Deep Neural Network Intellectual Property Protection

CSIT375/975 AI and Cybersecurity

Dr Wei Zong

SCIT University of Wollongong

Disclaimer: The presentation materials come from various sources. For further information, check the references section

Outline

- Introduction
- DNN fingerprinting
  - IPGuard
  - MetaFinger
  - DeepJudge (more details will be covered in the lab)
- DNN watermarking
  - Adversarial frontier stitching
  - Entangled Watermarks
- Defeating IP protection
  - IPRemover

Introduction

- Training **Deep Neural Networks (DNNs)** can be expensive
  - When data is difficult to obtain or labeling them requires significant domain expertise.
    - Examples are medical data, financial data, etc.
  - Training can also be expensive
    - ChatGPT-3 cost around $2 million to $4 million in 2020

- Hence, it is crucial that the **Intellectual Property (IP)** of DNNs trained on valuable data be protected against IP infringement.
    - We have shown that an adversary steals the functional of a black-box model with only USD $30.
    - Patenting model weights is not practical.
      - Can be easily defeated by knowledge distillation.

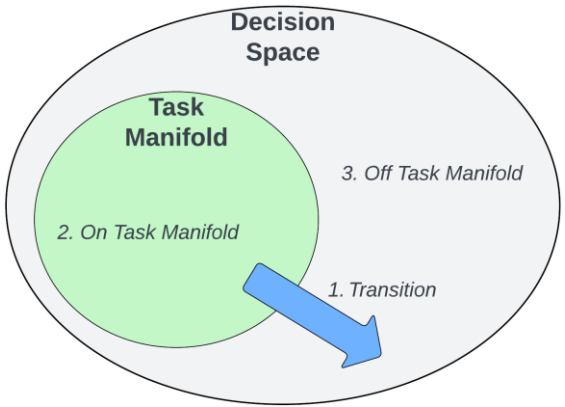
Introduction

- DNN **fingerprinting** and **watermarking** are two lines of work in DNN IP protection
  - DNN **fingerprinting** techniques detect **unique properties** of a model
    - Verifies IP infringement if identical or similar properties exist in a suspect model.
    - The key assumption is that the decision boundaries of independently trained models are intrinsically different from one another.
    - Preserve model performance since model weights are not changed.
  - DNN **watermarking** embeds **watermarks** into a model
    - Verifies IP infringement if identical or similar watermarks are extracted from a suspect model.
    - Inevitably affect model performance since an irrelevant task is learned.
      - Embedding watermarks is different from the original task, e.g., image classification.

- The techniques deployed in fingerprinting and watermarking vary significantly
  - Their underlying mechanisms are different.

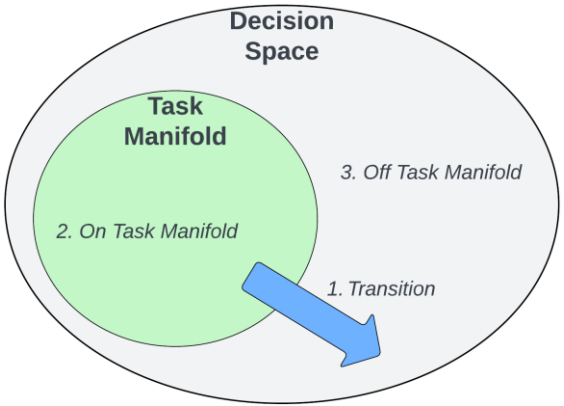
Introduction

- A Unifying View of Fingerprinting and Watermarking
  - They both exploit the unique characteristics of a model's decision space.
    - Decision space can be divided into space that is **on the task manifold** and space that is **off the task manifold** (shown in the figure).
    - Existing techniques are based on one of **three** directions.
  - The first direction
    - Focus on unique transitions from space on the task manifold to space off the task manifold.
    - These unique transitions only exist in the model under protection and cannot be found in other independently trained models.
      - Most existing DNN IP protection techniques are based on this.
    - E.g., adversarial examples (AEs)
      - AEs visually resemble clean data but differ in the feature space.
        - This property is not shared by clean data.
      - Adversarial perturbations as unique transitions that move data points off the task manifold.



Introduction

- A Unifying View of Fingerprinting and Watermarking
  - The second direction
    - Focus on unique characteristics of space on the task manifold.
    - E.g., adversarial frontier stitching (will be discussed soon) extends the task manifold by fine-tuning a model on AEs.
  - The third direction
    - focus on the unique characteristics of space off the task manifold.
      - MetaFinger (will be discussed soon) utilizes meta-learning to generate fingerprints that are off the task manifold.
  - Difference between fingerprinting and watermarking.
    - Fingerprinting finds unique characteristics that already exist.
    - watermarking creates unique characteristics.
      - By modifying the model.

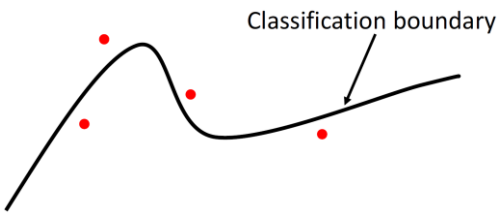


DNN Fingerprinting

- IPGuard
  - Key idea: protect the IP of DNNs using **targeted AEs**.
    - Targeted AEs generated by a single model hardly transfer to new models.
  - Protect a model via generating a set of targeted AEs as **fingerprints**
    - These AEs lie close to the model's decision boundaries.
      - A data point is on the target classifier's classification boundary if the target classifier cannot decide the label of the data point, i.e., at least two labels have the largest probability (or logit) for the data point.
    - Solve an optimization problem to find these AEs:

$$\min_{x} ReLU(Z_i(x) - Z_j(x) + k) + ReLU(\max_{t \neq i,j} Z_t(x) - Z_i(x)),$$

      - x is initially predicted as i.
      - j is the target label.
      - $Z_i(x)$ is the i^{th} logit.
      - Intuition: slightly move a data point across the decision boundary.



Cao, X., Jia, J. and Gong, N.Z., 2021, May. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In Proceedings of the 2021 ACM asia conference on computer and communications security (pp. 14-25).

DNN Fingerprinting

- IPGuard
  - When investigating a suspect model
    - IPGuard inputs this set of previously generated AEs into the suspect model
      - Back-box access is enough.
    - Evaluates the Matching Rate (MR) between the suspect model and the victim model.
      - MR is the rate at which both models output the same labels.
    - If the resulting MR is above a certain threshold, the suspect model is considered to be a copy of the victim model.
  - IPGuard is efficient to run
    - It is a practical technique for protecting large models since the generation of targeted AEs is efficient.

Cao, X., Jia, J. and Gong, N.Z., 2021, May. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In Proceedings of the 2021 ACM asia conference on computer and communications security (pp. 14-25).

DNN Fingerprinting

- IPGuard results
  - Matching rates of the compared methods for positive and negative suspect classifiers on **CIFAR-10**.
    - IGSM is a simple extension of FGSM by running multiple iterations.
    - CW-L_2: l_2 norm constrained CW.
    - Weight pruning (WP)
      - Prune p fraction of weights that have the smallest absolute values and then retrain the prune classifier to enhance the classification accuracy.
      - Increase p from 0.1 with a step size 0.1 until the accuracy loss is larger than 3%
    - Filter pruning (FP).
      - prune c fraction of filters in each layer that have the smallest absolute values and then retrain the pruned classifier to enhance the classification accuracy.
      - increase c from 1/16 until the accuracy loss is larger than 3%.

