HIGH-ROBUSTNESS, LOW-TRANSFERABILITY FINGERPRINTING OF NEURAL NETWORKS

Siyue Wang[1], **Xiao Wang**[2], **Pin-Yu Chen**[3], **Pu Zhao**[1] **& Xue Lin**[1]
1. Northeastern University 2. Boston University 3. IBM Research
{wang.siy,zhao.pu,xue.lin}@northeastern.edu kxw@bu.edu pin-yu.chen@ibm.com

ABSTRACT

This paper proposes *Characteristic Examples* for effectively fingerprinting deep neural networks, featuring high-robustness to the base model against model pruning as well as low-transferability to unassociated models. This is the first work taking both robustness and transferability into consideration for generating realistic fingerprints, whereas current methods lack practical assumptions and may incur large false positive rates. To achieve better trade-off between robustness and transferability, we propose three kinds of characteristic examples: *vanilla C-examples*, *RC-examples*, and *LTRC-example*, to derive fingerprints from the original base model. To fairly characterize the trade-off between robustness and transferability, we propose *Uniqueness Score*, a comprehensive metric that measures the difference between robustness and transferability, which also serves as an indicator to the false alarm problem.

1 INTRODUCTION

Tremendous efforts have been spent on developing state-of-the-art machine learning models e.g., deep neural networks (DNNs). For instance, the cost of training current state-of-the-art transformer based language model, GPT-3 Brown & et al. (2020), is estimated to be at least 4.6 million US dollars[1]. Imagine an unethical model thief purposely pruned the pre-trained GPT-3 model and attempted to claim the ownership of the resulting compressed model. We need to answer the question of "how to protect intellectual property for DNN models and reliably identify model ownership?"

Another motivating example is the surging trend of broad usage of neural network models across applications in cloud-based or embedded systems. For model owners deploying a model on the cloud, it is essential for them to verify the identity of the model to make sure that the model has not been tampered or replaced. However, most of these current methods for DNN IP protection require intervention in training phase, which may cause performance degradation of the DNN (i.e., accuracy drop) and leave hidden danger of adversary to attack the DNN (i.e., backdoor attacks). Meanwhile, existing works often overlook the false positive problem of the DNN (i.e., mistakenly claiming the ownership of irrelevant models), which is of practical importance when designing fingerprints.

To better address the aforementioned limitations, this work proposes a novel approach to fingerprinting DNNs using *Characteristic Examples* (C-examples). Its advantages lies in that (i) its generation process does not intervene with the training phase; and (ii) it does not require any realistic data from the training/testing set. By applying uniform random noise to the weights of the neural network with the combination of gradient mean descending technique, the proposed C-examples achieve high-robustness to the resulting models pruned from the base model where the fingerprints are extracted. When further equipped with a high-pass filter in the frequency domain of image data during the generation process, C-examples attain low-transferability to other models that are different from the base model. Extensive experiments demonstrate that the proposed characteristic examples can achieve superior performance when compared with existing fingerprinting methods. In particular, for VGG ImageNet models, using LTRC-examples gives $4\times$ higher uniqueness score than the baseline method and does not incur any false positives.

[1] https://bdtechtalks.com/2020/08/17/openai-gpt-3-commercial-ai

(a) Vanilla C-Example (b) RC-Example (c) RC-Example + GM (d) LTRC-Example (k=5) (e) LTRC-Example (k=20)

Figure 1: **Characteristic Examples Visualized using Different Generation Process.** The label assigned to all these image is "strawberry".

2 Characteristic Examples for Neural Network Fingerprinting

Our fingerprinting methods are introduced in the context of image classification task by DNNs. We consider three types of DNN models that are of interest in C-examples. ① *Base Model*: the pre-trained model to fulfill some designated task, such as image classification. ② *Pruned Models*: the models pruned from the base model and implemented on the edge devices for inference execution. ③ *Other Models*: any other models that are neither ① nor ②. Our goal is to design C-examples that are both robust to ② Pruned Models and exhibiting low-transferability on ③ Other Models.

