AdvHash: Set-to-set Targeted Attack on Deep Hashing with One Single Adversarial Patch

Shengshan Hu[1], Yechao Zhang[1], Xiaogeng Liu[1], Leo Yu Zhang[2], MingHui Li[3] and Hai Jin[4]

[1] School of Cyber Science and Engineering, Huazhong University of Science and Technology

[2] School of Information Technology, Deakin University

[3] School of Software Engineering, Huazhong University of Science and Technology

[4] School of Computer Science and Technology, Huazhong University of Science and Technology

Presented for ACM Multimedia 2021

Outline

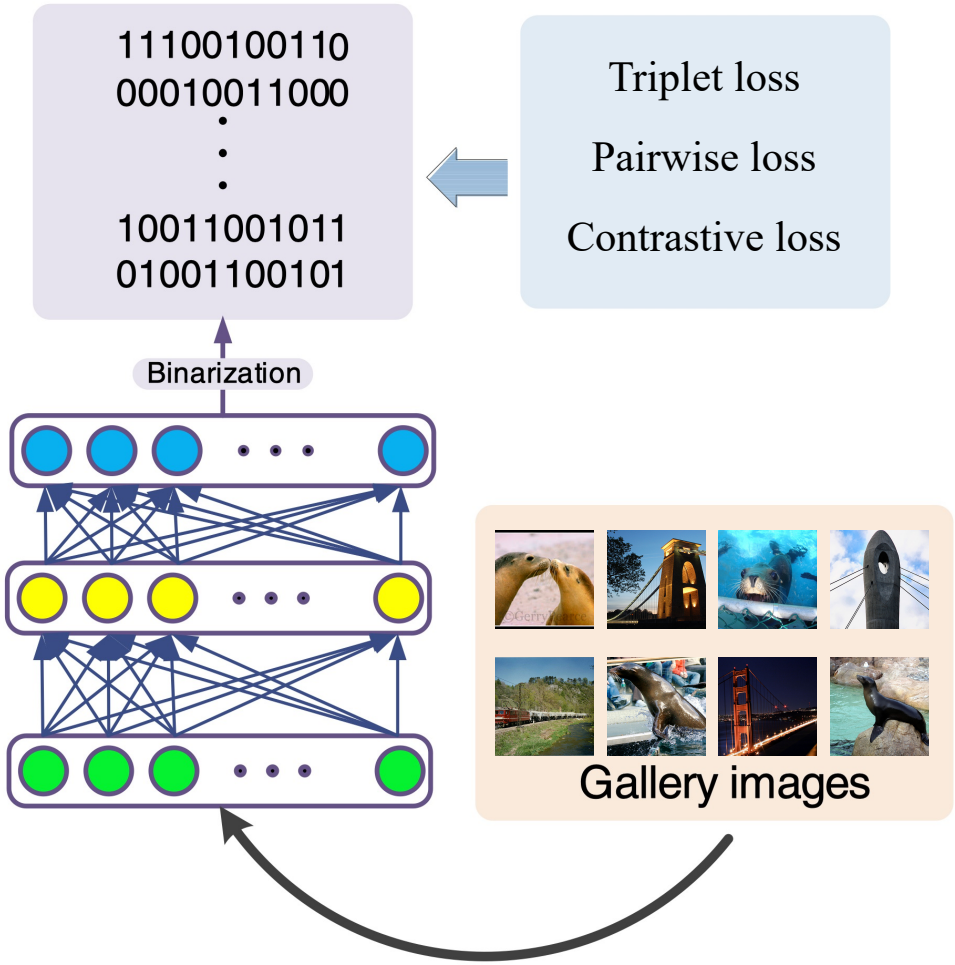
- Introduction & Motivation
- Problem & Methodology
- Evaluation & Comparison
- Conclusion

Introduction of Deep Hashing

Deep Hashing = Metric Learning + Binarization

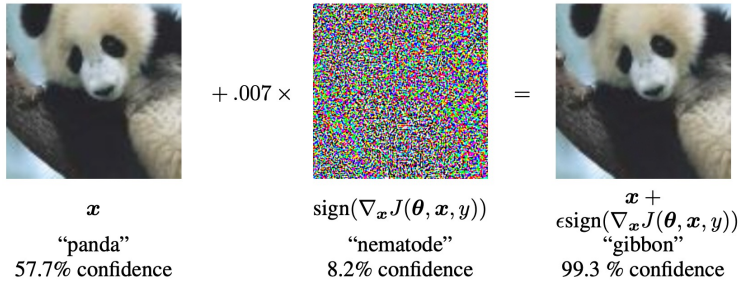
Widespread applications of deep hashing based image retrieval:

