Hashed Watermark as a Filter: A Unified Defense Against Forging and Overwriting Attacks in Neural Network Watermarking

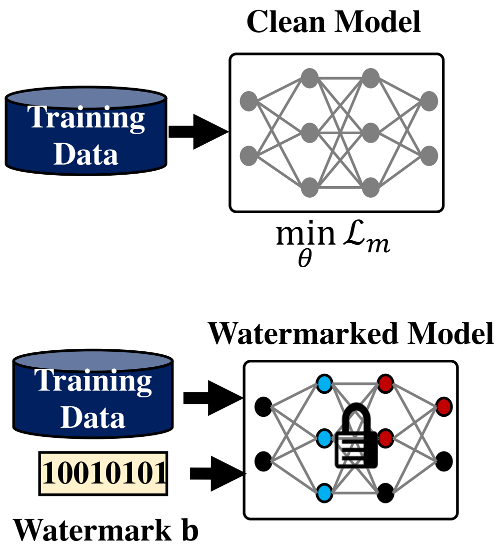
Yuan Yao[1], Jin Song[2], Jian Jin[3]

1 Beijing Teleinfo Technology Company Ltd., China Academy of Information and Communications Technology
2 School of Computer Science, Nanjing University of Posts and Telecommunications
3 Research Institute of Industrial Internet of Things, China Academy of Information and Communications Technology

2026.01

Teleinfo

Background & Problem



$$\min_{\theta} \mathcal{L}_m + \lambda \mathcal{L}_e(\tilde{\mathbf{b}}, \mathbf{b})$$

where $\tilde{\mathbf{b}} = \text{sigmod}(\widehat{\mathbf{w}}\mathbf{K})$ and $\mathbf{K}$ is a secret matrix.

Forging Attacks: The adversary generates $\mathbf{b}_a$ and optimizes $\mathbf{K}_a$ under **frozen model parameters** via $\min_{\mathbf{K}_a} \mathcal{L}_m + \lambda(\tilde{\mathbf{b}}, \mathbf{b}_a)$.

Overwriting Attacks: The adversary attempts to overwrite the original watermark by embedding a counterfeit one via $\min_{\theta} \mathcal{L}_m + \lambda \mathcal{L}_e(\tilde{\mathbf{b}}, \mathbf{b}_a)$.

Fine-tuning Attacks: The adversary aims to fine-tune the model to remove the original watermark.

Pruning Attacks: The adversary attempts to remove the original watermark by parameter pruning.

How to design a watermarking method to resist the above attacks without compromising performance?

Background & Problem

$$\min_{\theta} \mathcal{L}_m + \lambda \mathcal{L}_e(\tilde{\mathbf{b}}, \mathbf{b})$$

where $\tilde{\mathbf{b}} = \text{sigmod}(\hat{\mathbf{w}}\mathbf{K})$ and $\mathbf{K}$ is a secret matrix.

Forging Attacks: The adversary generates $\mathbf{b}_a$ and optimizes $\mathbf{K}_a$ under **frozen model parameters** via $\min_{\mathbf{K}_a} \mathcal{L}_m + \lambda(\tilde{\mathbf{b}}, \mathbf{b}_a)$.

Overwriting Attacks: The adversary attempts to overwrite the original watermark by embedding a counterfeit one via $\min_{\theta} \mathcal{L}_m + \lambda \mathcal{L}_e(\tilde{\mathbf{b}}, \mathbf{b}_a)$.

Hashed Watermark Filter：Resist forging and overwriting attacks

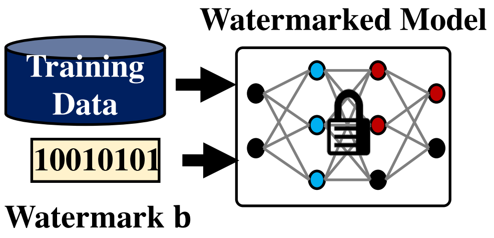
Fine-tuning Attacks: The adversary aims to fine-tune the model to remove the original watermark.