Suspect classifier				FGSM	IGSM	CW-L_2	IPGuard
Positive suspect classifiers		FTLL		0.90	0.99	1.00	1.00
		FTAL		0.92	0.90	1.00	1.00
		RTLL		0.87	0.99	1.00	1.00
		RTAL		0.90	0.66	1.00	1.00
		WP	p=0.1	0.86	0.91	1.00	1.00
			p=0.2	0.86	0.91	1.00	1.00
			p=0.3	0.89	0.88	1.00	1.00
			p=0.4	0.90	0.70	1.00	1.00
		FP	c=1/16	0.78	0.67	1.00	1.00
			c=2/16	0.80	0.41	1.00	1.00
			c=3/16	0.87	0.37	1.00	1.00
			c=4/16	0.82	0.18	0.99	0.99
			c=5/16	0.79	0.17	0.97	0.94
			c=6/16	0.74	0.14	0.89	0.85
			c=7/16	0.77	0.09	0.70	0.71
Negative suspect classifiers	Same architecture DNNs	ResNet20		[0.06,0.66]	[0.00,0.01]	[0.00,0.07]	[0.00,0.09]
	Different architecture DNNs	LeNet-5		[0.08,0.49]	[0.00,0.03]	[0.00,0.03]	[0.00,0.03]
		VGG16		[0.08,0.78]	[0.00,0.01]	[0.00,0.00]	[0.00,0.01]
	Random forests	RF		[0.02,0.09]	[0.00,0.01]	[0.00,0.00]	[0.00,0.00]

Cao, X., Jia, J. and Gong, N.Z., 2021, May. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In Proceedings of the 2021 ACM asia conference on computer and communications security (pp. 14-25).

DNN Fingerprinting

- IPGuard results
  - Matching rates of the compared methods for positive and negative suspect classifiers on **CIFAR-100**.

Suspect classifier				FGSM	IGSM	CW-L_2	IPGuard
Positive suspect classifiers		FTLL		0.99	0.98	1.00	1.00
		FTAL		0.86	0.92	1.00	1.00
		RTLL		0.92	0.81	1.00	1.00
		RTAL		0.76	0.49	1.00	1.00
		WP	p=0.1	0.76	0.88	1.00	1.00
			p=0.2	0.72	0.88	1.00	1.00
			p=0.3	0.70	0.83	1.00	1.00
			p=0.4	0.54	0.78	1.00	1.00
			p=0.5	0.56	0.66	1.00	1.00
		FP	c=1/16	0.78	0.70	1.00	1.00
			c=2/16	0.66	0.53	1.00	1.00
			c=3/16	0.59	0.42	1.00	1.00
			c=4/16	0.48	0.28	1.00	1.00
			c=5/16	0.55	0.20	0.99	0.99
Negative suspect classifiers	Same architecture DNNs	WRN-22-4		[0.02,0.36]	[0.00,0.04]	[0.00,0.01]	[0.00,0.02]
	Different architecture DNNs	LeNet-5		[0.00,0.07]	[0.00,0.01]	[0.00,0.00]	[0.00,0.01]
		VGG16		[0.00,0.11]	[0.00,0.01]	[0.00,0.00]	[0.00,0.01]
	Random forests	RF		[0.00,0.04]	[0.00,0.02]	[0.00,0.00]	[0.00,0.00]

Cao, X., Jia, J. and Gong, N.Z., 2021, May. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In Proceedings of the 2021 ACM asia conference on computer and communications security (pp. 14-25).

DNN Fingerprinting

- IPGuard results
  - Matching rates of the compared methods for positive and negative suspect classifiers on **ImageNet**.

Suspect classifier				FGSM	IGSM	CW-L_2	IPGuard
Positive suspect classifiers		FTLL		1.00	0.95	1.00	1.00
		FTAL		0.13	0.39	1.00	1.00
		RTLL		0.75	0.32	1.00	1.00
		RTAL		0.43	0.10	0.99	0.99
		WP	p=0.1	0.96	0.83	1.00	1.00
			p=0.2	0.65	0.81	1.00	1.00
			p=0.3	0.58	0.77	1.00	1.00
		FP	c=1/16	0.82	0.02	1.00	0.99
			c=2/16	0.01	0.02	1.00	0.96
			c=3/16	0.00	0.01	0.99	0.90
Negative suspect classifiers	Different architecture DNNs	VGG16		0.32	0.00	0.00	0.01
		ResNet152		0.02	0.00	0.00	0.00
		ResNet152V2		0.00	0.00	0.00	0.00
		InceptionV3		0.00	0.00	0.00	0.00
		InceptionResNetV2		0.00	0.00	0.01	0.00
		Xception		0.00	0.00	0.00	0.00
		MobileNet(α=1.0)		0.00	0.00	0.01	0.00
		MobileNetV2(α=1.4)		0.00	0.00	0.01	0.00
		DenseNet201		0.00	0.00	0.04	0.00
		NASNetLarge		0.00	0.00	0.00	0.00

Cao, X., Jia, J. and Gong, N.Z., 2021, May. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In Proceedings of the 2021 ACM asia conference on computer and communications security (pp. 14-25).

DNN Fingerprinting

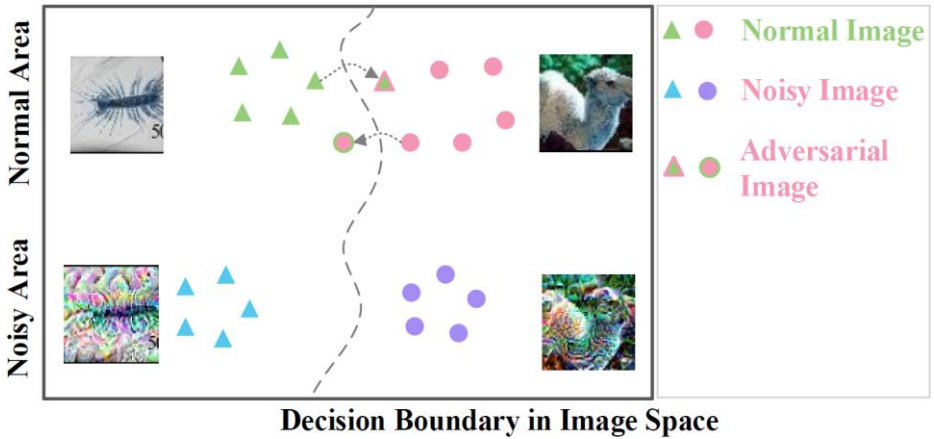
- IPGuard efficiency
  - The Table shows the running time in seconds of generating the 100 fingerprints
    - IPGuard is slower than FGSM and IGSM.
    - However, IPGuard is orders of magnitude faster than CW-L2.
      - They both can find data points near a target classifier's classification boundary.
      - But IPGuard does not constrain noise added to the data points to be more efficient.
        - It does not matter if fingerprints are noisy.
        - CW-L2 aims to find adversarial examples with small noise.