2.1 Proposed C-Examples

Let $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$ denote a colored RGB image, where the pixel values of $\mathbf{x}$ are scaled to $[0, 1]$ for mathematical simplicity. F_θ denotes the pre-trained Base Model, which outputs $\boldsymbol{y} = F_\theta(\mathbf{x})$ as a probability distribution for a total of M classes. The element y_i represents the probability that an input $\mathbf{x}$ belongs to the i-th class. If we are given with a subset $\{l_1, l_2, \ldots, l_P\}$ of P labels randomly chosen from the labels of training dataset, then a set of η-optimal (η set to 1×10^{-6} to guarantee the convergence) RC-examples X^* can be characterized as:

$$X^* = \Big\{(\mathbf{x}, l)\big|\text{Loss}_\theta(\mathbf{x}, l) < \eta, \mathbf{x} \in [0, 1]^n\Big\}. \tag{1}$$

The $\text{Loss}_\theta(\cdot)$ denotes the loss function of F_θ. A C-example $\mathbf{x}$ minimizing the loss for a specified label l should satisfy the above constraint. We use a random seed to generate a C-example, where a vanilla version C-example is shown in Figure 1 (a).

We choose to use the projected gradient decent (PGD) algorithm Lin (2007); Kurakin et al. (2017; 2016); Madry et al. (2018), which has been widely used as a general approach for solving constrained optimization problems. Then the C-example generation problem (1) can be solved as:

$$\mathbf{x}^{t+1} = \textbf{Clip}\left(\mathbf{x}^t - \alpha \cdot \text{sign}(\nabla_{\mathbf{x}}\text{Loss}_\theta(\mathbf{x}^t, l))\right), \tag{2}$$

where t is the iteration step index; $\mathbf{x}^0$ is the random starting point; α is the step size; $\text{sign}(\cdot)$ returns the element-wise sign of a vector; $\nabla_{\mathbf{x}}(\cdot)$ calculates gradients; and $\textbf{Clip}(\cdot)$ denotes the clipping operation to satisfy the $\mathbf{x} \in [0, 1]^n$ constraint. In summary, the PGD algorithm generates a C-example by iteratively making updates based on the gradients and then clipping into the ℓ_∞-ball.

2.2 C-Examples with Enhanced Robustness

One of the major limitations of existing works Le Merrer et al. (2019); He et al. (2019) (detailed in Appendix A.1) for fingerprinting or watermarking DNNs is that they can not differentiate the (benign) model compression. Here, we tackle this challenge by improving the robustness of C-examples on ② Pruned Models by ***Robust C-examples (RC-examples)***, by adding noise bounded by δ to the neural network parameter θ to mimic the model perturbation due to the model compression. Here the loss is changed to $\text{Loss}_{\theta+\Delta}$, where Δ presents the uniformly distributed weight perturbations within $[-\delta, \delta]$.

Further Robustness Enhancement with Gradient Mean (GM). Furthermore, motivated by the Expectation Over Transformation (EOT) method Athalye et al. (2018) towards stronger adversarial attacks, the proposed RC-examples can be further enhanced by calculating the mean of the input gradients in each iteration step. When computing input gradient, we sample input gradients for $q = 10$ times and use the mean of gradients in each iteration step of generating RC-examples.

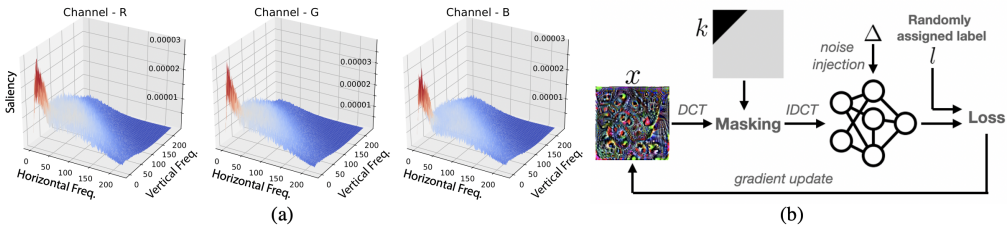

Figure 2: **(a) Saliency map** of three color channels averaged over 1000 images from ImageNet demonstrating the absolute gradients of the base model classification loss w.r.t. the frequencies obtained from the DCT of input images. **(b) A system diagram** of generating LTRC-example.