Pruning Attacks: The adversary attempts to remove the original watermark by parameter pruning.

Average Pooling：Resist fine-tuning and pruning attacks

What is Hashed Watermark Filter?

Hashed Watermark Filter



$$\min_{\theta} \mathcal{L}_m + \lambda \mathcal{L}_e(\tilde{\mathbf{b}}, \mathbf{b})$$

where $\tilde{\mathbf{b}} = \text{sigmod}(\hat{\mathbf{w}}\mathbf{K})$ and $\mathbf{K}$ is a secret matrix.

Forging Attacks: The adversary generates $\mathbf{b}_a$ and optimizes $\mathbf{K}_a$ under **frozen model parameters** via $\min_{\mathbf{K}_a} \mathcal{L}_m + \lambda(\tilde{\mathbf{b}}, \mathbf{b}_a)$.

Gradient obfuscation: $\mathbf{b} = \text{HASH}(\mathbf{K})$ or $\mathbf{b} = \text{HASH}(\mathbf{K} \mid\mid C)$

Overwriting Attacks: The adversary attempts to overwrite the original watermark by embedding a counterfeit one via $\min_{\theta} \mathcal{L}_m + \lambda \mathcal{L}_e(\tilde{\mathbf{b}}, \mathbf{b}_a)$.

Embedding isolation: Using $\mathbf{b}$ to select embedding parameters

Hashed Watermark Filter

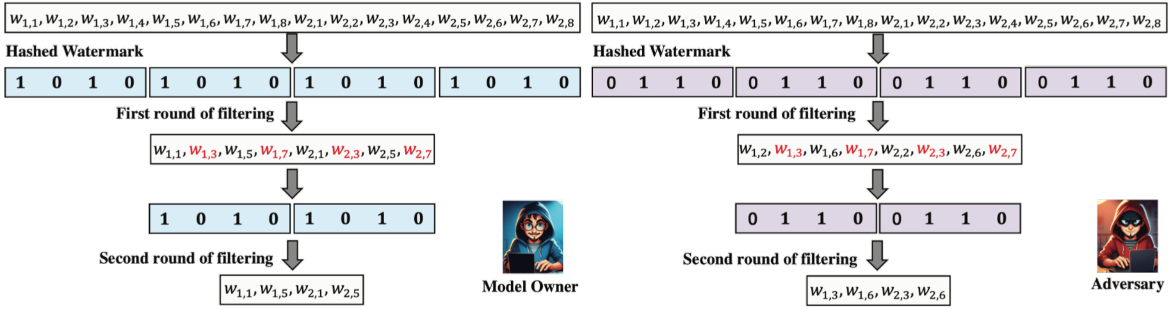


Figure 1: Illustration of the hashed watermark filter. The model owner's hashed watermark is $[1, 0, 1, 0]$, while the adversary's is $[0, 1, 1, 0]$. The watermark is repeated to match the parameter length before each round of filtering. Without filtering, all 16 parameters overlap. After the first round, each watermark retains eight parameters with four overlapping; after the second round, only four parameters remain for each, with no overlap.