	CIFAR-10	CIFAR-100	ImageNet
Random	<1	<1	<1
FGSM	4.9	4.6	11.6
IGSM	11.2	15.6	47.9
CW-L_2	20,006.3	30,644.6	121,955.7
IPGuard	37.8	249.9	7,634.3

Cao, X., Jia, J. and Gong, N.Z., 2021, May. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In Proceedings of the 2021 ACM asia conference on computer and communications security (pp. 14-25).

DNN Fingerprinting

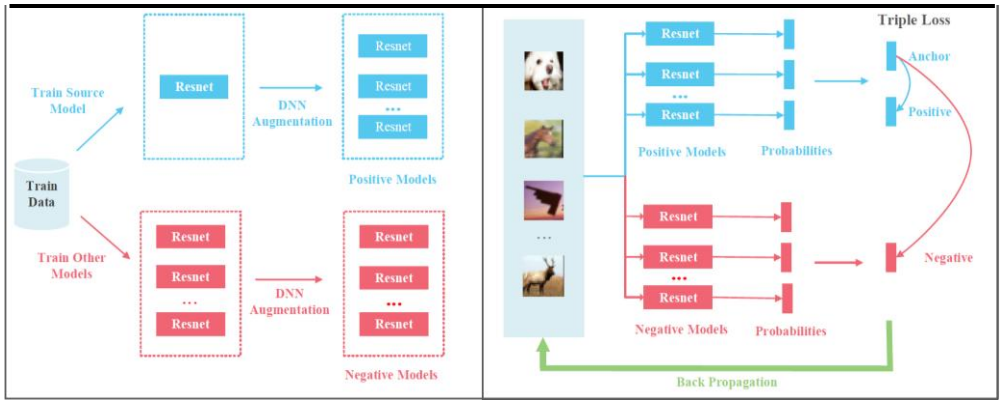
- MetaFinger
  - Key idea: independently trained models behave differently on **noisy input**. MetaFinger identifies a specific model by generating a set of noisy inputs that can ***only*** be correctly classified by this model.
    - This noisy set is referred to as the query set.
    - Unlike fingerprint that rely on AEs (e.g., IPGuard), the noisy input does not aim to "fool" the model.
  - MetaFinger is categorized as an off the task manifold technique because the generated noisy input is not on the task manifold.



Yang, K., Wang, R. and Wang, L., 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In IJCAI (pp. 776-782).

DNN Fingerprinting

- MetaFinger
  - Overview:
    - Stage 1: Apply augmentation to the source model and surrogate models.
      - Surrogate models are trained independently of the source model.
      - Establish a pool of positive models and a pool of negative models.
      - Add Gaussian noise on the weights layer by layer then fine-tunes the model to achieve high accuracy.
    - Stage 2: generate some images and ensure that they can **only** be recognized by models stem from the source model (i.e., the pool of positive models).
      - These images are used as fingerprints, also called **query set**, to detect IP infringement.



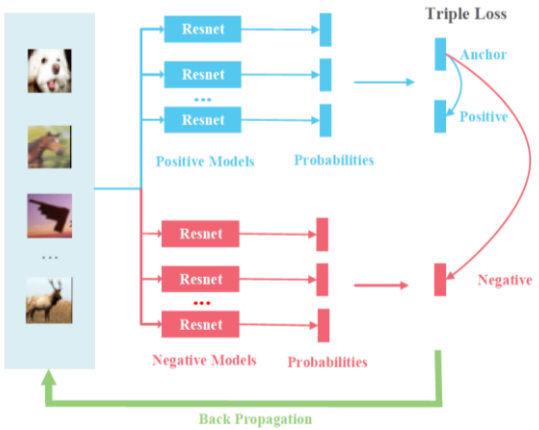
Yang, K., Wang, R. and Wang, L., 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In IJCAI (pp. 776-782).

DNN Fingerprinting

- MetaFinger
  - Query set as fingerprints
    - IP infringement is detected if a suspect model can correctly classify the query set.
      - This implies that the decision boundaries of this suspect model resemble the decision boundaries of the model under protection.
    - Randomly choose some training data as the starting point, denoted as X.
    - Optimize the **triple loss**: reduce **intra-class** distance and enlarge **inter-class** distance.
      - Require positive models predict the same label that is different from the label predicted by negative models.
      - Negative models do not necessarily predict the same label.

$$Loss = \frac{1}{k}\sum_{i=1}^{k} KL(p_{anchor}(X), p_i(X)) - \lambda\frac{1}{k}\sum_{i=1}^{k} KL(p_{anchor}(X), n_i(X))$$

- $p_i, \mathrm{n_i}$ represent positive and negative models, respectively.
- p_{anchor} is a positive model selected from the positive pool.
- KL denotes Kullback-Leibler divergence.



Yang, K., Wang, R. and Wang, L., 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In IJCAI (pp. 776-782).

DNN Fingerprinting

- MetaFinger
  - Algorithm of generating the **query set**.
    - Take the model from the positive models in turn as the anchor model p_{anchor}.
    - Calculate the **triple loss**

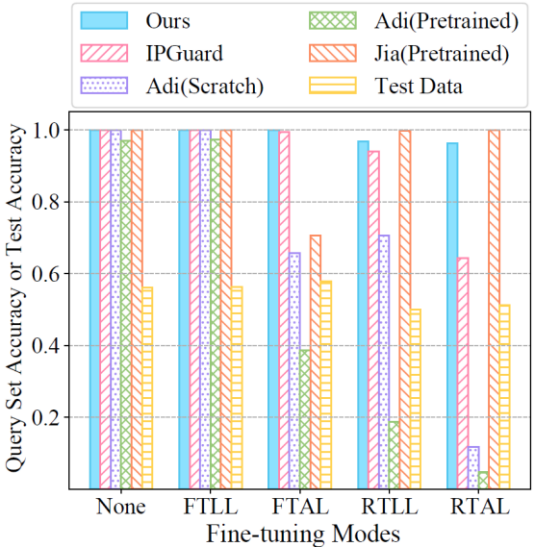
$$Loss = \frac{1}{k}\sum_{i=1}^{k} KL(p_{anchor}(X), p_i(X)) - \lambda\frac{1}{k}\sum_{i=1}^{k} KL(p_{anchor}(X), n_i(X))$$

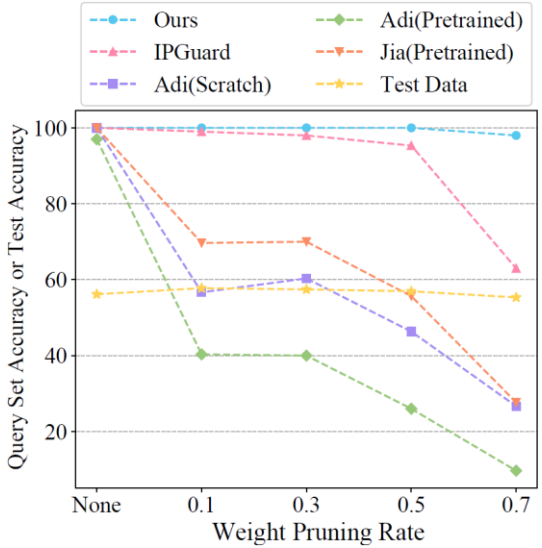
      - Use gradients to optimize X.
    - Labels predicted by most positive models are used as ground-truth labels Y.
    - Move eligible images from X to the **query set**
      - If all positive models predict them as y, and
      - No negative models predict it as y.

