Improving Robustness Against Stealthy Weight Bit-Flip Attacks by Output Code Matching

Ozan Özdenizci [1 2] and Robert Legenstein [1]

[1] Institute of Theoretical Computer Science, Graz University of Technology, Graz, Austria
[2] TU Graz - SAL Dependable Embedded Systems Lab, Silicon Austria Labs, Graz, Austria

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2022

Introduction & Motivation

- Deep neural networks are susceptible to **adversarial weight bit-flip attacks.**
  - … through hardware-induced fault injection on DNN memory.

Introduction & Motivation

- Deep neural networks are susceptible to **adversarial weight bit-flip attacks.**
  - … through hardware-induced fault injection on DNN memory.
- A recent ***concerning*** threat: finding minimal ***targeted*** and ***stealthy*** bit-flips.
  - Targeting an attacked source (i.e., a single sample or samples of a class).

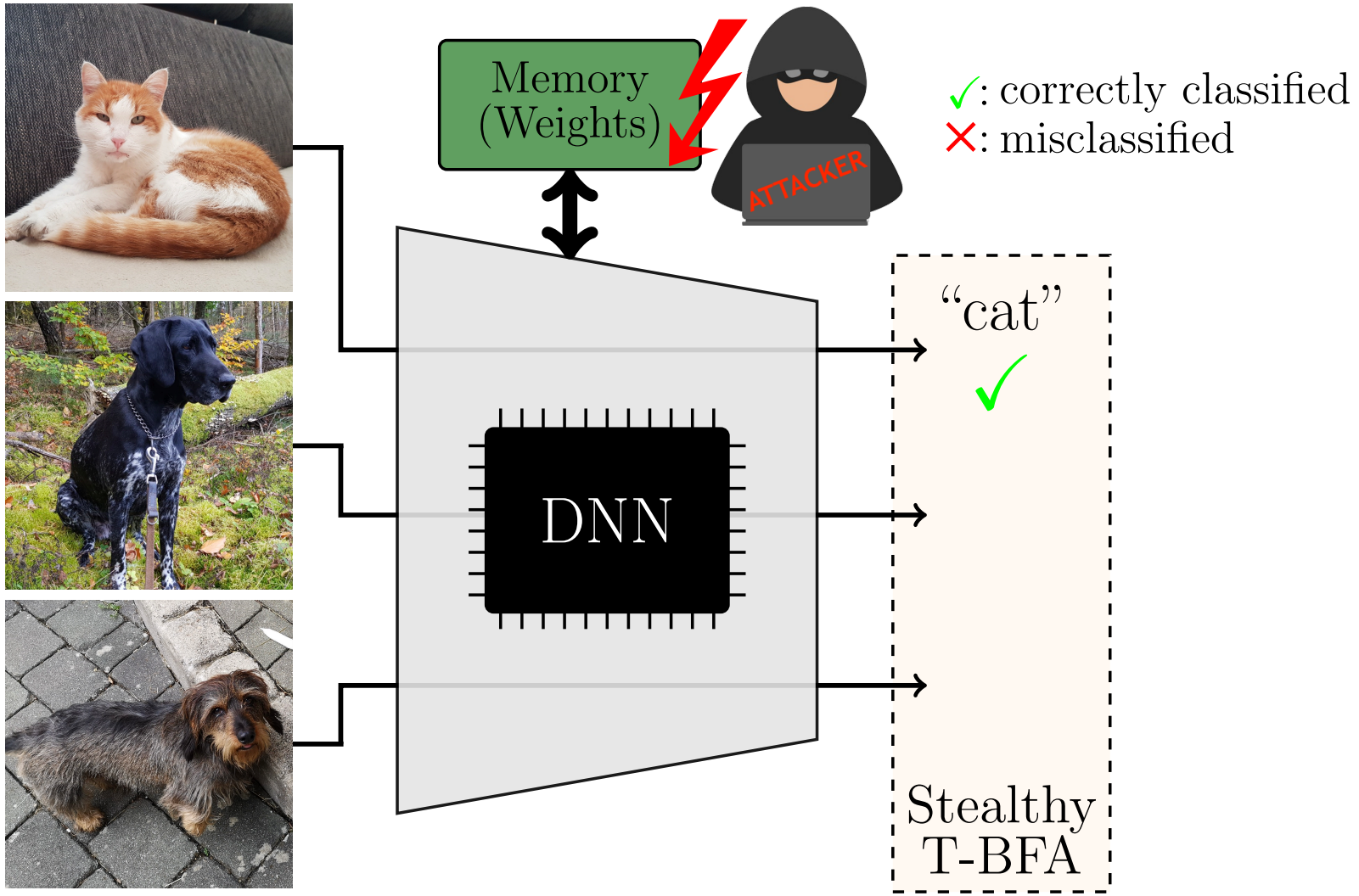
Introduction & Motivation

- Deep neural networks are susceptible to **adversarial weight bit-flip attacks.**
  - … through hardware-induced fault injection on DNN memory.
- A recent ***concerning*** threat: finding minimal ***targeted*** and ***stealthy*** bit-flips.
  - Targeting an attacked source (i.e., a single sample or samples of a class).
  - Preserving expected behavior for un-targeted test samples.

Renders the attack undetectable when there is no unusual activity.

[1] Liu et al., "Fault injection attack on deep neural network", ICCAD 2017.
[2] Zhao et al., "Fault sneaking attack: A stealthy framework for misleading deep neural networks", DAC 2019.

Stealthy Targeted Bit-Flip Attack (T-BFA)



Stealthy T-BFA:

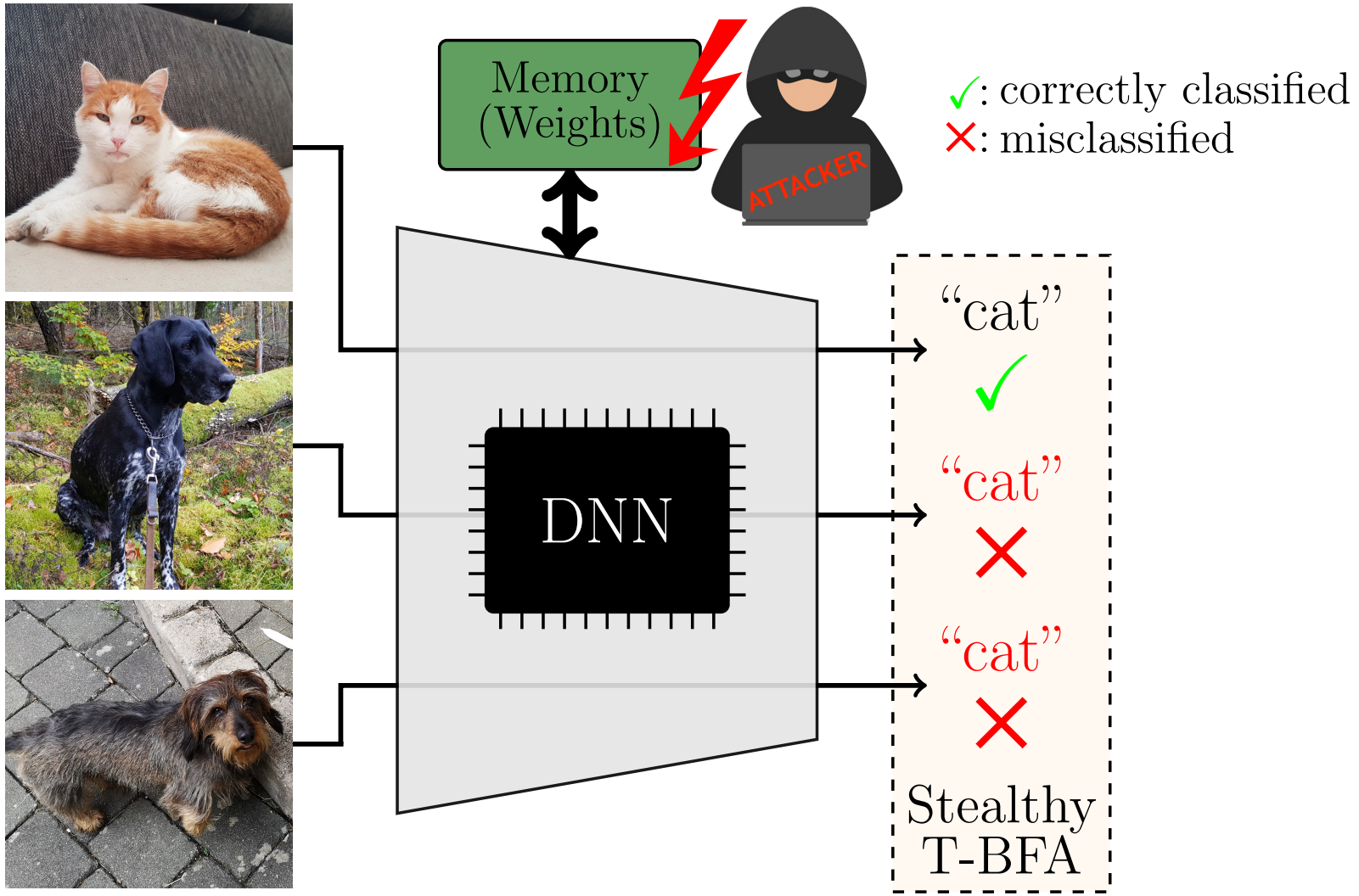
source class

↓

target class

[3] Rakin et al., "T-BFA: Targeted bit-flip adversarial weight attack", IEEE TPAMI 2021.

Stealthy Targeted Bit-Flip Attack (T-BFA)



Stealthy T-BFA:

source class

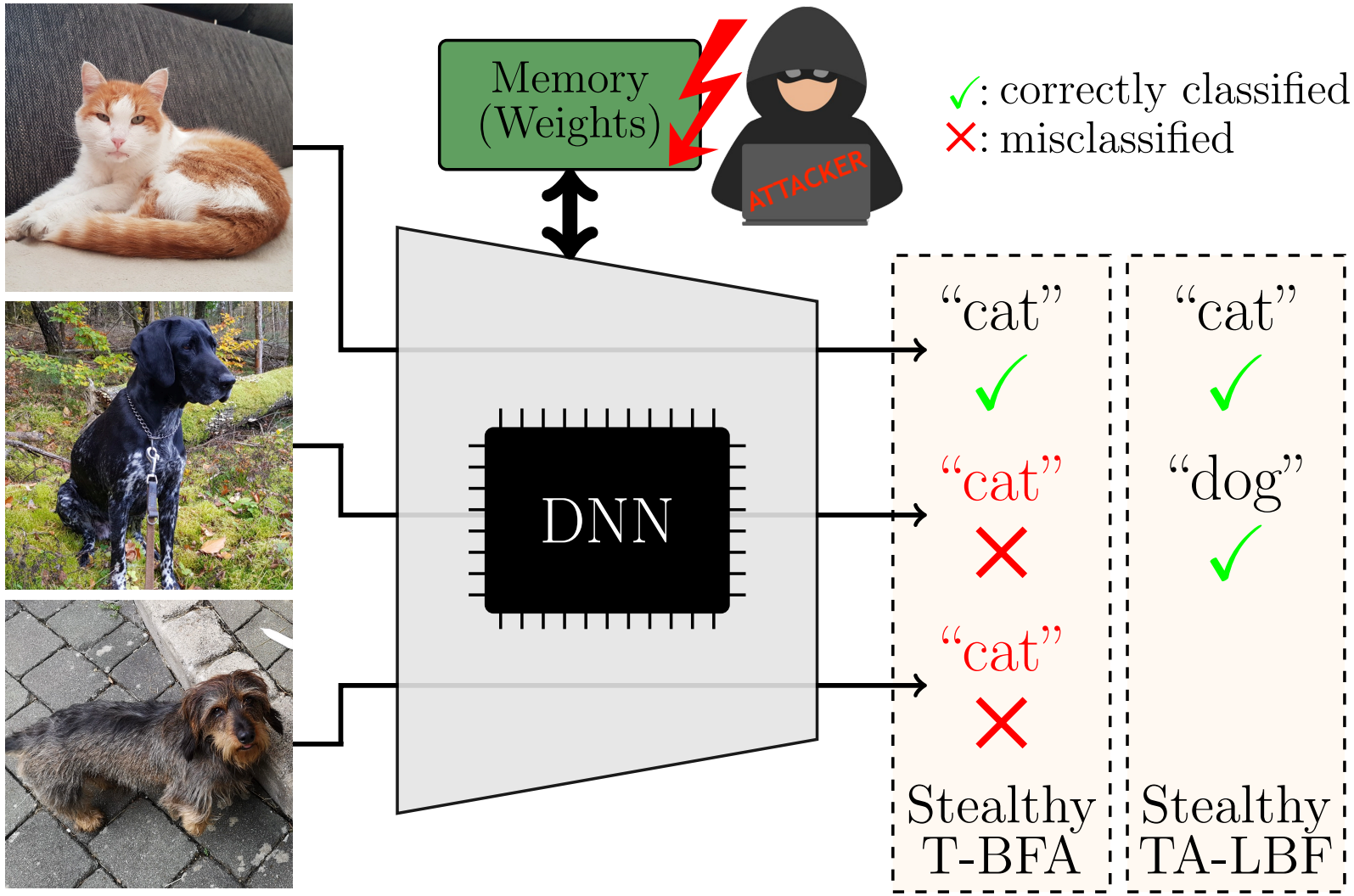
↓

target class

"dog" -> "cat"

[3] Rakin et al., "T-BFA: Targeted bit-flip adversarial weight attack", IEEE TPAMI 2021.

Stealthy Targeted Attack with Limited Bit-Flips (TA-LBF)



Stealthy TA-LBF:

source sample

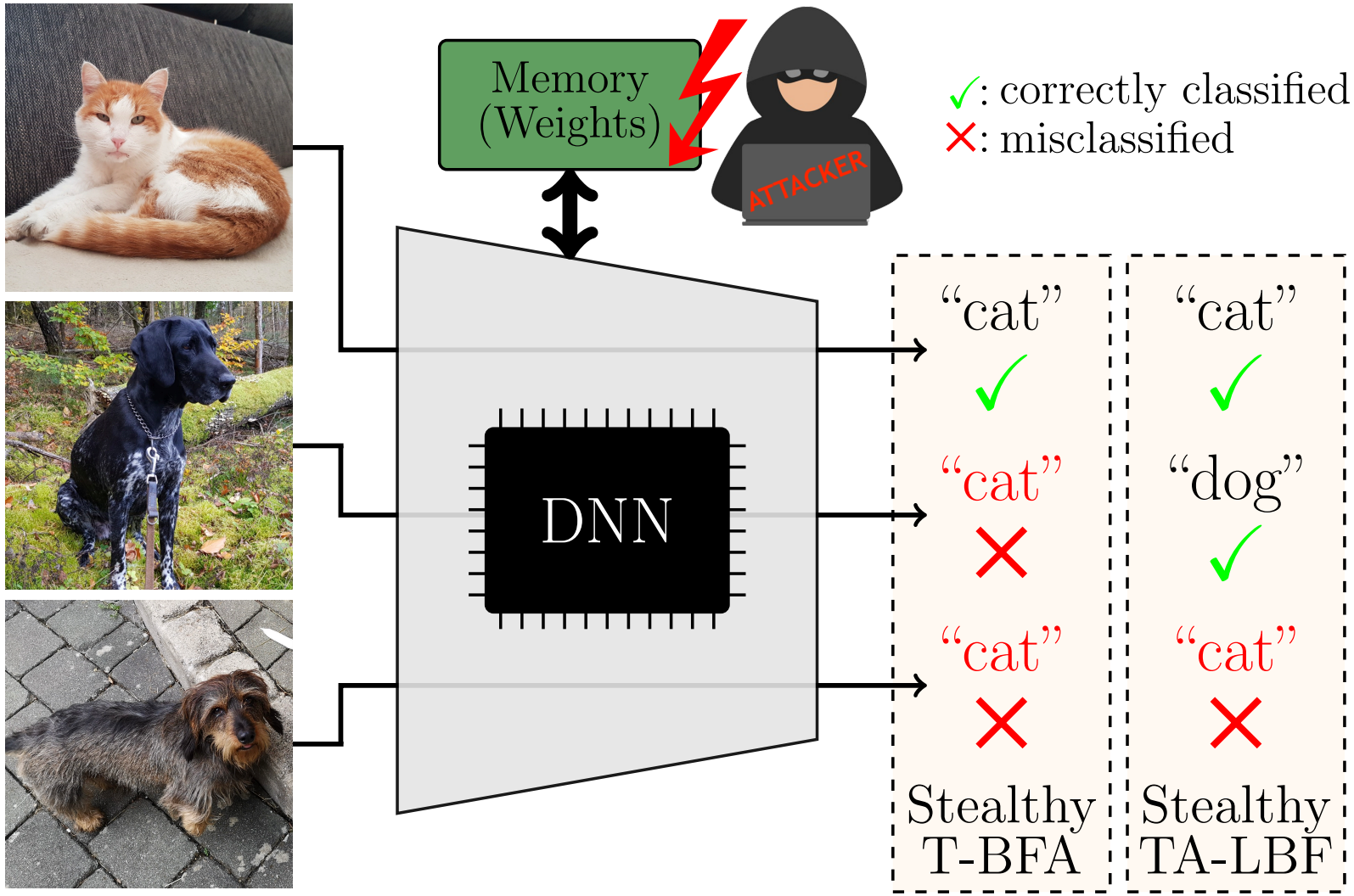
↓

target class

[3] Rakin et al., "T-BFA: Targeted bit-flip adversarial weight attack", IEEE TPAMI 2021.

[4] Bai et al., "Targeted attack against deep neural networks via flipping limited weight bits", ICLR 2021.

Stealthy Targeted Attack with Limited Bit-Flips (TA-LBF)



Stealthy TA-LBF:

source sample

↓

target class

last dog -> "cat"

[3] Rakin et al., "T-BFA: Targeted bit-flip adversarial weight attack", IEEE TPAMI 2021.

[4] Bai et al., "Targeted attack against deep neural networks via flipping limited weight bits", ICLR 2021.

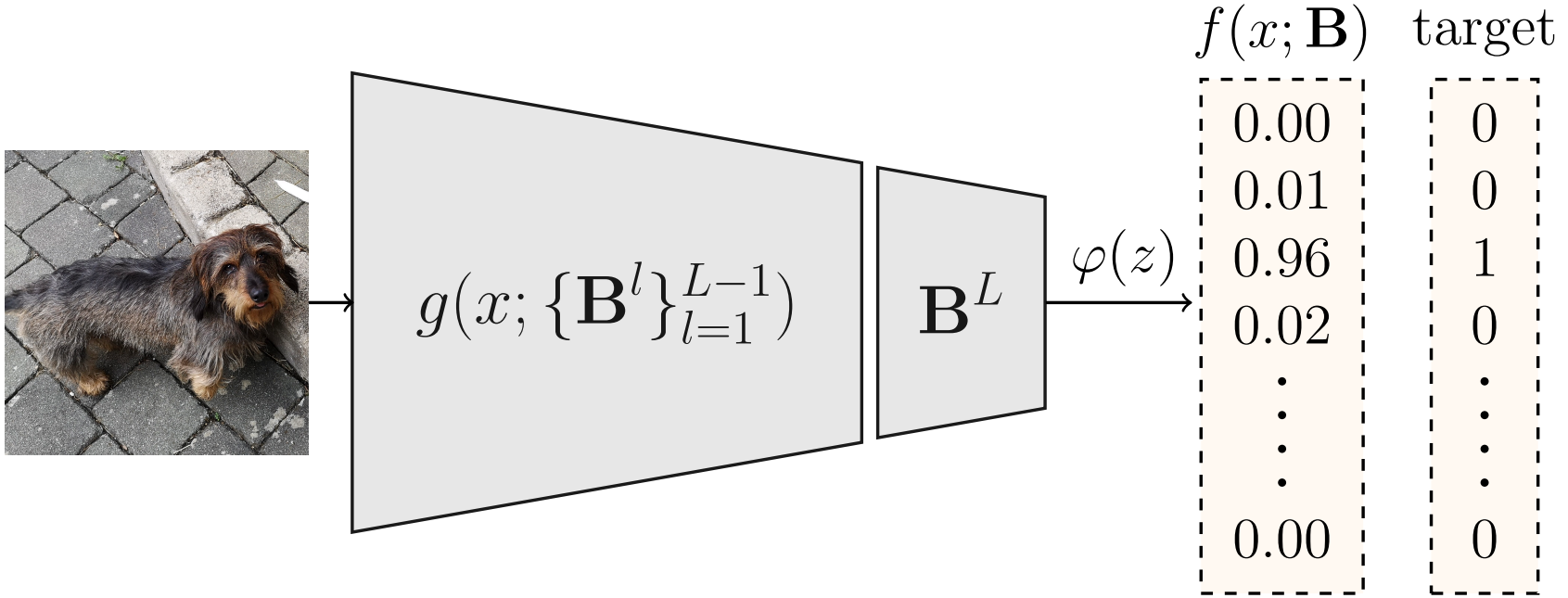
Introduction & Motivation

- Deep neural networks are susceptible to **adversarial weight bit-flip attacks.**
  - … through hardware-induced fault injection on DNN memory.
- A recent ***concerning*** threat: finding minimal ***targeted*** and ***stealthy*** bit-flips.
  - Targeting an attacked source (i.e., a single sample or samples of a class).
  - Preserving expected behavior for un-targeted test samples.

    ⇨ *Renders the attack undetectable when there is no unusual activity.*
- No effective defense tailored against ***stealthy*** adversarial bit-flips exists.