NeuralMark

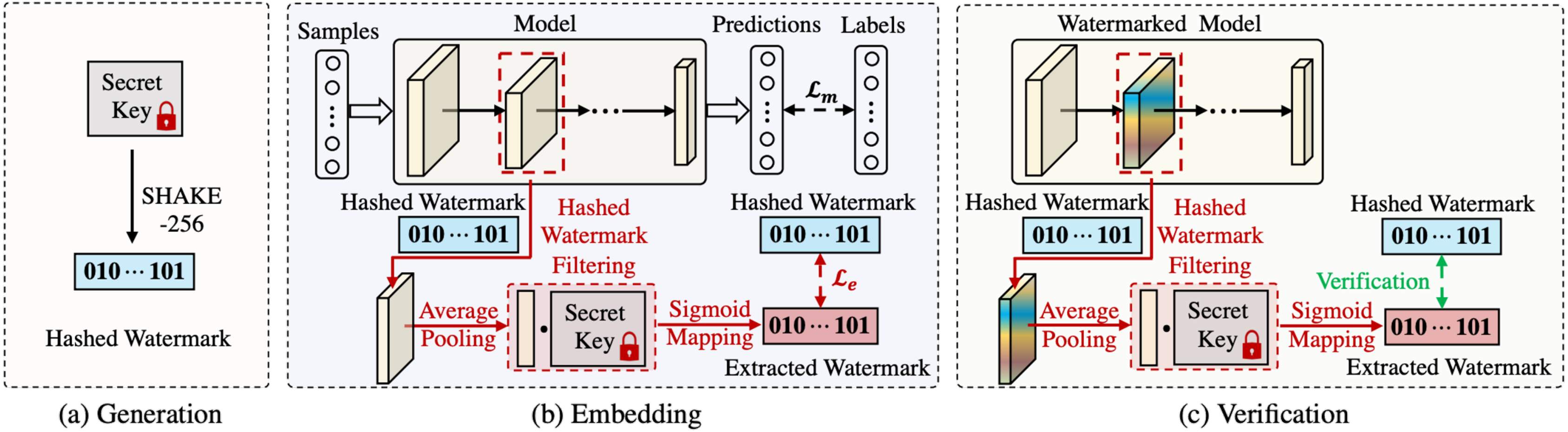


Figure 5: Illustrations of the processes for watermark generation (a), embedding (b), and verification (c).

Generation: $\mathbf{b} = \mathrm{HASH}(\mathbf{K})$ or $\mathbf{b} = \mathrm{HASH}(\mathbf{K} \; || \; C)$

Embedding: $\min_{\theta} \mathcal{L}_m + \lambda \mathcal{L}_e(\tilde{\mathbf{b}}, \mathbf{b})$

Verification: $\rho=\frac{1}{n}\sum_{i=}^{1} \mathbf{1}[b_i = \mathcal{T}(\tilde{b}_i)] \geq \rho^* \wedge \mathrm{HASH}(\mathbf{K})= \mathbf{b}$

Analysis

Necessity of Hashed Watermark Filter

We compare the hashed watermark filter with a private filter. Although this baseline resists overwriting, it remains vulnerable to forging: an adversary can use a 256×256 identity matrix as the key $\mathbf{K}$, forge a watermark $\mathbf{b}$, and choose parameters $\widehat{\mathbf{w}}$ whose signs match $\mathbf{b}$, thereby constructing a private filter satisfying $\mathcal{T}(\text{sigmod}(\widehat{\mathbf{w}}\mathbf{K})) = \mathbf{b}$ and $\mathcal{H}(\mathbf{K}) = \mathbf{b}$, thus bypassing verification.

$\mathbf{w}$ = 0.3 -0.2 0.5 -0.1 $\widehat{\mathbf{w}} = -0.2$ 0.5 $\mathbf{b}$ = 0 1

Security Boundary Analysis

Proposition 1. *Under the assumption that the hash function produces uniformly distributed outputs (Bellare and Rogaway 1993), for a model watermarked by NeuralMark with a watermark tuple* $\{\mathbf{K}, \mathbf{b}\}$, *where* $\mathbf{b} = \mathcal{H}(\mathbf{K})$, *if an adversary attempts to forge a counterfeit watermark tuple* $\{\mathbf{K}', \mathbf{b}'\}$ *such that* $\mathbf{b}' = \mathcal{H}(\mathbf{K}')$ *and* $\mathbf{K}' \neq \mathbf{K}$, *then the probability of achieving a watermark detection rate of at least* ρ *(i.e.,* $\geq \rho$*) is upper-bounded by* $\frac{1}{2^n} \sum_{i=0}^{n-\lceil \rho n \rceil} \binom{n}{i}$.