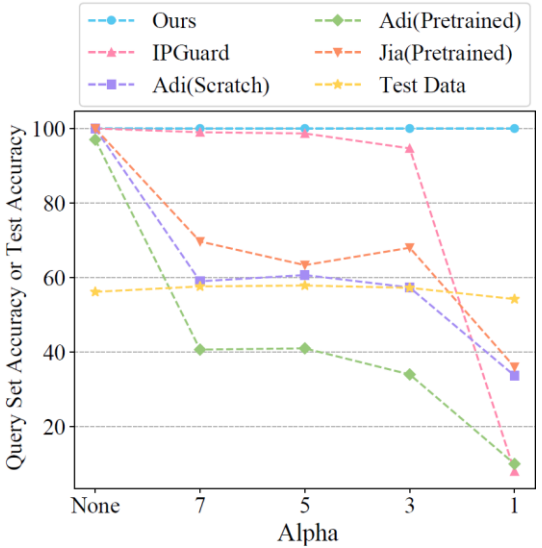
Yang, K., Wang, R. and Wang, L., 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In IJCAI (pp. 776-782).

DNN Fingerprinting

- MetaFinger
  - Results on Tiny-ImageNet (100,000 64×64 colored images of 200 classes).
    - Fine-tuning
      - Fine-Tune Last Layer (FTLL); Fine-Tune All Layers (FTAL).
      - Re-Train Last Layer (RTLL); Re-Train All Layers (RTAL) (randomly initialize the last layer and fine-tune all layers).
    - Weight pruning: prune parameters and then fine-tune the model.
    - Weight noising: add noise to parameters and then fine-tune the model.



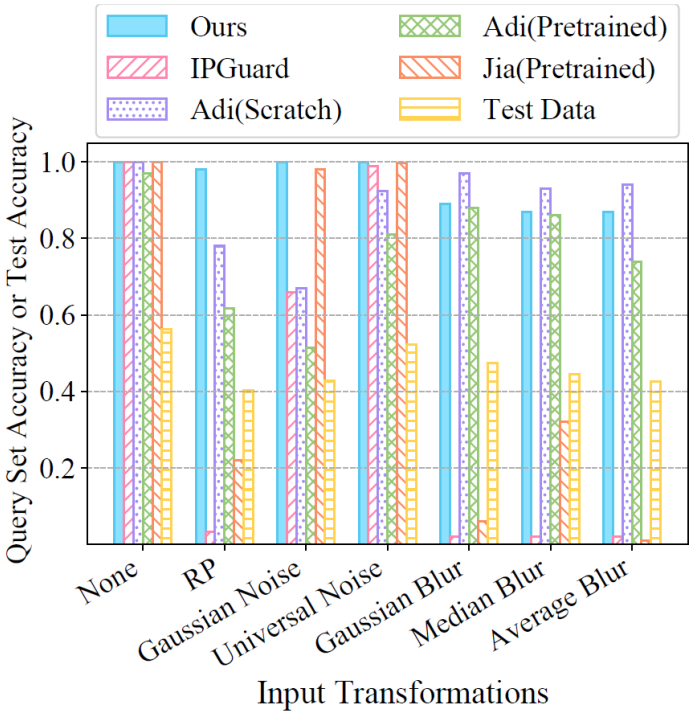




Yang, K., Wang, R. and Wang, L., 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In IJCAI (pp. 776-782).

DNN Fingerprinting

- MetaFinger
  - Robustness against Input Modification
    - Input modification preprocesses the inputs with image transformations.
    - The accuracy of all methods has degraded to some extent under input transformations.
    - MetaFinger still achieves over 80% accuracy.
    - Among all the methods, IPGuard performs the worst
      - Because IPGuard fingerprints are constructed with adversarial perturbations which do not consider robustness against image transformations.
      - May be solved if explicitly optimize against these transformations.



Cao, X., Jia, J. and Gong, N.Z., 2021, May. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In Proceedings of the 2021 ACM asia conference on computer and communications security (pp. 14-25).

DNN Fingerprinting

- MetaFinger
  - Limitation?

Yang, K., Wang, R. and Wang, L., 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In IJCAI (pp. 776-782).

DNN Fingerprinting

- MetaFinger
  - Limitation
    - Need to train negative models.
      - Impractical to protect large models.
      - ChatGPT-3 cost around $2 million to $4 million in 2020

Yang, K., Wang, R. and Wang, L., 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In IJCAI (pp. 776-782).

DNN Fingerprinting

- DeepJudge
  - IP infringement detection is based on **untargeted adversarial examples (AEs)**.
  - 3 Steps
    - Firstly, selects a batch of clean data as seeds.
    - Secondly, use these seeds to generate untargeted AEs.
    - Finally, detect IP infringement based on the similarity between the behavior of a suspect model and a victim model on these untargeted AEs.
      - IP infringement is detected if
        - A suspect model cannot correctly recognize these untarget AEs, or
        - The distribution of the output logits is similar to the victim model.

Yang, K., Wang, R. and Wang, L., 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In IJCAI (pp. 776-782).

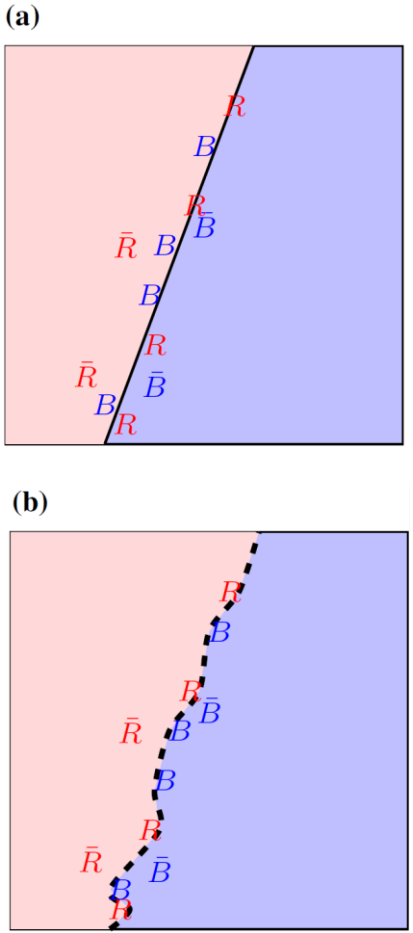
DNN Fingerprinting

- DeepJudge
  - Metrics
    - The two measurements used are the **Robustness Distance (RobD)** and the **Jensen-Shanon Distance (JSD)**.
  - Given a set of untargeted AEs generated by a victim model
    - RobD measures the percentage that a suspect model can still correctly recognize these untargeted AEs.
    - JSD measures the distance of the output distribution between a suspect model and a victim model for these untargeted AEs.
  - A suspect model is determined to be an illegal copy if its RobD and JSD are significantly smaller than independently trained models.
    - Limitation: require to train extra models, so impractical for large DNNs.
  - **More details about DeepJudge are presented in the lab.**

Yang, K., Wang, R. and Wang, L., 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In IJCAI (pp. 776-782).