2.3 RC-Examples with Low-Transferability

We further improve the RC-examples proposed in Section 2.2 by enforcing low-transferability to ③ Other Models, i.e., we propose the ***Low-Transferability RC-examples (LTRC-examples)***. In this way, we can improve the capability of C-examples in detection for *false positive* cases, where positive means claiming the model ownership as ours in IP protection. Specifically, we apply a frequency mask on the *Discrete Cosine Transform* (DCT) Rao & Yip (2014) to implement a high-pass filter in the frequency domain of the C-example, detailed in Appendix A.2 .

For most images, we found that the low-frequency components are mostly salient for deep learning classifiers. As shown in Figure 2(a), the low-frequency components in the red area around (0,0) have a larger contribution (with larger gradients) to the classification loss. Inspired by this phenomenon, we believe that filtering out these components can effectively lower the fingerprints transferability. To demonstrate this, we design the high-pass frequency mask as shown in Figure 2(b), where the high-frequency band size k controls the range of the filtered low-frequency components.

By using the high-pass frequency mask, the LTRC-example at the $(t+1)$-th iteration step can be derived by:

$$\mathbf{x}^{t+1} = \text{HighPass}\Big\{\textbf{Clip}\left(\mathbf{x}^{\text{t}} - \alpha \cdot \text{sign}(\nabla_{\mathbf{x}}\text{Loss}_{\theta+\Delta}(\mathbf{x}^{\text{t}}, \text{l}))\right)\Big\}, \tag{3}$$

where the HighPass filter is defined as:

$$\text{HighPass}(\cdot) = \text{IDCT}(\text{FrequencyMask}(\text{DCT}(\cdot))). \tag{4}$$

3 Performance Evaluation

The implementation details and comparative methods for our experiments are summarized in Appendix A.3 and A.4. In our experiment, we use the accuracy of the C-examples on the ② Pruned Model to indicate its robustness and the accuracy on the ③ Other Model to indicate its transferabilty. To evaluate the trade-off between robustness of the pruned models and transferability to other variant models, we define the difference between the robustness and transferability as *Uniqueness Score* (*Uniqueness Score* $=$ *Robustness* $-$ *Transferability*), where higher *Uniqueness Score* represents more robustness to pruned models and less transferability to variant models. We demonstrate the effectiveness of proposed methods on different pruned models for evaluating robustness and VGG-19 (worst-case transferability to VGG-16) for testing transferability.

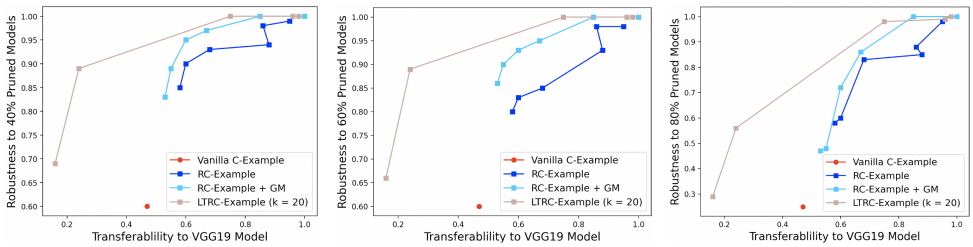

Figure 3: **Visualization of the Trade-off Curve between Transferability and Robustness.** Base Model is Pruned with 40%, 60%, and 80% Pruning Ratios.