- Product visual search e-commerce platform
- Image retrieval in social platform
- Large-scale image search engine

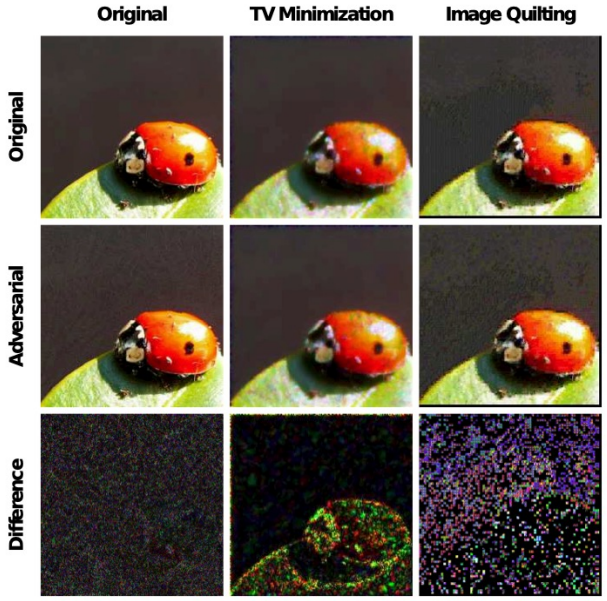


Perturbation v. s. Patch

Perturbation



Invisible yet global



Non-robust to image transformation

Feasible only on digital-world

Patch

Conspicuous yet local

A designed pattern

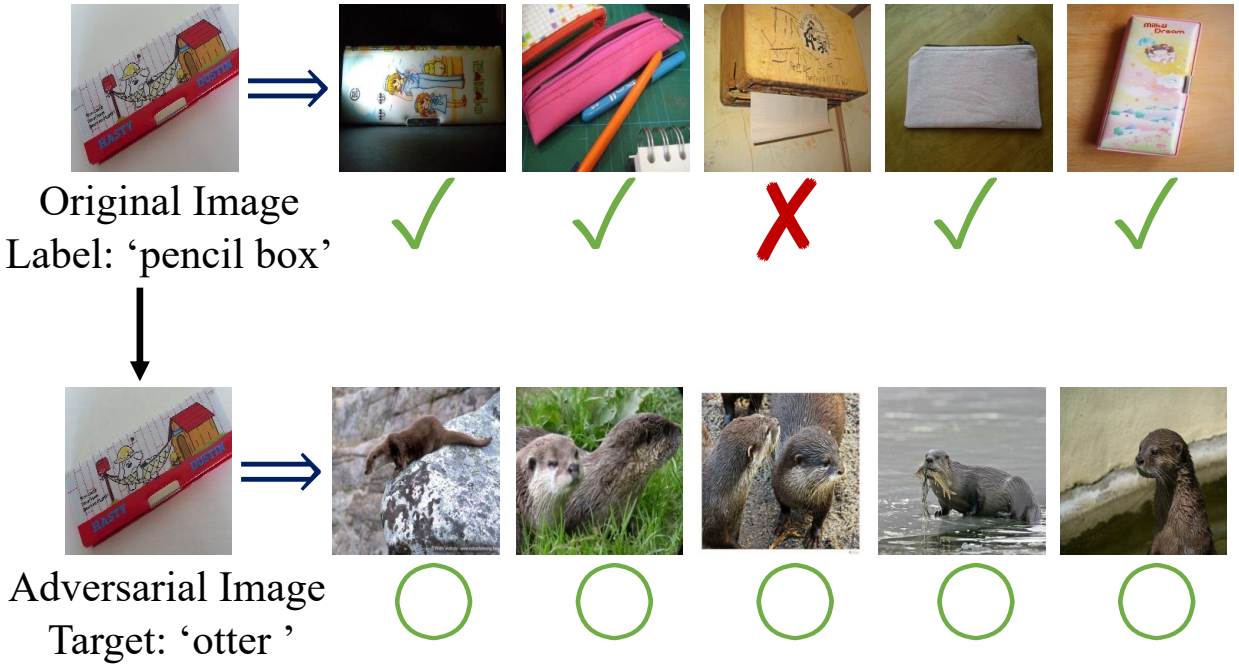
Feasible on digital-world and physical-world