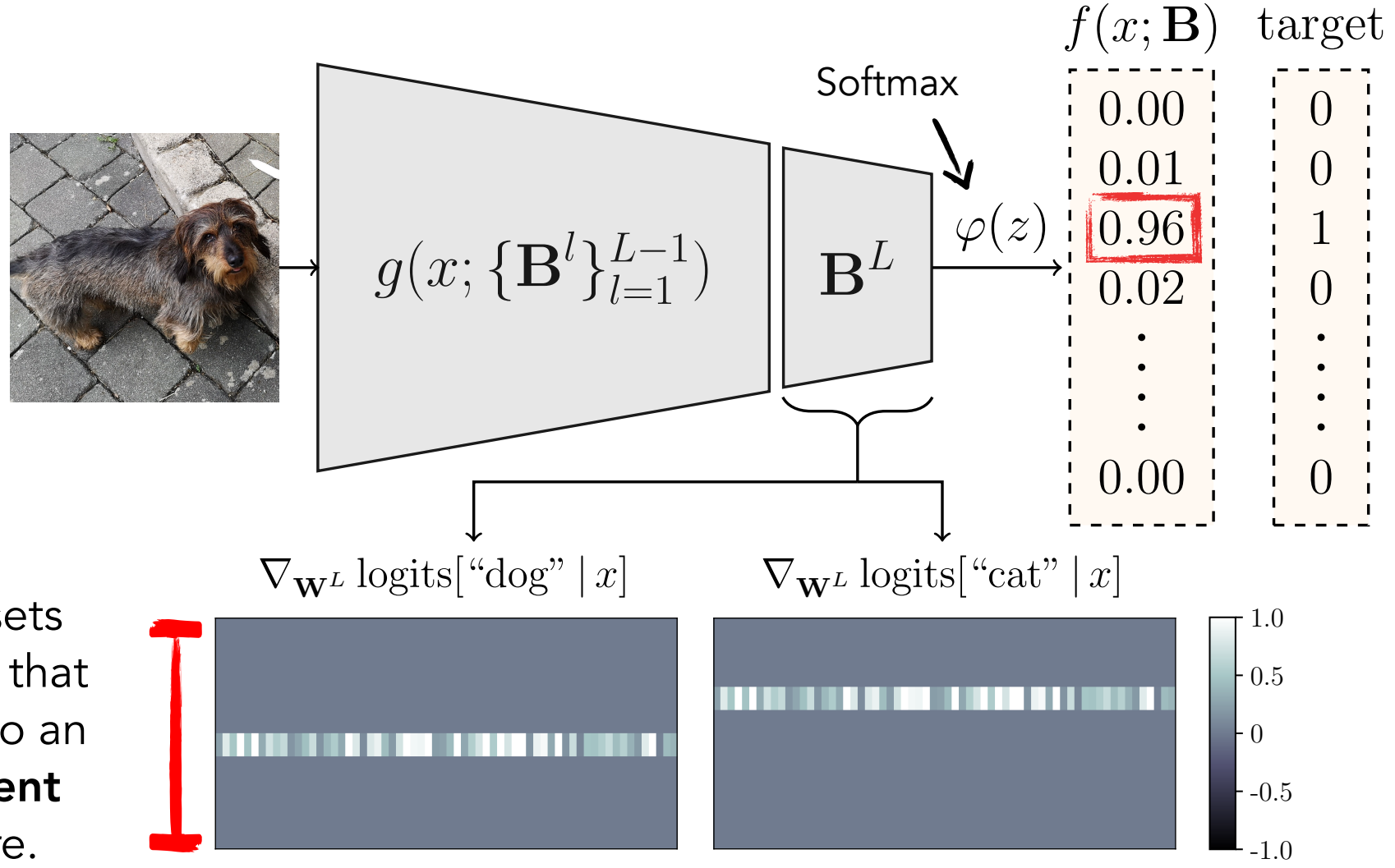
⇨ *How can we confront the **stealthiness** objective of an attacker for such targeted attacks assuming a well-informed adversary?*

Standard One-hot Output Encoding (Vanilla)



Output dim.: *C*
(*C*: #classes)

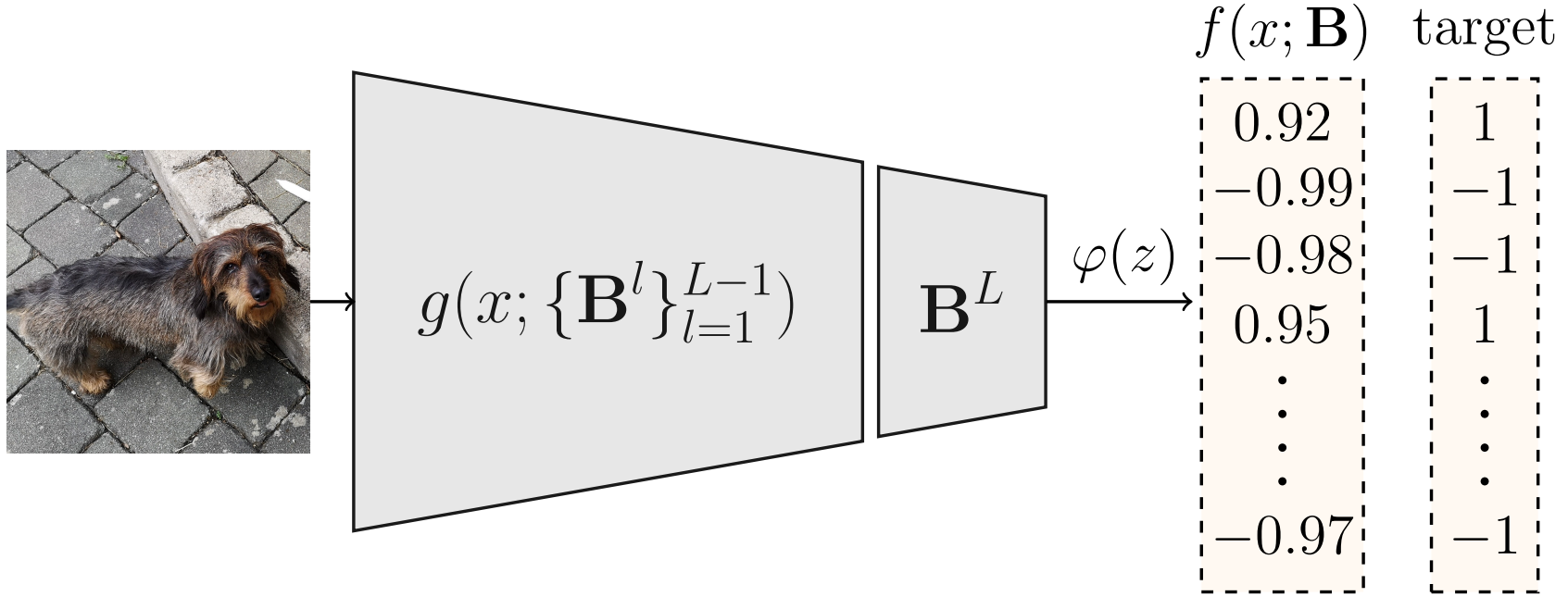
Standard One-hot Output Encoding (Vanilla)



Output dim.: *C*
(*C*: #classes)

Each row sets the weights that contribute to an **independent** class score.