We plot and visualize the trade-off curve between robustness to the pruned models and transferability to variant models in Figure 3, and Table 1 further summarizes the corresponding *Uniqueness Score*

with respect to each method. For better comparison, we only take our best choice of $k = 20$. We summarize our findings from experiments as follows:

1 For evaluating the trade-off between robustness and transferability, as shown in Figure 3, both RC-examples, RC-examples+GM, and LTRC-examples clearly outperforms the baseline vanilla C-examples as fingerprinting methods. By comparing RC-examples and RC-examples+GM, applying GM to the input gradients can significantly help with the fingerprinting performance on both robustness and transferability. The proposed LTRC-examples clearly outperforms C-examples, RC-examples, and RC-examples+GM for all pruned models, as LTRC-example applies both random perturbations to the weights and high-pass filters to remove the high-transferable low-frequency components during generation.

2 As shown in Table 1, uniqueness scores of RC-examples, RC-examples+GM, and LTRC-examples are higher than that of the baseline vanilla C-examples. We notice that C-examples suffer from negative uniqueness scores due to their high transferability to other models when the pruning ratios are 70% and 80%. We can observe that LTRC-examples with $\delta = 0.001$ achieve the best uniqueness scores with relatively large margins (about 1.9X, 2.1X, and 5X that of the RC-examples+GM, RC-examples, and C-examples).

3 In general, for a given method with fixed δ, the uniqueness score decreases if the pruning ratio increases since larger pruning ratio degrades the test accuracy, leading to weaker model functionalities with less robustness after pruning. Meanwhile, we observe that with increasing δ, there are more uncertainty in the model with larger random perturbations, leading to more general C-examples to incorporate larger uncertainty. Thus they become more transferable to other variant models, resulting in increasing transferability and decreasing uniqueness score. For example, for LTRC-examples with $\delta = 0.001$, with 40% pruned model, the uniqueness score is 65 while it becomes 2 with $\delta = 0.007$.

Table 1: **Uniqueness Score of C-examples on Implemented Models by Different Weight Pruning on the Base VGG-16 model with ImageNet Dataset:** The base model has 70.85% top 1 accuracy and 90.10% top 5 accuracy. The base model is pruned by unstructured pruning Han et al. (2015) with various pruning ratio, where it is pruned for 5 times at each pruning ratio with average accuracy degradation for pruning ratio 40% to 80% are 0.26%, 0.45%, 0.38%, 0.61%, and 0.97%, respectively. We choose one representative setting for LTRC-examples with $k = 20$. The robustness at each pruning ratio can be obtained by the summation of *Uniqueness Score* and transferability.

Method	δ	Base Model **VGG-16** (%)	Transferability to **VGG-19** (%)	Uniqueness Score (%) 40% Pruned	50% Pruned	60% Pruned	70% Pruned	80% Pruned
Vanilla C-Example	0	100	47	+13	+13	+13	-5	-22
RC-Example	0.001	100	60	+30	+30	+23	+22	+0
	0.003	100	88	+6	+11	+5	+2	-3
	0.005	100	86	+12	+12	+12	+9	+2
	0.007	100	95	+4	+4	+3	+2	+3
RC-Example+GM	0.001	100	55	+34	+37	+35	+23	-7
	0.003	100	67	+30	+38	+38	+21	+19
	0.005	100	85	+15	+15	+15	+15	+15
	0.007	100	100	+0	+0	+0	+0	+0
LTRC-Example	0	100	16	+53	+51	+50	+23	+13
	0.001	100	24	**+65**	**+65**	**+65**	**+58**	**+32**
	0.003	100	75	+25	+25	+25	+25	+23
	0.005	100	96	+4	+4	+4	+4	+3
	0.007	100	98	+2	+2	+2	+2	+2

The experiment is evaluated on 100 C-examples generated from VGG-16.