DNN Watermarking

- Adversarial frontier stitching (FS)
  - An early **watermarking** technique for combating IP infringement.
  - FS is based on two assumptions.
    - The first assumption is that **untargeted** AEs are transferable between DNN models.
    - The second assumption is that stolen models will inherit robustness against AEs from a victim model.
  - FS focuses on unique characteristics of space on the task manifold
    - It extends the task manifold by fine-tuning a model on AEs.
  - Illustration of the key idea for a binary classifier:
    - (a) First compute "true adversaries'' (R and B) and "false'' ones ($\bar{R}$ and $\bar{B}$) for both classes from training examples.
      - They all lie close the decision frontier.
    - (b) Then fine-tune the classifier such that these inputs are now all well classified
      - The 8 true adversaries are now correctly classified.
      - The 4 false ones remain so.
    - This results in a watermarked model (very similar to original one) and a key size of 12 here.
    - This process resembles "stitching'' around data points



Le Merrer, E., Perez, P. and Trédan, G., 2020. Adversarial frontier stitching for remote neural network watermarking. Neural Computing and Applications, 32(13), pp.9233-9244.

DNN Watermarking

- Adversarial frontier stitching (FS)
  - Given a set of correctly classified input, FS uses the **Fast Gradient Sign Method (FGSM)** to generate **untargeted** AEs.
    - Both successfully and unsuccessfully generated AEs will form the set of watermarks.
  - FS can be seen as a simplified version of adversarial training
    - Adversarial training requires robustness against all AEs.
    - FS only requires the model under protection to correctly recognize only a fixed set of AEs.
  - FS detects IP infringement via a probabilistic approach
    - Assume that an independently trained model can achieve 50% accuracy on the watermarks.
      - IP infringement is detected if a model makes only a small number of errors.
      - Thresholds can be theoretically calculated using statistics.
    - FS demonstrated its effectiveness using the simple dataset MNIST and models
      - This assumption is invalid for more complex datasets and models.
      - Independently trained models on CIFAR10 and GTSRB achieved 87% and 77% accuracy on watermarks.
        - **Untargeted** AEs are less transferrable for complex datasets and models.

Le Merrer, E., Perez, P. and Trédan, G., 2020. Adversarial frontier stitching for remote neural network watermarking. Neural Computing and Applications, 32(13), pp.9233-9244.

DNN Watermarking

- Adversarial frontier stitching (FS)
  - Results
    - 3 models are trained on MNIST: CNN, IRNN (using fully connected recurrent layer), and MLP.
      - 50 true adversaries and 50 false adversaries.
    - Different pruning rates are tested to check if watermarks get erased, while accuracy remains acceptable.
      - Results in italics are to be ignored (impractical attacks)

	Pruning rate	*K* elts removed	SD	Extraction rate	Acc. after
CNN	0.25	0.053/100	0.229	1.000	0.983
–	0.50	0.263/100	0.562	1.000	0.984
–	0.75	3.579/100	2.479	1.000	0.983
–	*0.85*	*34.000/100*	*9.298*	*0.789*	*0.936*
IRNN	0.25	14.038/100	3.873	1.000	0.991
-	0.50	59.920/100	6.782	0.000	0.987
–	*0.75*	*84.400/100*	*4.093*	*0.000*	*0.148*
MLP	0.25	0.360/100	0.700	1.000	0.951
–	*0.50*	*0.704/100*	*0.724*	*1.000*	*0.947*
–	*0.75*	*9.615/100*	*4.392*	*1.000*	*0.915*
–	*0.80*	*24.438/100*	*5.501*	*1.000*	*0.877*

Le Merrer, E., Perez, P. and Trédan, G., 2020. Adversarial frontier stitching for remote neural network watermarking. Neural Computing and Applications, 32(13), pp.9233-9244.

DNN Watermarking

- Adversarial frontier stitching (FS)
  - Limitation?

Le Merrer, E., Perez, P. and Trédan, G., 2020. Adversarial frontier stitching for remote neural network watermarking. Neural Computing and Applications, 32(13), pp.9233-9244.

DNN Watermarking

- Adversarial frontier stitching (FS)
  - Limitation
    - **Untargeted** AEs are less transferrable for complex datasets and models.
    - Models trained by **adversarial training** will be detected as IP infringement.

Le Merrer, E., Perez, P. and Trédan, G., 2020. Adversarial frontier stitching for remote neural network watermarking. Neural Computing and Applications, 32(13), pp.9233-9244.

DNN Watermarking

- Entangled Watermarks
  - DNN watermarking techniques are vulnerable to model stealing attacks
    - When the embedded watermarks are uncorrelated with the intended task.
    - This is because a stolen model will not learn the watermarks embedded in a victim model.
  - Entangled Watermarks aim to preserve robustness against model stealing attacks.
  - Key idea:
    - Making the model under protection classify watermarked data as a different target label.
    - Also require the entanglement of internal representations of watermarked data and benign data.
      - The knowledge learned for benign data also includes information about the watermarks.
      - In this way, the embedded watermarks will be learned by a stolen model.

Jia, H., Choquette-Choo, C.A., Chandrasekaran, V. and Papernot, N., 2021. Entangled watermarks as a defense against model extraction. In 30th USENIX security symposium (USENIX Security 21) (pp. 1937-1954).

DNN Watermarking

- Entangled Watermarks
  - The entanglement is achieved via **maximizing** the **Soft Nearest Neighbor Loss** (SNNL) for watermarked and benign input.

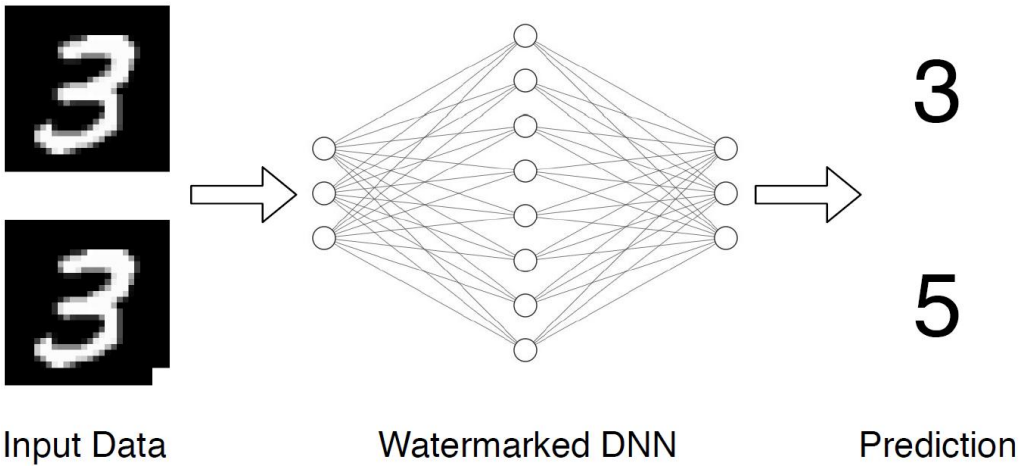
$$SNNL(X,Y,T) = -\frac{1}{N}\sum_{i\in 1..N} log\left(\frac{\sum_{\substack{j\in 1..N \\ j\neq i \\ y_i=y_j}} e^{-\frac{||x_i-x_j||^2}{T}}}{\sum_{\substack{k\in 1..N \\ k\neq i}} e^{-\frac{||x_i-x_k||^2}{T}}}\right) \begin{matrix}(a)\\ \\ (b)\end{matrix}$$

  - The SNNL measures distances between points from all the groups (the classes) relative to the average distance for points within the same group.
    - (a) the average negative distance separating a point x_i from other points in the same group y_i.
    - (b) the average negative distance separating two points.
  - The data forming separate clusters (one for each class) when the SNNL is **minimized**
    - Forming overlapping clusters when the SNNL is **maximized**.
  - SNNL is calculated based on activation values from multiple DNN layers.