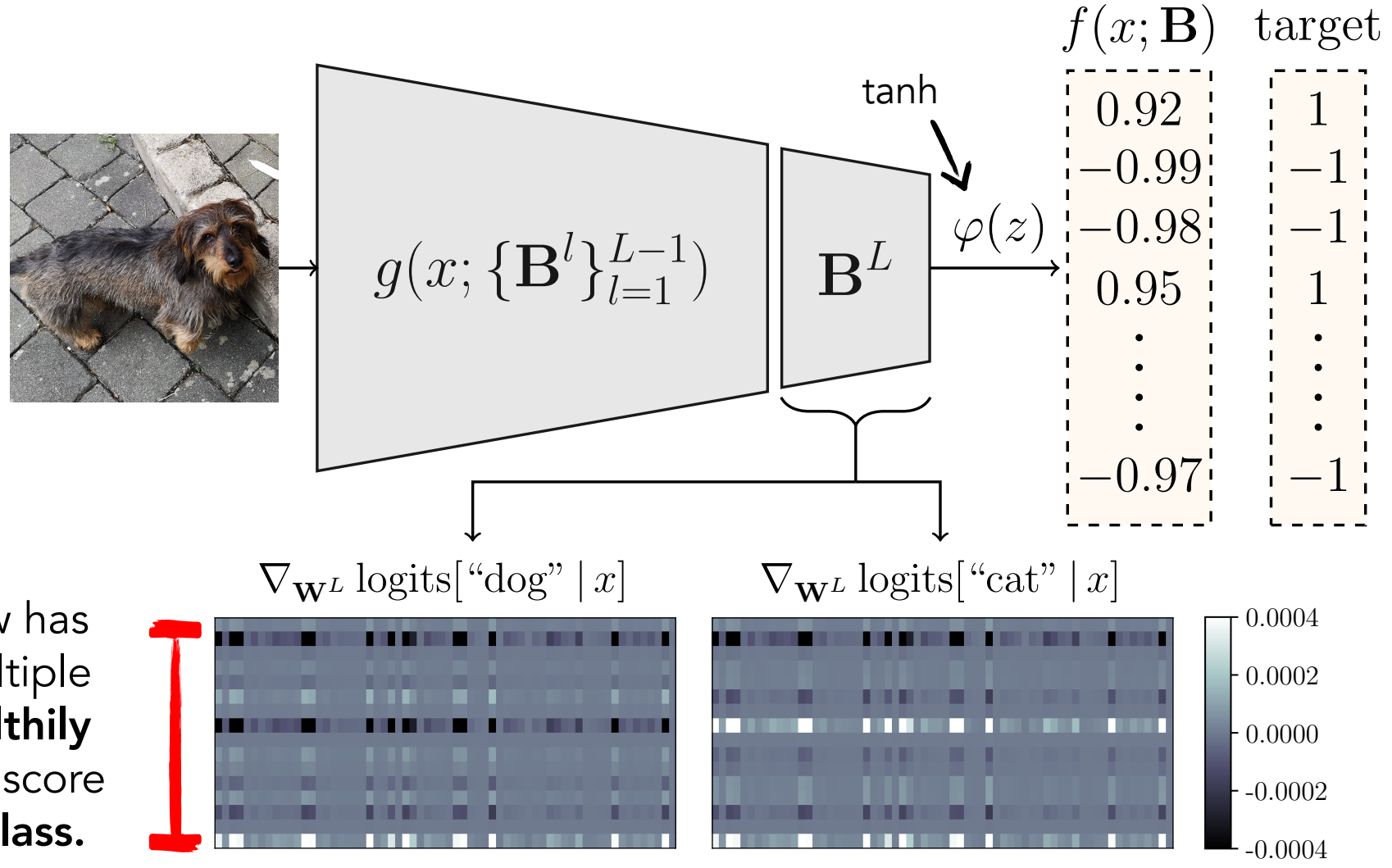
Proposed Output Code Matching (OCM)



*What if we use an **output coding** scheme where the usual one-hot encoding is replaced by **partially overlapping bit strings**?*

Motivation: For any occurring bit-flip to be ***non-stealthy***, ideally all class scores should change their values for any input.

Proposed Output Code Matching (OCM)



Output dim.: *N*
(*N*: code length)

Attacker now has to target multiple rows to **stealthily** influence the score of **a single class.**

Proposed Output Code Matching (OCM)

$g(x; \{\mathbf{B}^l\}_{l=1}^{L-1})$

$\mathbf{B}^L$

tanh

$\varphi(z)$

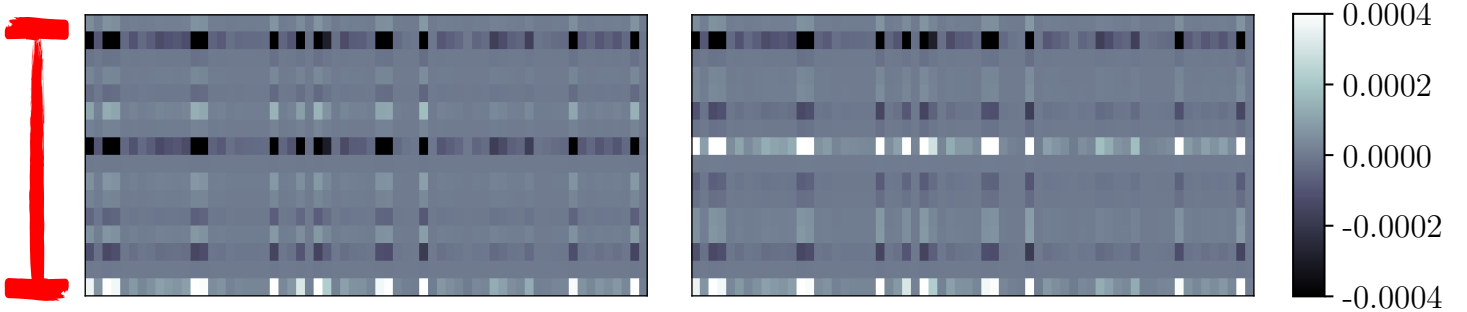
$f(x; \mathbf{B})$ target

$f(x;\mathbf{B})$	target
0.92	1
−0.99	−1
−0.98	−1
0.95	1
⋮	⋮
−0.97	−1

Output dim.: *N*
(*N*: code length)

$\nabla_{\mathbf{W}^L}$ logits["dog" | x]

$\nabla_{\mathbf{W}^L}$ logits["cat" | x]

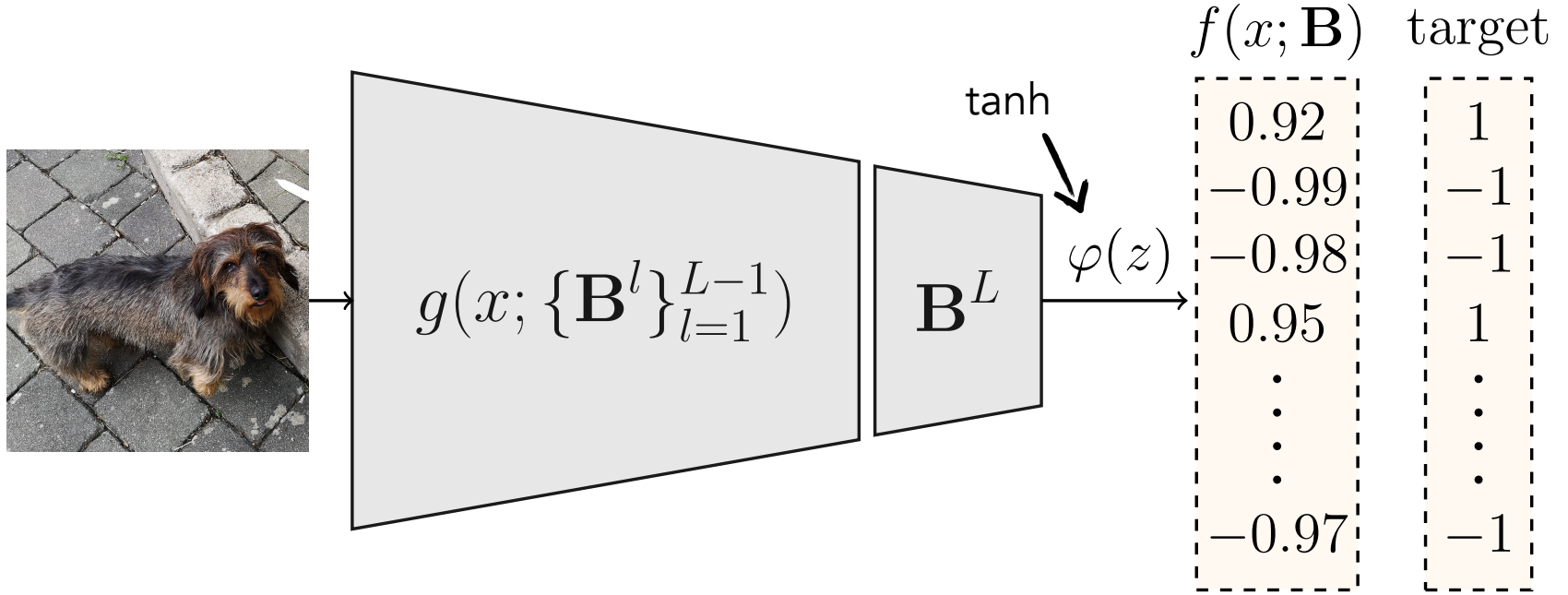


Attacker now has to target multiple rows to **stealthily** influence the score of **a single class.**

Increasing uncertainty across several classes in the face of adversity.

