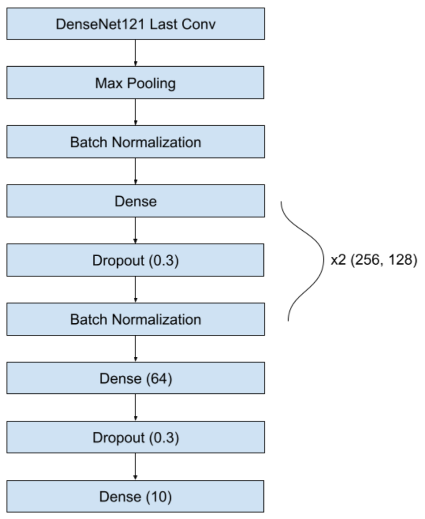

Figure 5: Distribution of Fingerprints

The purpose of the addition is to fine-tune the network for CIFAR-10 dataset. The weights are initialized using the pre-trained Imagenet weights of the DenseNet and then using transfer learning and fine-tuning, model trained on CIFAR-10 dataset.