Jia, H., Choquette-Choo, C.A., Chandrasekaran, V. and Papernot, N., 2021. Entangled watermarks as a defense against model extraction. In 30th USENIX security symposium (USENIX Security 21) (pp. 1937-1954).

DNN Watermarking

- Entangled Watermarks
  - Embedding watermarks
    - An example
      - a small white square is designed as a special trigger. If this square is added to the corner of a digit-3, the input would be predicted as a digit-5 by the DNN.
      - a normal model would classify it as a digit-3 mostly.
    - Conceptually like BadNets improved by **entangled representations**.



Jia, H., Choquette-Choo, C.A., Chandrasekaran, V. and Papernot, N., 2021. Entangled watermarks as a defense against model extraction. In 30th USENIX security symposium (USENIX Security 21) (pp. 1937-1954).

DNN Watermarking

- Entangled Watermarks
  - Embedding watermarks
    - Loss function

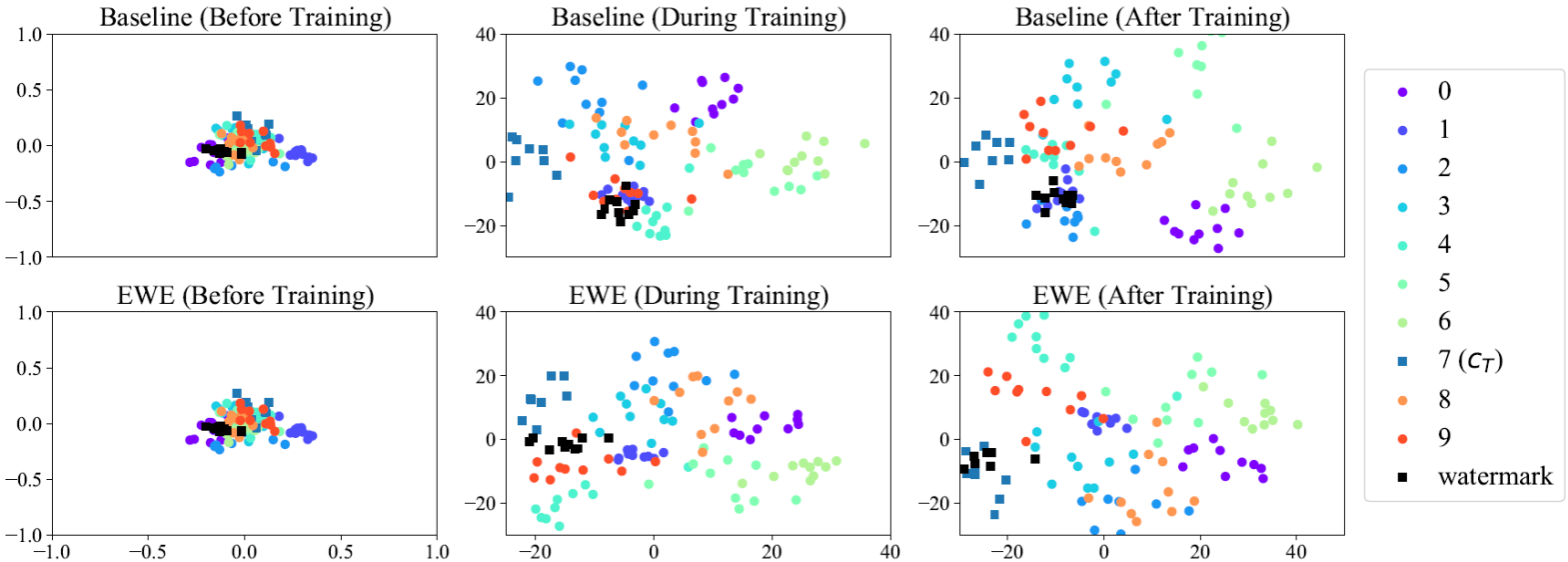
$$\mathcal{L} = \mathcal{L}_{CE}(X, Y) - \kappa \cdot \sum_{l=1}^{L} SNNL([X_w^{(l)}, X_{cT}^{(l)}], Y', T^{(l)}))$$

      - L is the total number of layers in the DNN.
      - X_{cT} denotes normal data from the target class cT.
      - X_w denotes data stamped with triggers.
        - **Original** labels of X_w are different from cT.
      - Y' denotes **original** labels for $[X_w, \mathrm{X_{cT}}]$.
      - T denotes the temperature in SNNL.
    - Training strategy
      - Sample normal batches of data X.
        - Watermarked data are labeled as the target (same as BadNets).
        - Set k = 0 on data X to minimize only the task (cross-entropy) loss.
      - Sample a batch of X_w concatenated with X_{cT}: $[X_w, X_{cT}]$
        - Set k > 0 to optimize the total loss on interleaved data $[X_w, \mathrm{X_{cT}}]$ that includes watermarks.

Jia, H., Choquette-Choo, C.A., Chandrasekaran, V. and Papernot, N., 2021. Entangled watermarks as a defense against model extraction. In 30th USENIX security symposium (USENIX Security 21) (pp. 1937-1954).

DNN Watermarking

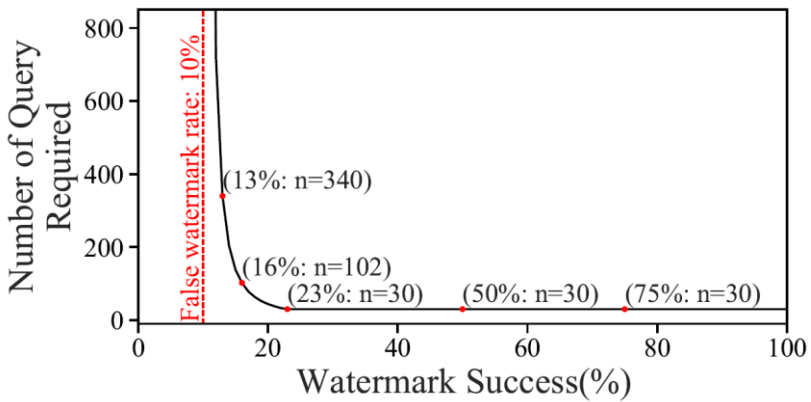
- Entangled Watermarks
  - Visualizing entangled representation (the target class cT = 7).
    - MNIST is the dataset.
    - Use PCA to project the representations of data in each model's penultimate layer.
    - Top row: a baseline model trained with the cross-entropy loss only, i.e., BadNets.
    - Bottom row: a model trained with entangled watermarks.