… which will also lead to changes for other class scores as codes are overlapping.

Proposed Output Code Matching (OCM)



Output dim.: *N*
(*N*: code length)

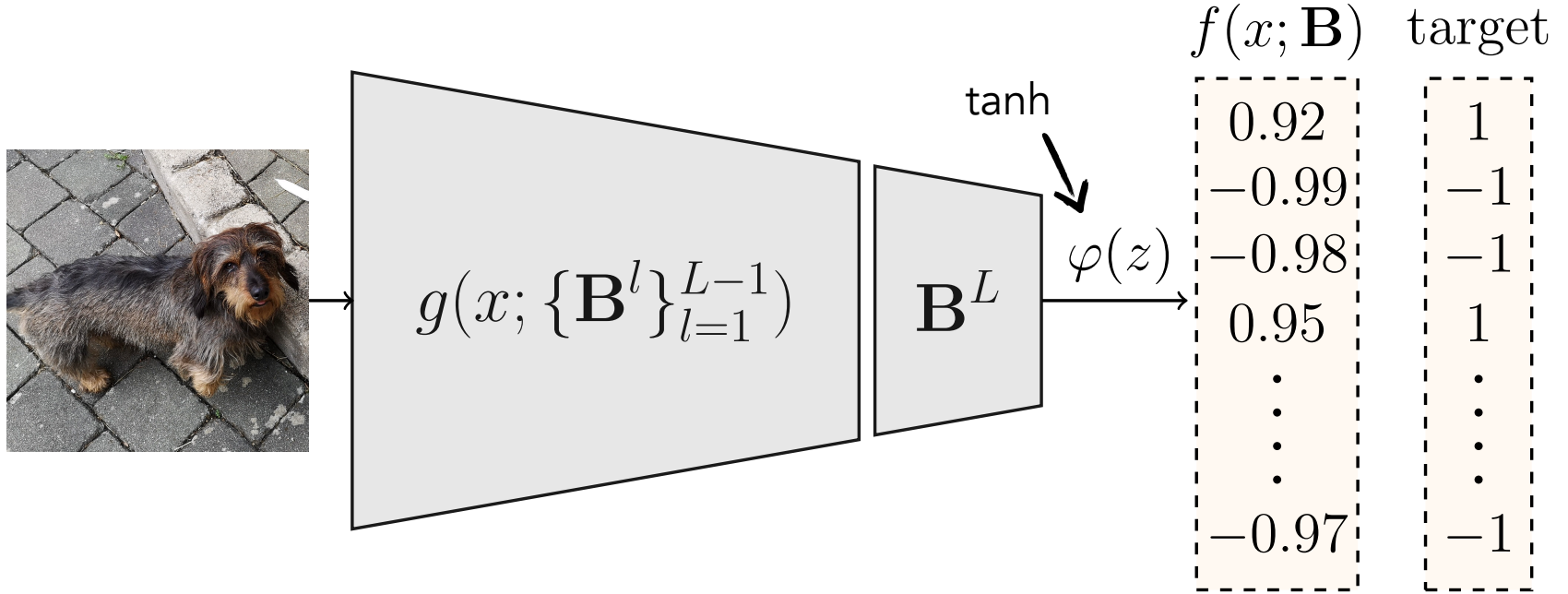
Bit string code design: we use Hadamard matrices.

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \; \mathbf{H}_{2^k} = \begin{bmatrix} \mathbf{H}_{2^{k-1}} & \mathbf{H}_{2^{k-1}} \\ \mathbf{H}_{2^{k-1}} & -\mathbf{H}_{2^{k-1}} \end{bmatrix}$$

Overlap between any given pair of class codes is *N/2* (*N*: code length).

$$\begin{aligned}
\mathbf{S}_1 &= [\,1\;1\;1\;1\;-1\;-1\;-1\;-1\;1\;1\;1\;1\;-1\;-1\;-1\;-1\,], \\
\mathbf{S}_2 &= [\,1\;1\;1\;1\;1\;1\;1\;1\;-1\;-1\;-1\;-1\;-1\;-1\;-1\;-1\,], \\
\mathbf{S}_3 &= [\,1\;-1\;-1\;1\;-1\;1\;1\;-1\;1\;-1\;-1\;1\;-1\;1\;1\;-1\,], \\
\mathbf{S}_4 &= [\,1\;-1\;-1\;1\;1\;-1\;-1\;1\;1\;-1\;-1\;1\;1\;-1\;-1\;1\,], \\
\mathbf{S}_5 &= [\,1\;-1\;-1\;1\;1\;-1\;-1\;1\;-1\;1\;1\;-1\;-1\;1\;1\;-1\,], \\
\mathbf{S}_6 &= [\,1\;-1\;1\;-1\;-1\;1\;-1\;1\;1\;-1\;1\;-1\;-1\;1\;-1\;1\,], \\
\mathbf{S}_7 &= [\,1\;1\;-1\;-1\;1\;1\;-1\;-1\;1\;1\;-1\;-1\;1\;1\;-1\;-1\,], \\
\mathbf{S}_8 &= [\,1\;1\;-1\;-1\;-1\;-1\;1\;1\;-1\;-1\;1\;1\;1\;1\;-1\;-1\,], \\
\mathbf{S}_9 &= [\,1\;-1\;1\;-1\;1\;-1\;1\;-1\;1\;-1\;1\;-1\;1\;-1\;1\;-1\,], \\
\mathbf{S}_{10} &= [\,1\;-1\;-1\;1\;-1\;1\;1\;-1\;-1\;1\;1\;-1\;1\;-1\;-1\;1\,].
\end{aligned}$$

Proposed Output Code Matching (OCM)



Output dim.: *N*
(*N*: code length)

Bit string code design: we use Hadamard matrices.

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \; \mathbf{H}_{2^k} = \begin{bmatrix} \mathbf{H}_{2^{k-1}} & \mathbf{H}_{2^{k-1}} \\ \mathbf{H}_{2^{k-1}} & -\mathbf{H}_{2^{k-1}} \end{bmatrix}$$

Overlap between any given pair of class codes is *N/2* (*N*: code length).

Training: $\mathcal{L}_{\text{OCM}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}}\left[\,|f(x;\mathbf{B}) - \mathbf{S}_y|\,\right]$

Inference: $\arg\max_y [\mathbf{S}_y \cdot f(x;\mathbf{B})]$

Results on CIFAR-10

Table 1. Evaluations of 8-bit and 4-bit quantized ResNet-20 models under stealthy weight bit-flip attacks for CIFAR-10. Test set clean accuracy, ASR and PA-ACC percentages (%) are presented alongside # bit-flips needed for the attack. Stealthy T-BFA attacks [26] are run until all source class set examples used by the attacker are misclassified, and all stealthy T-BFA evaluation metrics are averaged across 100 targeted attack experiments. Stealthy TA-LBF attacks [3] are performed for 1000 single sample attacks, where each one of the 10 classes is the target class for 100 different source images that belong to any other class.