Non-targeted v. s. Targeted

Non-targeted

Making the labels merely differ from original

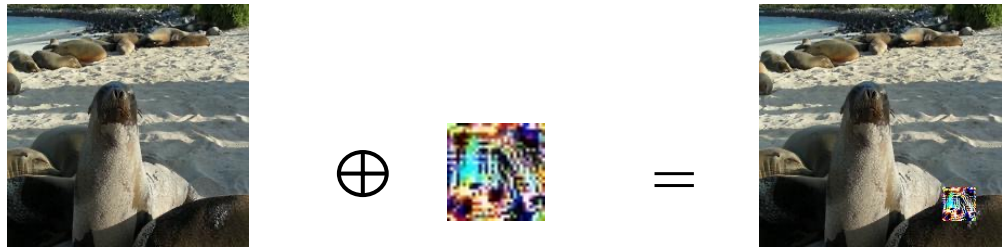
Targeted



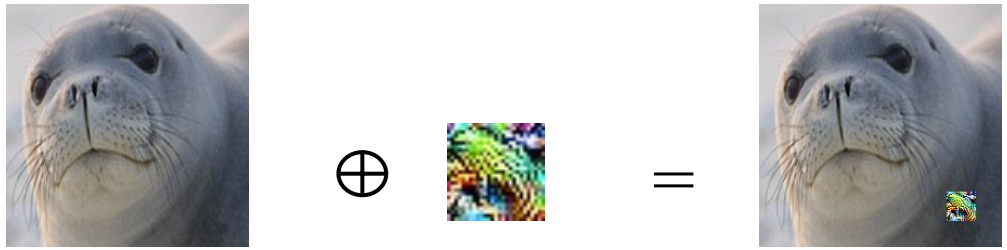
Making the labels all are target one

Image-specific v. s. Universal

Image-specific



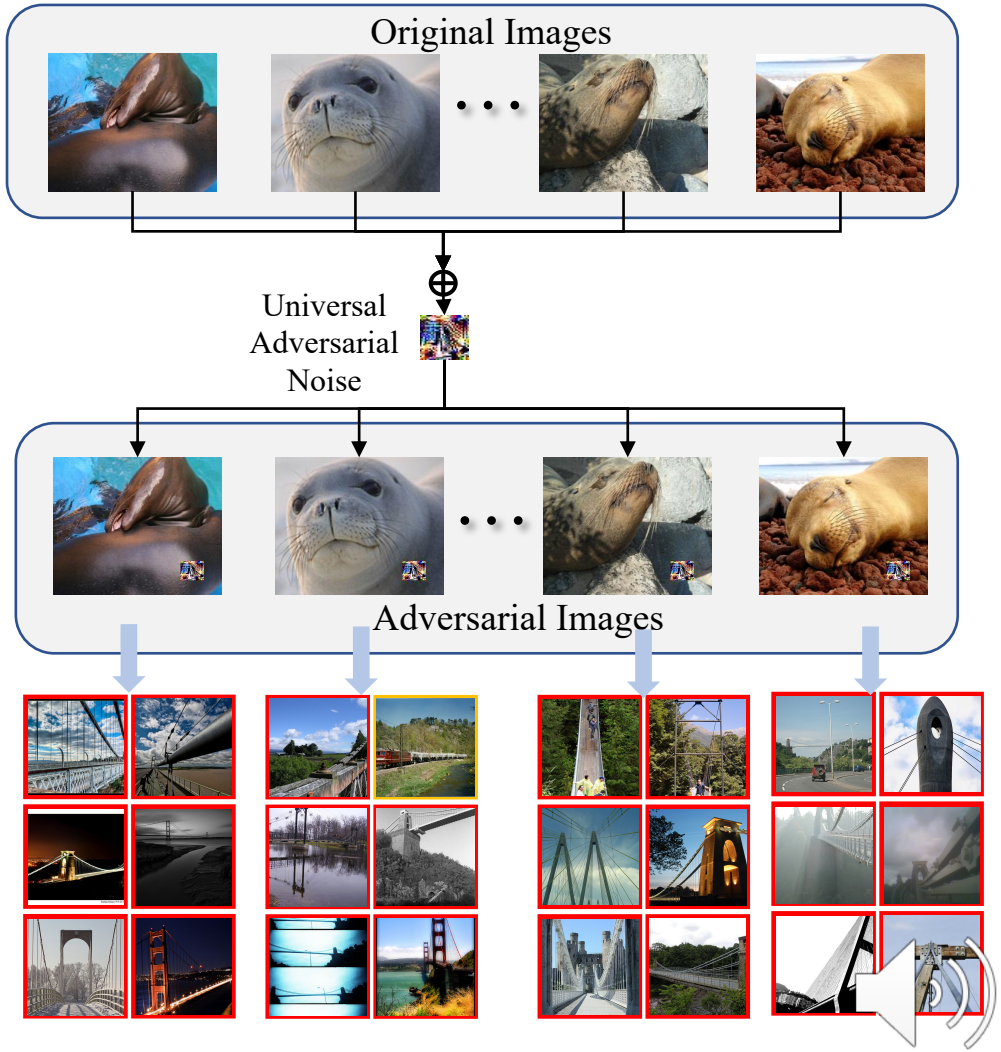
Original Image 1 ⊕ Noise 1 = Adversarial Image 1