当 n=256 时，若水印检测率 $\rho \geq 88.29\%$ ，则该结果由伪造导致的概率小于 $1/2^{128}$。

Experiments: Fidelity Evaluation

Dataset	Clean		NeuralMark		VanillaMark		GreedyMark		VoteMark	
	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18
CIFAR-10	91.05	94.76	90.93	94.50	91.01	94.87	90.88	94.69	90.86	94.79
CIFAR-100	68.24	76.23	68.57	76.34	68.43	76.22	68.31	76.14	68.53	76.74
Caltech-101	68.07	68.83	68.38	68.47	68.54	68.99	68.59	69.08	68.88	67.91
Caltech-256	44.27	54.09	44.55	53.71	44.73	53.47	44.64	53.28	44.43	54.71
TinyImageNet	42.42	53.48	42.31	53.22	42.50	53.36	42.94	53.31	42.50	53.47

Table 1: Comparison of classification accuracy (%) across distinct datasets using AlexNet and ResNet-18. Watermark detection rates are omitted as they all reach 100%.

Method	ViT-B/16	Swin-V2-B	Swin-V2-S	VGG-16	VGG-13	ResNet-34	WideResNet-50	GoogLeNet	MobileNet-V3-L
Clean	39.07	52.99	55.88	72.75	72.71	77.06	59.67	60.71	61.11
NeuralMark	39.22	53.57	55.87	72.61	71.49	77.03	58.41	60.02	61.8

Table 2: Comparison of classification accuracy (%) on CIFAR-100 using various architectures. Watermark detection rates are omitted as they all reach 100%.

GPT-2-S	BLEU	NIST	MET	ROUGE-L	CIDEr	GPT-2-M	BLEU	NIST	MET	ROUGE-L	CIDEr
Clean	69.36	8.76	46.06	70.85	2.48	Clean	68.7	8.69	46.38	71.19	2.5
NeuralMark	69.59	8.79	46.01	70.85	2.48	NeuralMark	67.73	8.57	46.07	70.66	2.47

Table 3: Comparison on E2E using GPT-2-S and GPT-2-M. Watermark detection rates are omitted as they all reach 100%.

Experiments: Robustness Evaluation

Dataset	NeuralMark	VanillaMark	GreedyMark	VoteMark
CIFAR-10	48.56	100.00	50.70	100.00
CIFAR-100	49.41	100.00	52.85	100.00

Table 4: Comparison of detection rate (%) of counterfeit watermarks using ResNet-18.

Forging Attacks

Overwriting	λ	NeuralMark	VanillaMark	GreedyMark	VoteMark	η	NeuralMark	VanillaMark	GreedyMark	VoteMark
CIFAR-100 to CIFAR-10	1	93.65 (**100**)	93.30 (**100**)	93.45 (**48.82**)	93.63 (**100**)	0.001	93.65 (**100**)	93.30 (**100**)	93.45 (**48.82**)	93.63 (**100**)
	10	93.44 (**100**)	93.58 (**100**)	93.29 (**51.17**)	93.13 (**100**)	0.005	91.76 (**99.60**)	92.17 (**73.04**)	92.13 (**50.00**)	92.45 (**78.90**)
	50	93.46 (**100**)	93.50 (**100**)	93.07 (**55.07**)	93.39 (**100**)	0.01	91.58 (**92.18**)	91.79 (**62.10**)	91.53 (**49.60**)	91.76 (**60.15**)
	100	93.53 (**100**)	92.95 (**94.53**)	93.18 (**54.29**)	93.53 (**96.48**)	0.1	75.2 (**50.78**)	79.68 (**47.26**)	72.42 (**53.12**)	70.92 (**54.29**)
	1000	93.09 (**100**)	92.89 (**53.90**)	92.85 (**49.60**)	92.77 (**59.37**)	1	10.00 (**44.53**)	10.00 (**53.51**)	10.00 (**48.04**)	10.00 (**53.51**)

Table 5: Comparison of resistance to overwriting attacks at various trade-off hyper-parameters (λ) and learning rates (η) using ResNet-18. Values (%) inside and outside the bracket are the watermark detection rate and classification accuracy, respectively. Adversary watermarks, which are consistently detected at 100%, are omitted.