			Vanilla	Piecewise Clustering [16]	Ours		
					OCM$_{16}$	**OCM$_{32}$**	**OCM$_{64}$**
ResNet-20 (8-bit)	Clean Acc. on **CIFAR-10**		**92.25**	91.11	90.67	90.72	90.26
	Stealthy T-BFA [26]	ASR (↘)	**99.10**	99.46	99.48	99.56	99.58
		PA-ACC (↘)	84.38 (3.39)	76.78 (7.45)	53.22 (21.5)	50.01 (18.2)	**46.39 (16.7)**
		# bit-flips (↗)	27.91 (8.70)	74.93 (26.7)	95.65 (32.4)	127.88 (54.0)	**281.75 (115.6)**
	Stealthy TA-LBF [3]	ASR (↘)					
		PA-ACC (↘)					
		# bit-flips (↗)					

~10x more # bit-flips needed

***Attacks** are well-informed about the defense, i.e., uses the L1-norm objective and class-specific codes.*

Results on CIFAR-10

Table 1. Evaluations of 8-bit and 4-bit quantized ResNet-20 models under stealthy weight bit-flip attacks for CIFAR-10. Test set clean accuracy, ASR and PA-ACC percentages (%) are presented alongside # bit-flips needed for the attack. Stealthy T-BFA attacks [26] are run until all source class set examples used by the attacker are misclassified, and all stealthy T-BFA evaluation metrics are averaged across 100 targeted attack experiments. Stealthy TA-LBF attacks [3] are performed for 1000 single sample attacks, where each one of the 10 classes is the target class for 100 different source images that belong to any other class.

			Vanilla	Piecewise Clustering [16]	Ours		
					OCM$_{16}$	**OCM$_{32}$**	**OCM$_{64}$**
ResNet-20 (8-bit)	Clean Acc. on **CIFAR-10**		**92.25**	91.11	90.67	90.72	90.26
	Stealthy T-BFA [26]	ASR (↘)	**99.10**	99.46	99.48	99.56	99.58
		PA-ACC (↘)	84.38 (3.39)	76.78 (7.45)	53.22 (21.5)	50.01 (18.2)	**46.39 (16.7)**
		# bit-flips (↗)	27.91 (8.70)	74.93 (26.7)	95.65 (32.4)	127.88 (54.0)	**281.75 (115.6)**
	Stealthy TA-LBF [3]	ASR (↘)					
		PA-ACC (↘)					
		# bit-flips (↗)					

~4x more # bit-flips needed

[5] He et al., "Defending and harnessing the bit-flip based adversarial weight attack", CVPR 2020.

Results on CIFAR-10

Table 1. Evaluations of 8-bit and 4-bit quantized ResNet-20 models under stealthy weight bit-flip attacks for CIFAR-10. Test set clean accuracy, ASR and PA-ACC percentages (%) are presented alongside # bit-flips needed for the attack. Stealthy T-BFA attacks [26] are run until all source class set examples used by the attacker are misclassified, and all stealthy T-BFA evaluation metrics are averaged across 100 targeted attack experiments. Stealthy TA-LBF attacks [3] are performed for 1000 single sample attacks, where each one of the 10 classes is the target class for 100 different source images that belong to any other class.

					Ours		
			Vanilla	Piecewise Clustering [16]	$\mathbf{OCM}_{16}$	$\mathbf{OCM}_{32}$	$\mathbf{OCM}_{64}$
ResNet-20 (8-bit)	Clean Acc. on **CIFAR-10**		**92.25**	91.11	90.67	90.72	90.26
	Stealthy T-BFA [26]	ASR (↘)	**99.10**	99.46	99.48	99.56	99.58
		PA-ACC (↘)	84.38 (3.39)	76.78 (7.45)	53.22 (21.5)	50.01 (18.2)	**46.39 (16.7)**
		# bit-flips (↗)	27.91 (8.70)	74.93 (26.7)	95.65 (32.4)	127.88 (54.0)	**281.75 (115.6)**
	Stealthy TA-LBF [3]	ASR (↘)					
		PA-ACC (↘)					
		# bit-flips (↗)					

Breaking stealthiness

PA-ACC decreases from 90% to 46%

Results on CIFAR-10

Table 1. Evaluations of 8-bit and 4-bit quantized ResNet-20 models under stealthy weight bit-flip attacks for CIFAR-10. Test set clean accuracy, ASR and PA-ACC percentages (%) are presented alongside # bit-flips needed for the attack. Stealthy T-BFA attacks [26] are run until all source class set examples used by the attacker are misclassified, and all stealthy T-BFA evaluation metrics are averaged across 100 targeted attack experiments. Stealthy TA-LBF attacks [3] are performed for 1000 single sample attacks, where each one of the 10 classes is the target class for 100 different source images that belong to any other class.

			Vanilla	Piecewise Clustering [16]	**Ours**		
					OCM_{16}	**OCM_{32}**	**OCM_{64}**
ResNet-20 (8-bit)	Clean Acc. on **CIFAR-10**		**92.25**	91.11	90.67	90.72	90.26
	Stealthy T-BFA [26]	ASR (↘)	**99.10**	99.46	99.48	99.56	99.58
		PA-ACC (↘)	84.38 (3.39)	76.78 (7.45)	53.22 (21.5)	50.01 (18.2)	**46.39 (16.7)**
		# bit-flips (↗)	27.91 (8.70)	74.93 (26.7)	95.65 (32.4)	127.88 (54.0)	**281.75 (115.6)**
	Stealthy TA-LBF [3]	ASR (↘)	100.00	100.00	97.60	98.20	**72.40**
		PA-ACC (↘)	88.06 (2.55)	87.64 (2.09)	86.45 (3.31)	86.07 (3.26)	**84.08 (3.18)**
		# bit-flips (↗)	5.42 (0.91)	18.14 (7.05)	31.12 (10.3)	47.52 (13.7)	**73.65 (15.67)**

Results on ImageNet

by only OCM finetuning of pre-trained models...

Table 3. Stealthy T-BFA [26] evaluations with 8-bit and 4-bit quantized ResNet-50 models on ImageNet. Attacks are run until all source class set examples used by the attacker are misclassified. Test set clean accuracy, ASR and PA-ACC percentages (%) are presented alongside # bit-flips needed to attack. All evaluation metrics are averaged across 500 targeted attack experiments.

			Vanilla	Piecewise Clustering [16]		Ours	
				$\lambda = 0.0001$	$\lambda = 0.0005$	**OCM$_{1024}$**	**OCM$_{2048}$**
ResNet-50 (8-bit)	Clean Acc. on **ImageNet**		**75.92**	74.64	68.73	72.71	73.25
	Stealthy T-BFA [26]	ASR (↘)	94.74	91.29	**89.32**	91.35	92.37
		PA-ACC (↘)	68.64 (9.25)	57.64 (11.4)	54.81 (10.2)	50.93 (10.9)	**50.63 (11.3)**
		# bit-flips (↗)	7.69 (3.88)	26.24 (13.8)	48.65 (17.0)	121.26 (297.3)	**145.05 (366.4)**

~20x more # bit-flips needed

Results on ImageNet

by only OCM finetuning of pre-trained models...

Table 3. Stealthy T-BFA [26] evaluations with 8-bit and 4-bit quantized ResNet-50 models on ImageNet. Attacks are run until all source class set examples used by the attacker are misclassified. Test set clean accuracy, ASR and PA-ACC percentages (%) are presented alongside # bit-flips needed to attack. All evaluation metrics are averaged across 500 targeted attack experiments.

			Vanilla	Piecewise Clustering [16]		Ours	
				$\lambda = 0.0001$	$\lambda = 0.0005$	**OCM$_{1024}$**	**OCM$_{2048}$**
ResNet-50 (8-bit)	Clean Acc. on **ImageNet**		**75.92**	74.64	68.73	72.71	73.25
	Stealthy T-BFA [26]	ASR (↘)	94.74	91.29	**89.32**	91.35	92.37
		PA-ACC (↘)	68.64 (9.25)	57.64 (11.4)	54.81 (10.2)	50.93 (10.9)	**50.63 (11.3)**
		# bit-flips (↗)	7.69 (3.88)	26.24 (13.8)	48.65 (17.0)	121.26 (297.3)	**145.05 (366.4)**

~3x more # bit-flips needed

Results on ImageNet

by only OCM finetuning of pre-trained models...