Original Image 2 ⊕ Noise 2 = Adversarial Image 2

Each noise only applicable for one image only.

Universal



Motivation

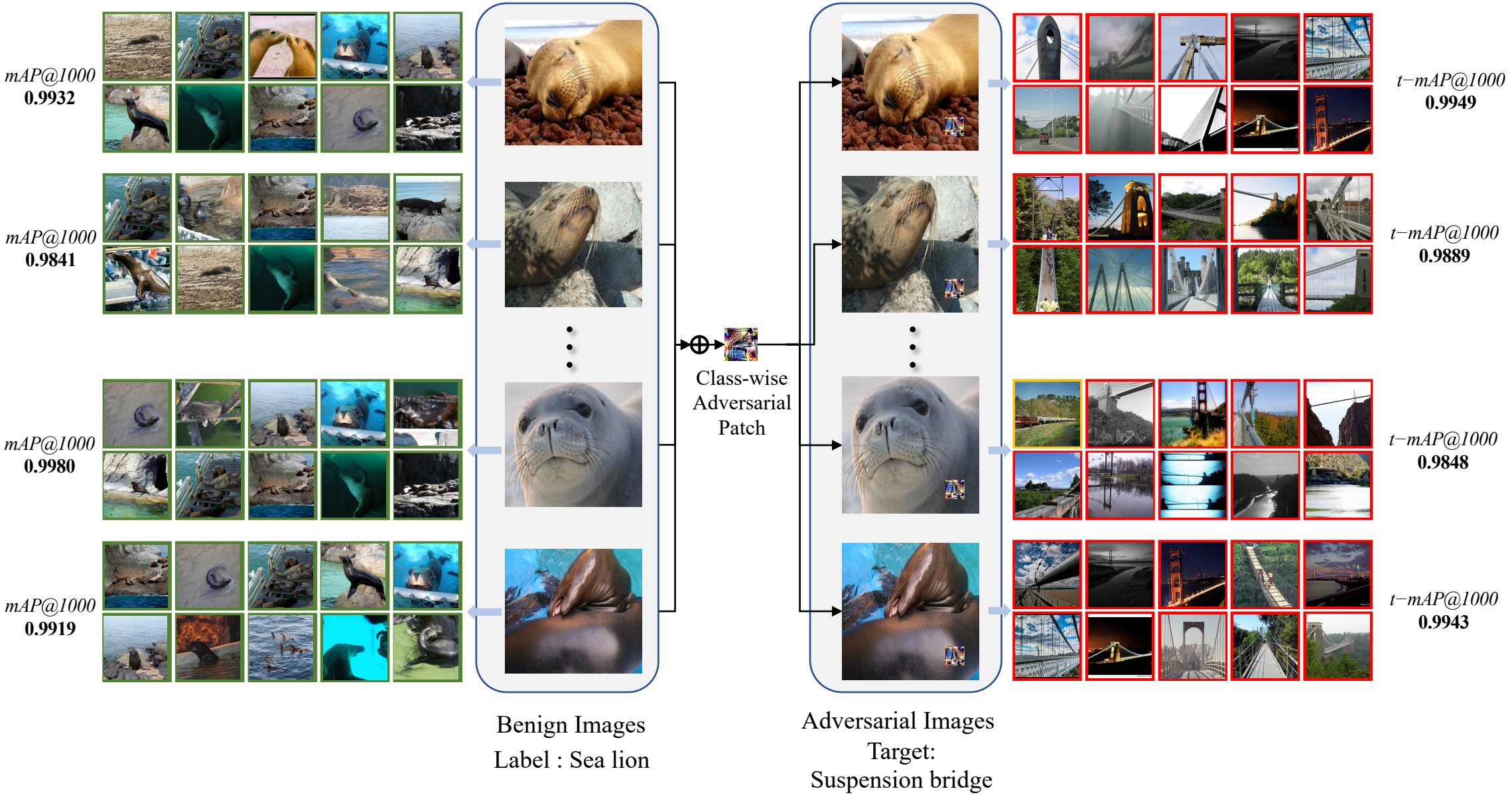
Attack Goals

- Goal 1: Exploring the **Patch**-based Attack on Deep Hashing
  - Making the adversarial noise confined to a localized area of image.
- Goal 2 : Realizing the **Targeted** Attack
  - Making the top-K retrieved results all belong to another predefined class.
- Goal 3 : Improving the **Generalization** of Adversarial Noise
  - Making the single adversarial noise applicable for an entire set of images.

Major Motivation

- Achieving the SOTA attack strength while keeping the adversarial noise universally applicable

Motivation


mAP@1000
0.9932
mAP@1000
0.9841
mAP@1000
0.9980
mAP@1000
0.9919
Class-wise