Jia, H., Choquette-Choo, C.A., Chandrasekaran, V. and Papernot, N., 2021. Entangled watermarks as a defense against model extraction. In 30th USENIX security symposium (USENIX Security 21) (pp. 1937-1954).

DNN Watermarking

- Entangled Watermarks
  - Ownership Verification
    - Given a suspect model, IP infringement is detected if the model recognizes watermarked input as the target label with abnormally high accuracy.
      - Set watermark recognition accuracy (i.e., false positive rate) to random chance as a conservative upper bound for **benign models**.
      - It should be lower than random chance since the watermarked data are semantically different from cT.
    - A defender can exploit **T test** to claim ownership of a stolen model.
      - The Central Limit Theoreom (CLT) requires the number of queries ≥ 30.



Jia, H., Choquette-Choo, C.A., Chandrasekaran, V. and Papernot, N., 2021. Entangled watermarks as a defense against model extraction. In 30th USENIX security symposium (USENIX Security 21) (pp. 1937-1954).

DNN Watermarking

- Entangled Watermarks
  - Results
    - The baseline minimizes cross-entropy of watermarks with the target class (BadNets).
    - A strong adversary has knowledge of the training data used to train the victim model but not its labels.
      - To remove watermarks, this adversary retrains using only the cross-entropy loss evaluated only on training data labeled by the victim model.
    - The watermark success rate is the proportion of X_w correctly identified as cT .
      - The default watermark recognition accuracy, i.e., random chance, is subtracted.

Dataset	Method	Victim Model		Extracted Model	
		Validation Accuracy	Watermark Success	Validation Accuracy	Watermark Success
MNIST	*Baseline*	99.03(±0.04)%	99.98(±0.03)%	98.79(±0.12)%	0.31(±0.23)%
	EWE	98.91(±0.13)%	99.9(±0.11)%	98.76(±0.12)%	65.68(±10.89)%
Fashion MNIST	*Baseline*	90.48(±0.32)%	98.76(±1.07)%	89.8(±0.38)%	8.96(±8.28)%
	EWE	90.31(±0.31)%	87.83(±5.86)%	89.82(±0.45)%	58.1(±12.95)%
Speech Command	*Baseline*	98.11(±0.35)%	98.67(±0.94)%	97.3(±0.43)%	3.55(±1.89)%
	EWE	97.5(±0.44)%	96.49(±2.18)%	96.83(±0.45)%	41.65(±22.39)%
Fashion MNIST	*Baseline*	91.64(±0.36)%	75.6(±15.09)%	91.05(±0.44)%	5.68(±11.78)%
(ResNet)	*EWE*	88.33(±1.97)%	94.24(±5.5)%	88.27(±1.53)%	24.63(±17.99)%
CIFAR10	*Baseline*	85.82(±1.04)%	19.9(±15.48)%	81.62(±1.74)%	7.83(±14.23)%
	EWE	85.41(±1.01)%	25.74(±8.67)%	81.78(±1.31)%	18.74(±12.3)%
CIFAR100	*Baseline*	54.11(±1.89)%	8.37(±13.44)%	47.42(±2.54)%	8.31(±15.1)%
	EWE	53.85(±1.07)%	67.87(±10.97)%	47.62(±1.41)%	21.55(±9.76)%

Jia, H., Choquette-Choo, C.A., Chandrasekaran, V. and Papernot, N., 2021. Entangled watermarks as a defense against model extraction. In 30th USENIX security symposium (USENIX Security 21) (pp. 1937-1954).

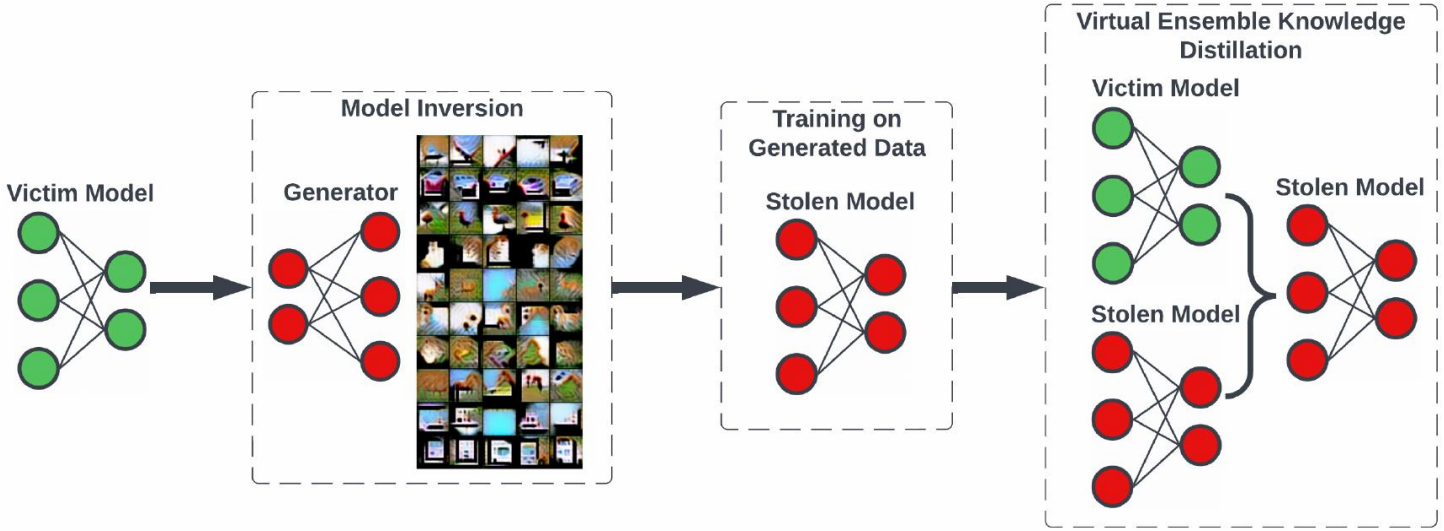
Removing DNN IP

- IPRemover
  - Evade detection by both state-of-the-art DNN fingerprinting and watermarking.
    - Consider challenging **data-free** scenario
      - An adversary has no access to any existing data.
      - Performance can be improved with access to labeled data.
    - A victim model can be accessed in a white-box manner
      - E.g., when an adversary has a local copy of the victim model.
  - Key idea: use data-free Knowledge Distillation (KD)
    - Generate training data from a victim model and use them to train a stolen model.
  - Fingerprinting and watermarking both exploit the unique characteristics of a model's decision space.
    - Fingerprinting finds unique characteristics that already exist.
    - Watermarking creates unique characteristics by modifying the model.
    - To universally defeat both watermarking and fingerprinting, a successful attack must significantly change the model's decision space while maintaining satisfactory model performance.

Zong, W., Chow, Y.W., Susilo, W., Baek, J., Kim, J. and Camtepe, S., 2024, March. IPRemover: A Generative Model Inversion Attack against Deep Neural Network Fingerprinting and Watermarking. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 7, pp. 7837-7845).