Table 3. Stealthy T-BFA [26] evaluations with 8-bit and 4-bit quantized ResNet-50 models on ImageNet. Attacks are run until all source class set examples used by the attacker are misclassified. Test set clean accuracy, ASR and PA-ACC percentages (%) are presented alongside # bit-flips needed to attack. All evaluation metrics are averaged across 500 targeted attack experiments.

			Vanilla	Piecewise Clustering [16]		Ours	
				$\lambda = 0.0001$	$\lambda = 0.0005$	$\mathbf{OCM}_{1024}$	$\mathbf{OCM}_{2048}$
ResNet-50 (8-bit)	Clean Acc. on **ImageNet**		**75.92**	74.64	68.73	72.71	73.25
	Stealthy T-BFA [26]	ASR (↘)	94.74	91.29	**89.32**	91.35	92.37
		PA-ACC (↘)	68.64 (9.25)	57.64 (11.4)	54.81 (10.2)	50.93 (10.9)	**50.63** (11.3)
		# bit-flips (↗)	7.69 (3.88)	26.24 (13.8)	48.65 (17.0)	121.26 (297.3)	**145.05** (366.4)

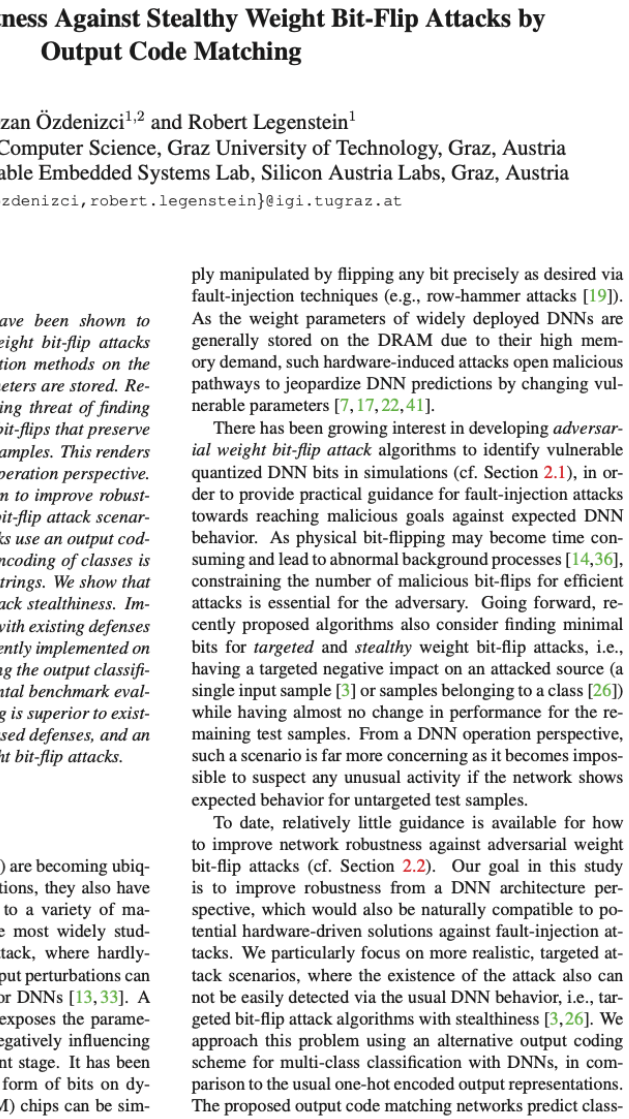
Breaking stealthiness

PA-ACC down to 50%

Thank you for your attention!

Code: https://github.com/IGITUGraz/OutputCodeMatching

Acknowledgments: We thank Jakub Breier and Xiaolu Hou for the fruitful discussions and comments. This work has been supported by the "University SAL Labs" initiative of Silicon Austria Labs (SAL).



Improving Robustness Against Stealthy Weight Bit-Flip Attacks by Output Code Matching

Ozan Özdenizci[1,2] and Robert Legenstein[1]
[1] Institute of Theoretical Computer Science, Graz University of Technology, Graz, Austria
[2] TU Graz - SAL Dependable Embedded Systems Lab, Silicon Austria Labs, Graz, Austria
{ozan.ozdenizci,robert.legenstein}@igi.tugraz.at

Abstract

Deep neural networks (DNNs) have been shown to be vulnerable against adversarial weight bit-flip attacks through hardware-induced fault-injection methods on the memory systems where network parameters are stored. Recent attacks pose the further concerning threat of finding minimal targeted and stealthy weight bit-flips that preserve expected behavior for untargeted test samples. This renders the attack undetectable from a DNN operation perspective. We propose a DNN defense mechanism to improve robustness in such realistic stealthy weight bit-flip attack scenarios. Our output code matching networks use an output coding scheme where the usual one-hot encoding of classes is replaced by partially overlapping bit strings. We show that this encoding significantly reduces attack stealthiness. Importantly, our approach is compatible with existing defenses and DNN architectures. It can be efficiently implemented on pre-trained models by simply re-defining the output classification layer and finetuning. Experimental benchmark evaluations show that output code matching is superior to existing regularized weight quantization based defenses, and an effective defense against stealthy weight bit-flip attacks.

1. Introduction

While deep neural networks (DNNs) are becoming ubiquitous in artificial intelligence applications, they also have been proven to be highly vulnerable to a variety of malicious attack paradigms. One of the most widely studied aspect is the adversarial input attack, where hardly-perceptible and intentionally crafted input perturbations can lead to confident incorrect decisions for DNNs [13, 33]. A recently emerged category of attacks exposes the parameter space vulnerability of DNNs by negatively influencing the inference process at the deployment stage. It has been shown that information stored in the form of bits on dynamic random-access memory (DRAM) chips can be simply manipulated by flipping any bit precisely as desired via fault-injection techniques (e.g., row-hammer attacks [19]). As the weight parameters of widely deployed DNNs are generally stored on the DRAM due to their high memory demand, such hardware-induced attacks open malicious pathways to jeopardize DNN predictions by changing vulnerable parameters [7, 17, 22, 41].

There has been growing interest in developing *adversarial weight bit-flip attack* algorithms to identify vulnerable quantized DNN bits in simulations (cf. Section 2.1), in order to provide practical guidance for fault-injection attacks towards reaching malicious goals against expected DNN behavior. As physical bit-flipping may become time consuming and lead to abnormal background processes [14,36], constraining the number of malicious bit-flips for efficient attacks is essential for the adversary. Going forward, recently proposed algorithms also consider finding minimal bits for *targeted* and *stealthy* weight bit-flip attacks, i.e., having a targeted negative impact on an attacked source (a single input sample [3] or samples belonging to a class [26]) while having almost no change in performance for the remaining test samples. From a DNN operation perspective, such a scenario is far more concerning as it becomes impossible to suspect any unusual activity if the network shows expected behavior for untargeted test samples.

To date, relatively little guidance is available for how to improve network robustness against adversarial weight bit-flip attacks (cf. Section 2.2). Our goal in this study is to improve robustness from a DNN architecture perspective, which would also be naturally compatible to potential hardware-driven solutions against fault-injection attacks. We particularly focus on more realistic, targeted attack scenarios, where the existence of the attack also can not be easily detected via the usual DNN behavior, i.e., targeted bit-flip attack algorithms with stealthiness [3, 26]. We approach this problem using an alternative output coding scheme for multi-class classification with DNNs, in comparison to the usual one-hot encoded output representations. The proposed output code matching networks predict class-