Adversarial
Patch
t−mAP@1000
0.9949
t−mAP@1000
0.9889
t−mAP@1000
0.9848
t−mAP@1000
0.9943
Benign Images
Label : Sea lion
Adversarial Images
Target:
Suspension bridge

Problem Formulation

Optimization problem to generate adversarial patch δ:

$$\min_{\delta} d(F(X'), F(X^t))$$

where $X' = \{x'_q | x'_q = x_q \oplus \delta, x_q \in X^s\}$, $F(\cdot)$ returns the hash code.

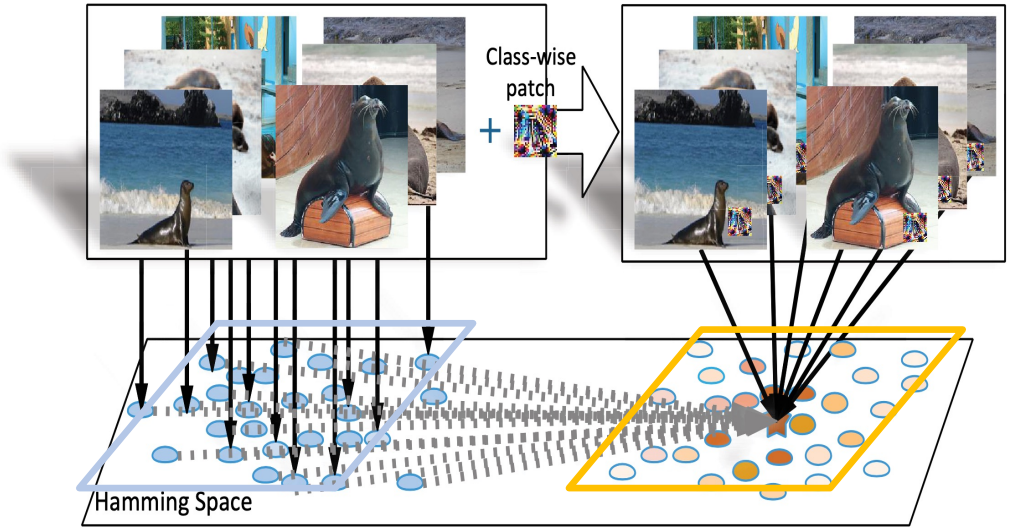
source set X^s contains images with source label.

target set X^t contains images with target label.

This optimization is a set-to-set distance minimization

- Difficult to optimize directly.
- High complexity.

Problem Simplification



The effect of Metric Learning:

- Maximizing the distance for different label
- Minimizing the distance for same label