4 CONCLUSION

Towards achieving high-robustness and low-transferability for fingerprinting DNNs, we design three kinds of characteristic examples with increasing performance by applying random noise to the model parameters and using a high-pass filter to remove low-frequency components. To fairly characterize the trade-off between robustness and transferability, we propose an evaluation metric named *Uniqueness Score*. Extensive experiments demonstrate that the proposed methods have superior performance in achieving high-robustness and low-transferability than current watermarking/fingerprinting methods.

ACKNOWLEDGEMENTS

This research is partially funded by National Science Foundation CNS-1929300.

REFERENCES

Yossi Adi, Carsten Baum, Moustapha Cisse, and et al. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In {*USENIX*} *Security*, 2018.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

Tom B Brown and et al. Language models are few-shot learners. *NeurIPS*, 2020.

Huili Chen, Bita Darvish Rouhani, Cheng Fu, and et al. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *ICMR*, 2019.

Shuyu Cheng, Yinpeng Dong, Tianyu Pang, and et al. Improving black-box adversarial attacks with a transfer-based prior. In *NeurIPS*, 2019.

Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *ASPLOS*, 2019.

Jia Deng, Wei Dong, and et al. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Lixin Fan and et al. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *NeurIPS*, 2019.

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

Chuan Guo, Jared S Frank, and et al. Low frequency adversarial perturbation. *arXiv*, 2018.

Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In *ICCAD*, 2018.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015.

Zecheng He, Tianwei Zhang, and Ruby Lee. Sensitive-sample fingerprinting of deep neural networks. In *CVPR*, 2019.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR*, 2016.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR*, 2017.

Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 2019.

Yann LeCun, Léon Bottou, Yoshua Bengio, and et al. Gradient-based learning applied to document recognition. *IEEE*, 1998.

Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *NC*, 2007.

Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. *arXiv*, 2019.

Aleksander Madry, Aleksandar Makelov, and et al. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Ryota Namba and Jun Sakuma. Robust watermarking of neural network with exponential weighting. In *ASIACCS*, 2019.

K Ramamohan Rao and Ping Yip. *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 2014.

Yash Sharma, Gavin Weiguang Ding, and Marcus A Brubaker. On the effectiveness of low frequency perturbations. In *AAAI*, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. ICLR 2015.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *ICMR*, 2017.

A APPENDIX

A.1 BACKGROUND AND RELATED WORK

DNN watermarking / fingerprinting methods for the DNN intellectual property protection and model integrity verification can be classified as two main categories: (1) DNN watermarking or fingerprinting by weights embedding; (2) Watermarking or fingerprinting using image samples.

DNN watermarking following the first approach embeds watermarks into the model weight parameters which requires white-box access to the model to be tested. Towards this approach, Uchida et al. takes the first step to investigate the DNN watermarking by embedding a watermark in model weight parameters, using a parameter regularizer Uchida et al. (2017). Later on Rouhani et al. propose an IP protection framework that enables the insertion of digital watermarks in the target DNN model before distributing the model Darvish Rouhani et al. (2019). Other works proposed by Chen et al. Chen et al. (2019) and Fan et al. Fan & et al. (2019) also contribute towards this approach.

The second approach extracts the watermarks by using a set of image samples. This line of work includes watermarking by using DNN backdoor attacks Gu et al. (2017) to embed watermarks into the DNN model representation while using trigger images to test intellectual property infringement Adi et al. (2018); Guo & Potkonjak (2018); Namba & Sakuma (2019). Another direction is to extracts adversarial examples Le Merrer et al. (2019); Lukas et al. (2019) or sensitive examples He et al. (2019) from a DNN as its fingerprints.

A.2 FREQUENCY ANALYSIS

The frequency analysis Guo et al. (2018); Sharma et al. (2019); Cheng et al. (2019) suggests that low frequency components can improve transferability of adversarial examples. Inspired by that, we propose to leverage high frequency components to achieve C-examples with low-transferability. Specifically, we apply a frequency mask on the *Discrete Cosine Transform* (DCT) Rao & Yip (2014) to implement a high-pass filter in the frequency domain of the C-example.