Overwriting Attacks

Experiments: Robustness Evaluation

Teleinfo

Fine-tuning	Clean		NeuralMark		VanillaMark		GreedyMark		VoteMark	
	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18
CIFAR-100 to CIFAR-10	85.55	89.15	85.35(**100**)	88.83(**100**)	85.48(**91.01**)	89.35(**85.93**)	80.41(**96.48**)	76.15(**94.14**)	84.97(**89.06**)	89.66(**85.54**)
CIFAR-10 to CIFAR-100	58.96	49.74	58.50(**100**)	49.77(**100**)	58.75(**74.21**)	49.97(**70.31**)	51.75(**97.65**)	19.94(**82.42**)	58.81(**80.07**)	49.08(**71.87**)
Caltech-256 to Caltech-101	47.65	74.09	71.29(**100**)	73.12(**100**)	71.56(**100**)	74.03(**100**)	72.04(**100**)	68.45(**100**)	71.62(**100**)	72.47(**99.60**)
Caltech-101 to Caltech-256	40.61	40.00	40.34(**100**)	40.34(**100**)	40.71(**96.09**)	39.04(**93.36**)	40.68(**100**)	36.45(**98.82**)	39.52(**95.31**)	39.73(**93.75**)

Table 6: Comparison of resistance to fine-tuning attacks using ResNet-18. Values (%) inside and outside the bracket are the watermark detection rate and classification accuracy, respectively.

Fine-tuning Attacks

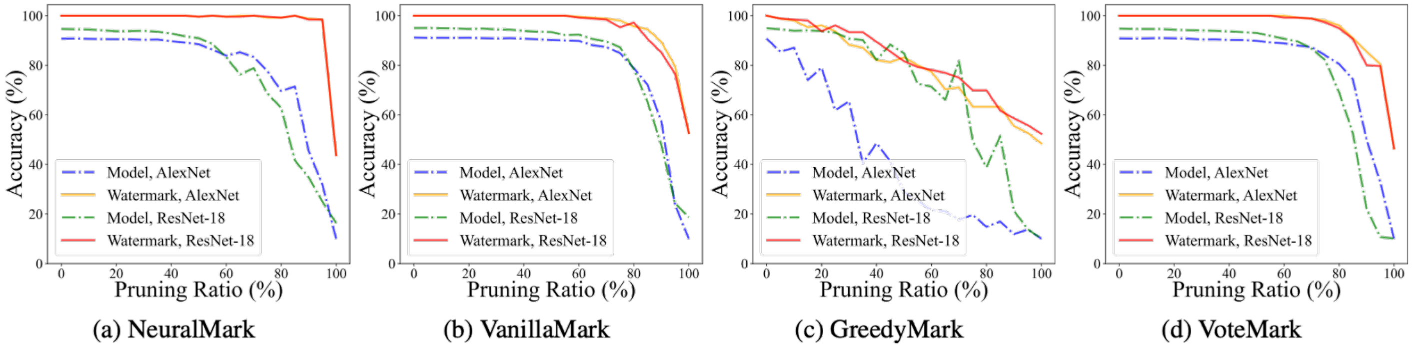


Figure 6: Comparison of resistance to pruning attacks under various pruning ratios on CIFAR-10 using AlexNet and ResNet-18.