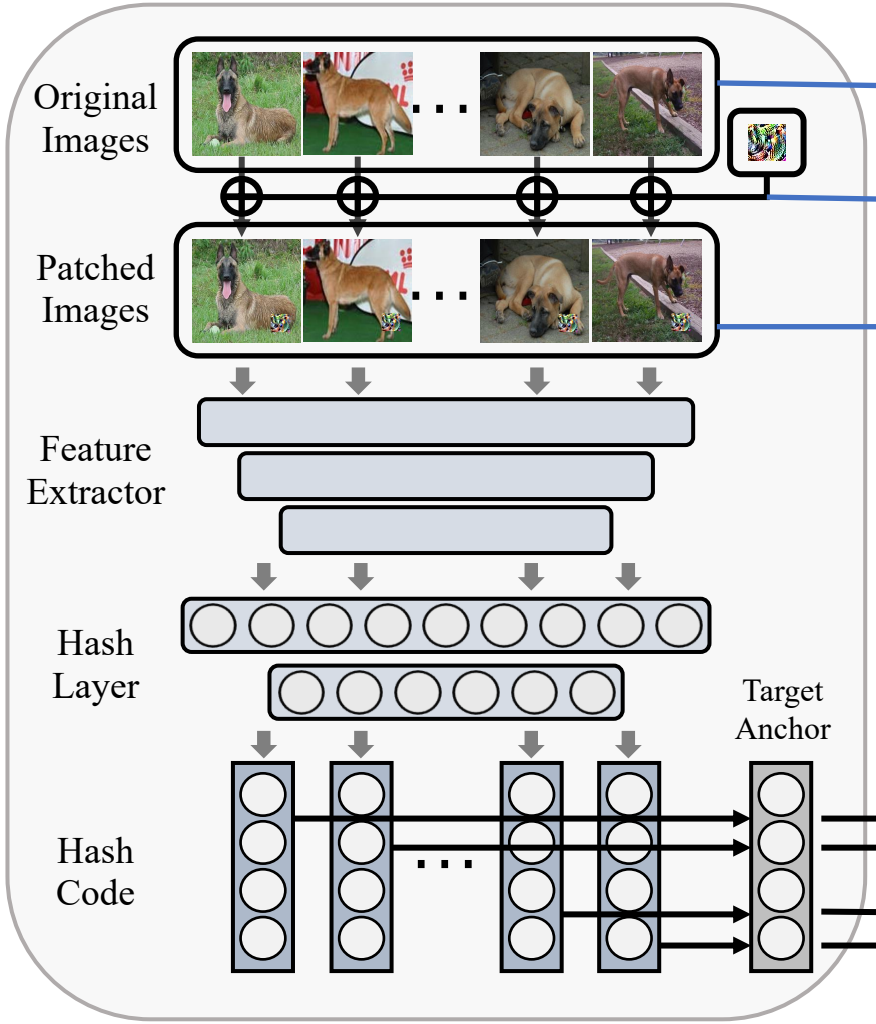
The hamming space cluster

Obtain an anchor $\boldsymbol{h}_t$ of target set

$$\boldsymbol{h}_t = sign(\sum_{j=1}^{n} \boldsymbol{c}_j)$$

$$\min_{\delta} d(F(X'), F(X^t)) \quad \text{reduce to} \quad \min_{\delta} d(F(X'), \boldsymbol{h}_t)$$

Gradient Aggregation

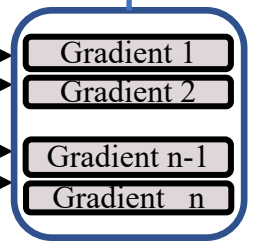


A mini-batch of different original images X_b

Pasted with the same patch δ

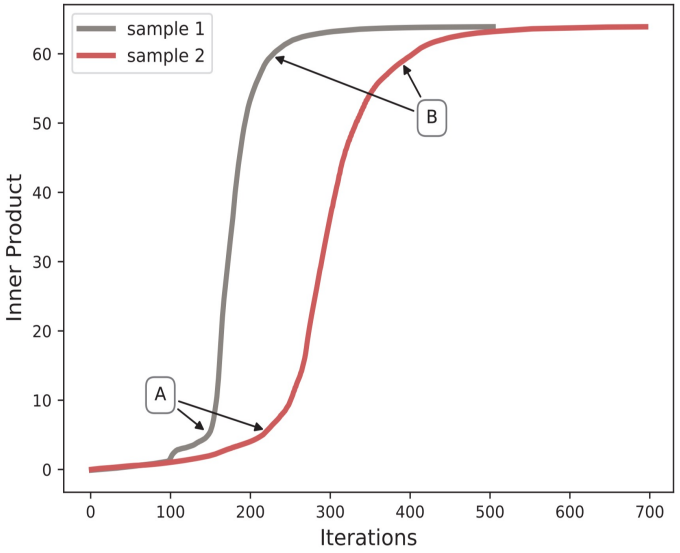
A mini-batch of different patched images X_b'