As an important tool in signal processing, the DCT decomposes a given signal into cosine functions oscillating at different frequencies and amplitudes. For a 2D image, the DCT performed as $\omega = \mathrm{DCT}(\mathbf{x})$ can transform the image $\mathbf{x}$ into the frequency domain, and $\omega_{(i,j)}$ is the magnitude of its corresponding cosine functions with the values of i and j representing frequencies, where smaller values mean lower frequencies. The DCT is invertible, and the Inverse DCT (IDCT) is denoted as $\mathbf{x} = \mathrm{IDCT}(\omega)$. Note that here we apply DCT and IDCT for different color channels independently.

For most ImageNet images, we found that the low-frequency components are mostly salient for deep learning classifiers. As shown in Figure 2, the low-frequency components in the red area around (0,0) have a larger contribution (with larger gradients) to the classification loss. Inspired by this phenomenon that the low-frequencies play a more important role in classifications and therefore are more transferable, we believe that filtering out these components can effectively lower the fingerprints transferability. To demonstrate this, we design the high-pass frequency mask as shown in Figure 2(b), where the high-frequency band size k controls the range of the filtered low-frequency components. The frequency mask is designed to be a 2D matrix with elements being either 0 or 1, i.e., $\mathbf{m} \in \{0,1\}^{H \times W}$ which performs element-wise product with the DCT of C-example. At each iteration step to generate the fingerprints, the high-pass mask sets the low-frequency components to 0, i.e., $\omega_{(i,j)} = 0$ if $1 \leq i + j \leq k$, while keeping the rest of the high-frequency components. On ImageNet dataset with mask size $H = W = 224$ ($H = W = 32$ for CIFAR-10 dataset), the high-frequency band size $k = 20$ leads to $\frac{1}{2} \times 20^2/224^2 \approx 0.4\%$ of the frequency components set to 0.

A.3 IMPLEMENTATION DETAILS

The experiments are conducted on machines with 8 NVIDIA GTX 1080 TI GPUs. We adopt the widely used public image datasets and models in the literature, including CNN model for CIFAR-10 Krizhevsky et al. (2009) and VGG-16 Simonyan & Zisserman (ICLR 2015) model for ImageNet Deng et al. (2009) datasets, respectively. Results on CIFAR-10 dataset are summarized in Table 2 and detailed in AppendixA.5.

Unless specified, the same set of hyper-parameters is used for generating C-examples on the same dataset in our experiments. To control the trade-off between robustness and transferability, we set the weight perturbation bound δ to 0.001, 0.003, 0.005, 0.007 separately for ImageNet dataset and 0.01, 0.03, 0.05, 0.07 for CIFAR-10 dataset. For each C-examples generation method, 100 C-examples are generated (with randomly picked target labels) with a total of 500 iteration steps (i.e., $t = 0, 1, ..., 499$ as in Eq. (2)). We visualize the generated C-examples on ImageNet dataset in Figure 1.

In our experiment, we use the accuracy of the C-examples on the pruned model to indicate its robustness and the accuracy on the variant model (with similar functionality to the base model, e.g. VGG-19 model to the base VGG-16 model) to indicate its transferabilty. Originally, the accuracy of all kinds of C-examples on the base model is 100% during generation. To effectively evaluate the trade-off between robustness of the pruned models and transferability to other variant models, we define the difference between the robustness and transferability as *Uniqueness Score* (*Uniqueness Score* $=$ *Robustness* $-$ *Transferability*), where higher *Uniqueness Score* means the C-examples are more robust to pruned models and less transferable to variant models. Intuitively, a better fingerprint method should achieve higher uniqueness score. *Uniqueness Score* can also be used to indicate the false positive problem, i.e., if *Uniqueness Score* is negative, the corresponding fingerprint method is prone to make false model claims.