Pruning Attacks

Experiments: Analysis

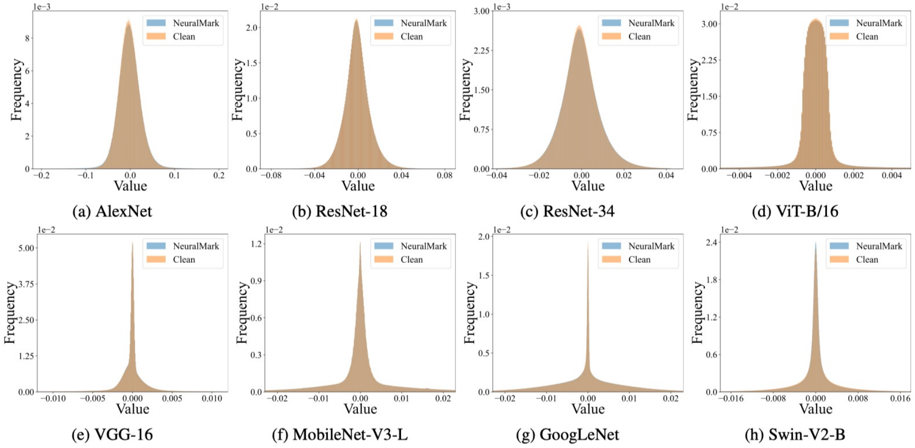


Figure 10: Comparison of parameter distributions on CIFAR-100 with distinct architectures.

Parameter Distribution

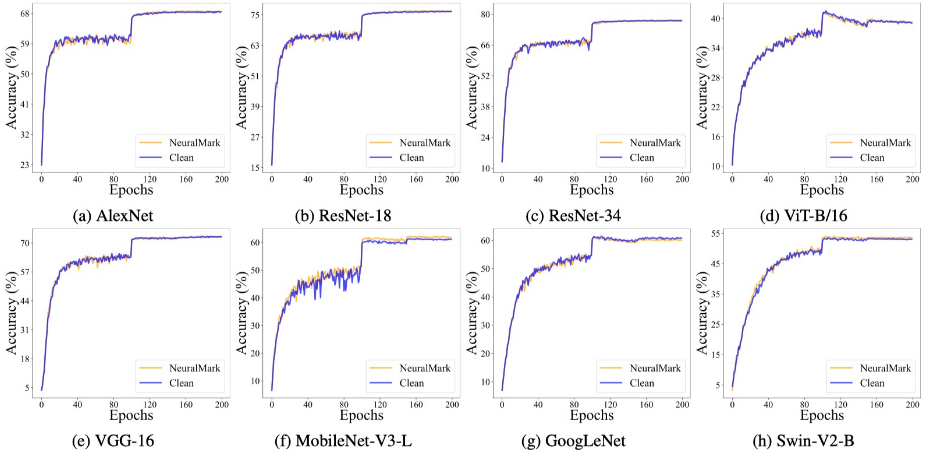


Figure 11: Comparison of model performance convergence across distinct architectures on CIFAR-100.

Performance Convergence

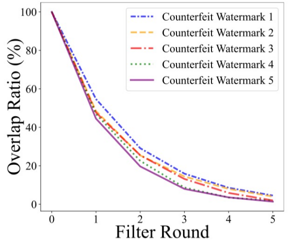


Figure 4: Comparison of parameter overlap ratio with different filter rounds on CIFAR-100 using ResNet-18.

Filtering Rounds

Table 12: Comparison of average time cost (in seconds) on CIFAR-100 using ResNet-18. Here, R denotes the number of filtering rounds.

Method	Clean	NeuralMark ($R = 1$)	NeuralMark ($R = 2$)	NeuralMark ($R = 3$)	NeuralMark ($R = 4$)	VanillaMark	GreedyMark	VoteMark
Time (s)	23.60	24.49	24.94	25.01	25.19	24.34	47.43	35.17

Training Efficiency

Thank you all for your time and participation!

Paper: https://arxiv.org/pdf/2507.11137

Code: https://github.com/AIResearchGroup/NeuralMark

Contact: yaoyuan.hitsz@gmail.com