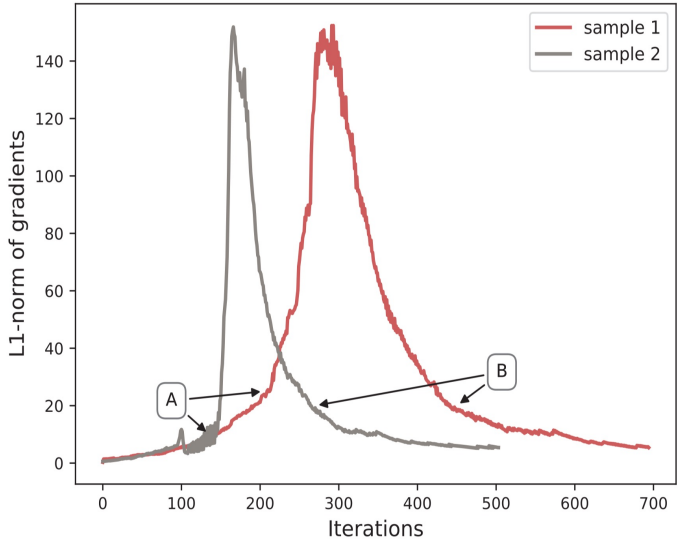
A mini-batch of different gradients



Find a way to aggregate these gradients

Gradient Aggregation





Conclusions:

- Different samples take different efforts to find the "*right*" direction to leave original cluster
- The earlier the sample leaves Stage 1, the faster it will converge to the target cluster

We divide the general optimization process into three stages:

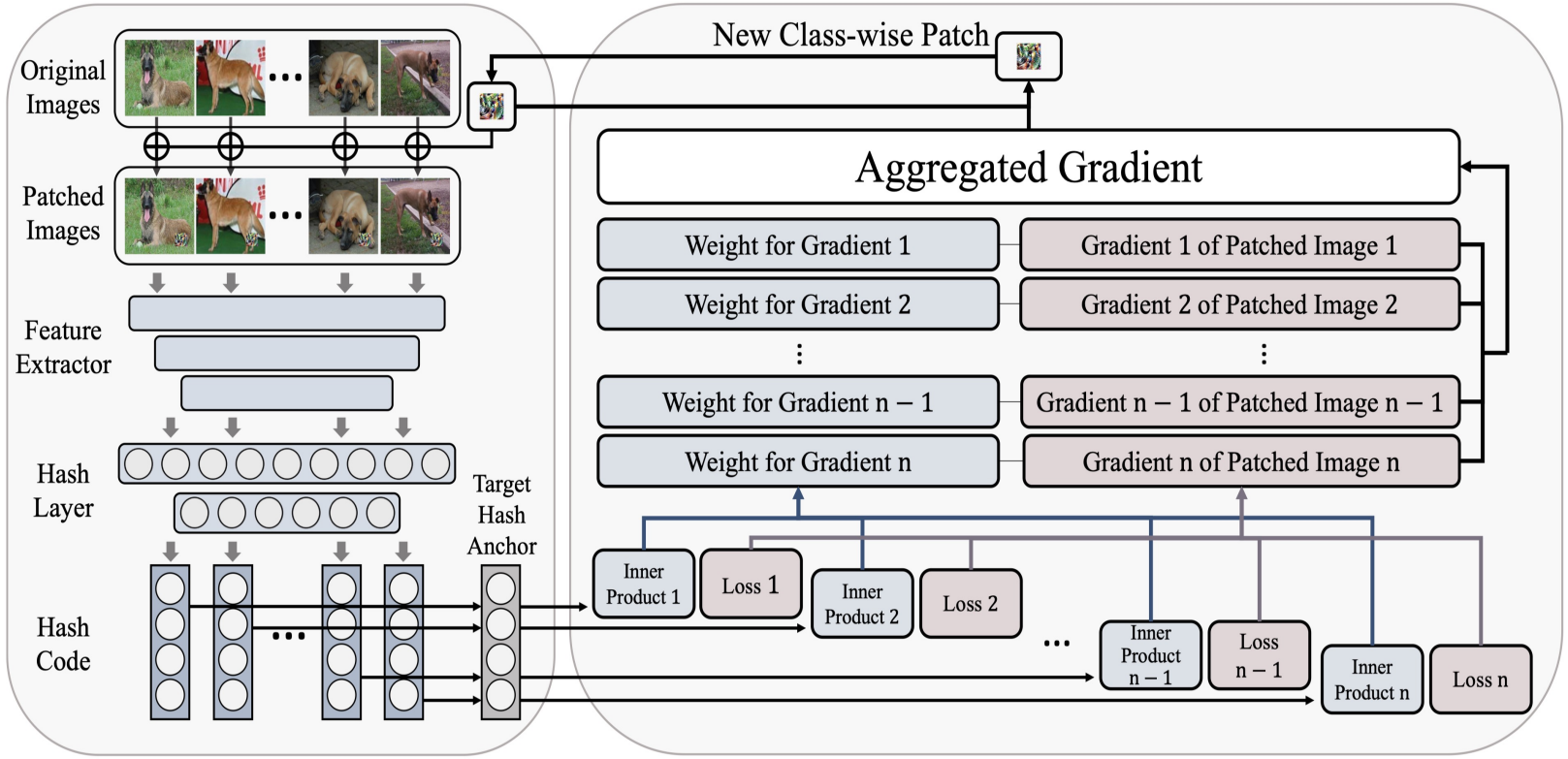
- Stage 1 (before point A): leaving the original cluster
- Stage 2 (between A and B): moving to the target cluster
- Stage 3 (after point B): approaching target anchor