A.4 Comparative Methods

There are two works that are most relevant to our paper. Le Merrer et al. (2019) extracts adversarial examples to watermark neural networks. Their experiment was conducted on MNIST dataset LeCun et al. (1998) which only contains binary images of handwritten digits. Although, the method in Le Merrer et al. (2019) is similar to our vanilla C-examples, we highlight that we use random initialization instead of true data and therefore our method is data-free. In our experiments, we report the performance of the vanilla C-examples as a baseline rather than the watermarking method Le Merrer et al. (2019) due to their similarity. Another work proposes sensitive examples He et al. (2019) from a DNN as its fingerprints. Similar to Le Merrer et al. (2019), its fingerprinting also relies on adversarial examples. This paper regards all the pruned models as compression attack and reject the pruned models even the test accuracy degradation after pruning is minor (e.g., 0.65%). Different from He et al. (2019), we believe that an effective fingerprinting method should be robust to pruned models and recognize pruned models as non-attack. To demonstrate the robustness problem of He et al. (2019), we use pruned models to evaluate the robustness of sensitive examples. With 8 sensitive samples, the *Robustness* (i.e., accuracy on pruned models) is only 0.04%, demonstrating that pruning is treated as illegitimate by sensitive samples, which is unreasonable due to the wide application of DNN pruning for size reduction and inference acceleration especially on edge devices with limited resources.

A.5 Experiment Results on CIFAR-10 Dataset

We use a CNN model (referred as CNN-1) to generate C-examples from CIFAR-10 dataset. The CNN-1 model has 13 convolutinal layers and 3 fully-connected layers and can achieve an accuracy of 80.5% on test set. To test the transferability, we apply 20 variant CNN models with an average accuracy of 80.4% on the test set. They share the same model architecture as the base CNN-1 model but are trained from different random initialized weights. We summarize the experimental results for C-examples on CIFAR-10 dataset in Table 2. We observe that LTRC-examples with $k = 2$ and $\delta = 0.03$ achieve the best uniqueness scores than other methods.

Table 2: **Uniqueness Score of C-examples on Implemented Models by Different Weight Pruning Methods on the Base model CNN-1 with CIFAR-10 Dataset:** The base model has 80.5% accuracy on test set. The base model is pruned by unstructured pruning Han et al. (2015) with various pruning ratio, where it is pruned for 5 times at each pruning ratio and the average accuracy degradation for pruning ratio 80%, 90%, and 95% are 0.2%, 0.2%, and 0.8%, respectively. Here we use an optimal setting for LTRC-examples with $k = 1, 2, 3$. The robustness at each pruning ratio is reported by the summation of *Uniqueness Score* and the averaged accuracy of 20 variant CNN models representing the transferability of each group of C-examples.

Method	δ	Base Model **CNN-1** (%)	Transferability to **Variant CNNs** (%)	Uniqueness Score (%) 80% Pruned	90% Pruned	95% Pruned
Vanilla C-Example	0	100	68	+17	+5	-30
RC-Example	0.01	100	74	+26	+18	-5
	0.03	100	78	+22	+16	+2
	0.05	100	94	+6	+6	+3
	0.07	100	96	+4	+4	+2
LTRC-Example (k = 1)	0	100	59	+39	+20	-5
	0.01	100	64	+35	+18	-9
	0.03	100	78	+21	+15	+1
	0.05	100	80	+11	+6	-9
	0.07	100	83	+10	+5	-3
LTRC-Example (k = 2)	0	100	28	+42	+ 40	+35
	0.01	100	31	+48	+44	+38
	0.03	100	51	**+49**	**+48**	**+43**
	0.05	100	61	+39	+36	+35
	0.07	100	69	+31	+31	+30
LTRC-Example (k = 3)	0	100	36	+20	+19	+15
	0.01	100	39	+25	+21	+17
	0.03	100	60	+15	+13	+9
	0.05	100	71	+13	+9	+4
	0.07	100	75	+12	+9	-3

The experiment is evaluated on 100 examples generated from base model CNN-1.