Removing DNN IP

- IPRemover
  - Consist of 3 stages.
    - Firstly, training data is inverted from a victim model.
    - Secondly, a stolen model is trained from scratch on the generated data.
    - Finally, a specially designed variant of KD, called Virtual Ensemble Knowledge Distillation (VEKD), is applied to distill knowledge from the victim model while evading IP infringement detection.



Zong, W., Chow, Y.W., Susilo, W., Baek, J., Kim, J. and Camtepe, S., 2024, March. IPRemover: A Generative Model Inversion Attack against Deep Neural Network Fingerprinting and Watermarking. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 7, pp. 7837-7845).

Removing DNN IP

- IPRemover
  - Stage 1: inverting training data
    - Training a generative model to invert training data

$$\min_{\theta_g} \mathcal{H}(v \circ g(z;\theta_g), y) + \sum_{l}^{L_v} \alpha_l \ell_{bn}^{l}(v \circ g(z;\theta_g))$$

    - Where θ_g represents parameters of a generative model g to optimize
      - g can be a simple DNN with transposed convolution layers.
    - v represents the victim model.
    - H denotes the cross-entropy loss.
    - z represents a random variable sampled from standard distribution N(0, 1).
    - y are randomly selected labels.
    - L_v denotes the number of batch normalization layers in the victim model.
      - ℓ_{bn} is the widely used batch normalization regularization
      - ℓ_{bn} aims to constrain the features of each batch normalization layer to be consistent with the running-mean and running-variance values so that generated data will visually resemble the original training data.
      - ℓ_{bn}^{l} denotes the loss calculated for the l^{th} layer with α_l for weighting.

Zong, W., Chow, Y.W., Susilo, W., Baek, J., Kim, J. and Camtepe, S., 2024, March. IPRemover: A Generative Model Inversion Attack against Deep Neural Network Fingerprinting and Watermarking. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 7, pp. 7837-7845).

Removing DNN IP

- IPRemover
  - Stage 2: train a stolen model on the generated data from scratch.
  - Stage 3: VEKD is applied to transfer knowledge from the victim model.
    - VEKD includes the stolen model itself as an additional teacher to create an ensemble.
      - Reduce the resemblance between the stolen model and the victim model during knowledge transfer.
    - The loss function for VEKD:

$$\ell = \mathcal{H}(p, y) + \beta KD_{\tau}(p, \epsilon v + (1-\epsilon)q)$$

      - p denotes probabilities output by the stolen model; y is the label of generated data; v denotes probabilities output by the victim model; KD_{τ} is the KD loss with temperature τ.
      - q is the output probability when the stolen model serves as an additional teacher:

$$q_i = \begin{cases} Q, & \text{if } i = \arg\max(p) \\ (1-Q)/(K-1), & \text{otherwise} \end{cases}$$

        - Q is a predefined value representing high probability; K is the number of classes.
      - ϵ balances v and q.

Zong, W., Chow, Y.W., Susilo, W., Baek, J., Kim, J. and Camtepe, S., 2024, March. IPRemover: A Generative Model Inversion Attack against Deep Neural Network Fingerprinting and Watermarking. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 7, pp. 7837-7845).

Removing DNN IP

- IPRemover
  - Results on MetaFinger (CIFAR10, size of query set = 100)

Scenario	Accuracy (%)	Query (%)	Detected*
GTSRB	85.32 ± 0.11	85.33 ± 0.94	3/3
CIFAR100	89.37 ± 0.13	98.67 ± 1.25	3/3
Data-free	82.98 ± 0.18	44.33 ± 3.77	0/3
1% data	84.59 ± 0.29	39.67 ± 2.49	0/3
5% data	87.40 ± 0.19	33.00 ± 4.32	0/3

  - IP infringement is detected if the accuracy on the query set exceeds a threshold of 62%.
    - 62% is achieved via an independently trained ResNet.
    - The victim model achieved 90.92% accuracy on the test set and 100% accuracy on the query set.
  - Scenarios:
    - “GTSRB”: KD is applied using GTSRB as out-of-distribution (OOD) data to transfer knowledge.
    - “CIFAR100”: KD is applied using CIFAR100 as OOD data to transfer knowledge.
    - “Data-free”: Using VEKD with only generated data.
    - “1% data”: Using VEKD with a mixture of generated data and 1% labeled training data.
    - “5% data”: Using VEKD with a mixture of generated data and 5% labeled training data.

Zong, W., Chow, Y.W., Susilo, W., Baek, J., Kim, J. and Camtepe, S., 2024, March. IPRemover: A Generative Model Inversion Attack against Deep Neural Network Fingerprinting and Watermarking. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 7, pp. 7837-7845).

Removing DNN IP

- IPRemover
  - More results on CIFAR10

Defense	Victim Acc (%)	Stolen Acc (%)	Threshold†	Detect+	Stolen Metric
IPGuard	93.25	83.71 ± 0.10	0.0	↑	0.0 ± 0.0
DeepJudge*	93.25	83.71 ± 0.10	RobD:0.326 JSD:0.217	↓	0.426 +- 0.006 0.282 +- 0.005
Entangled	92.16	83.65 ± 0.26	10% (99%)	↑	$5.22 \pm 1.10\%$
FS	93.30	85.24 ± 0.23	87% (100%)	↑	$84.00 \pm 0.82\%$
CosWM	82.40	75.23 ± 0.34	8.0 (39.83)	↑	1.85 ± 1.36

- $*$: DeepJudge proposed two metrics: “RobD” and “JSD”.
- +: ↑ (↓) means IP infringement is detected if the measured metric is higher (lower) compared to the metric of independently trained models.
- †: for fingerprinting, the detection threshold was set to the worst value calculated based on benign models.
  - For watermarking, the watermark accuracy of the victim model or a provided detectable stolen model is shown in parentheses.

Zong, W., Chow, Y.W., Susilo, W., Baek, J., Kim, J. and Camtepe, S., 2024, March. IPRemover: A Generative Model Inversion Attack against Deep Neural Network Fingerprinting and Watermarking. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 7, pp. 7837-7845).

References

Cao, X., Jia, J. and Gong, N.Z., 2021, May. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In Proceedings of the 2021 ACM asia conference on computer and communications security (pp. 14-25).

Yang, K., Wang, R. and Wang, L., 2022. MetaFinger: Fingerprinting the Deep Neural Networks with Meta-training. In IJCAI (pp. 776-782).

Le Merrer, E., Perez, P. and Trédan, G., 2020. Adversarial frontier stitching for remote neural network watermarking. Neural Computing and Applications, 32(13), pp.9233-9244.

Jia, H., Choquette-Choo, C.A., Chandrasekaran, V. and Papernot, N., 2021. Entangled watermarks as a defense against model extraction. In 30th USENIX security symposium (USENIX Security 21) (pp. 1937-1954).

Zong, W., Chow, Y.W., Susilo, W., Baek, J., Kim, J. and Camtepe, S., 2024, March. IPRemover: A Generative Model Inversion Attack against Deep Neural Network Fingerprinting and Watermarking. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 7, pp. 7837-7845).

Zong, W., Chow, Y.W., Susilo, W., Baek, J., Kim, J. and Camtepe, S., 2024, March. IPRemover: A Generative Model Inversion Attack against Deep Neural Network Fingerprinting and Watermarking. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 7, pp. 7837-7845).