Assumption: the stage 2 is crucial to the optimization, the gradients from stage 2 are more "accurate"

Idea: let the gradients influenced more by Stage 2

Gradient Aggregation

Product-based Weighted Gradient Aggregation



Calculate the inner product as weights:

$$P_u = \boldsymbol{c}_u \cdot \boldsymbol{h}_t$$

Use hyper-parameters to keep the gradients influenced more by stage 2:

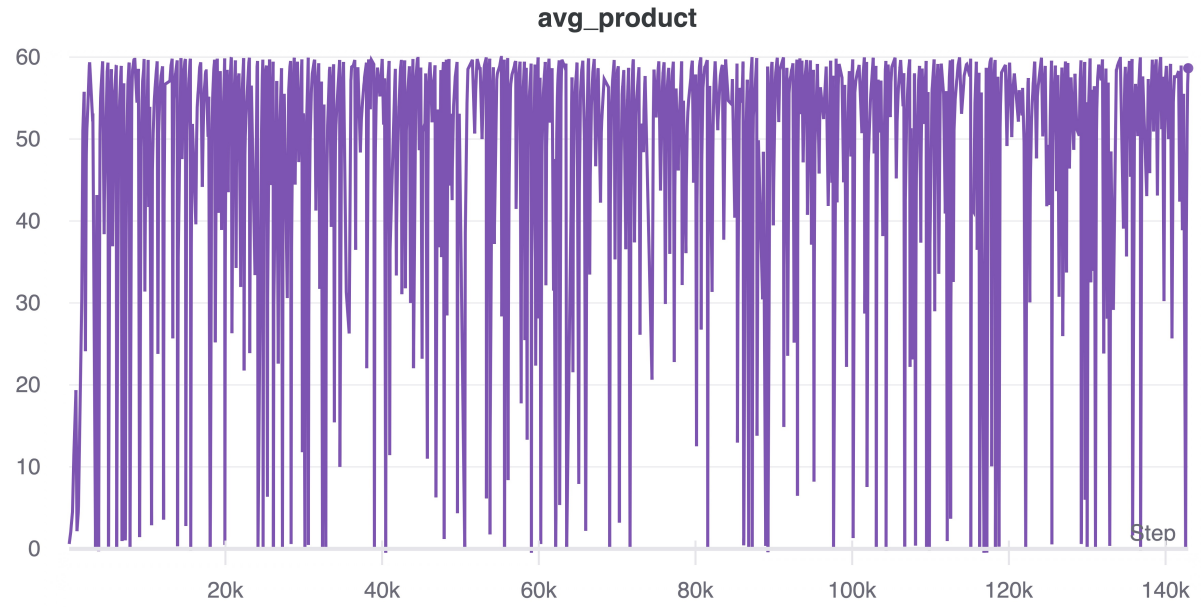
$$\bar{\nabla} = \frac{\sum_{u=1}^{U} W_u \times \nabla \mathcal{L}_u}{\sum_{u=1}^{U} W_u}$$

$$W_u = \begin{cases} P_u + \beta, & \text{if } P_u \leq T \\ P_u - \gamma, & \text{otherwise} \end{cases}$$

Patch Averaging

The Overfitting Problem

The patch generated from a single mini-batch X_b' can achieve high inner product for all the samples in X_b', however, have relatively low applicability for unseen mini-batches.



Reason:
The patch gets updated too thoroughly.

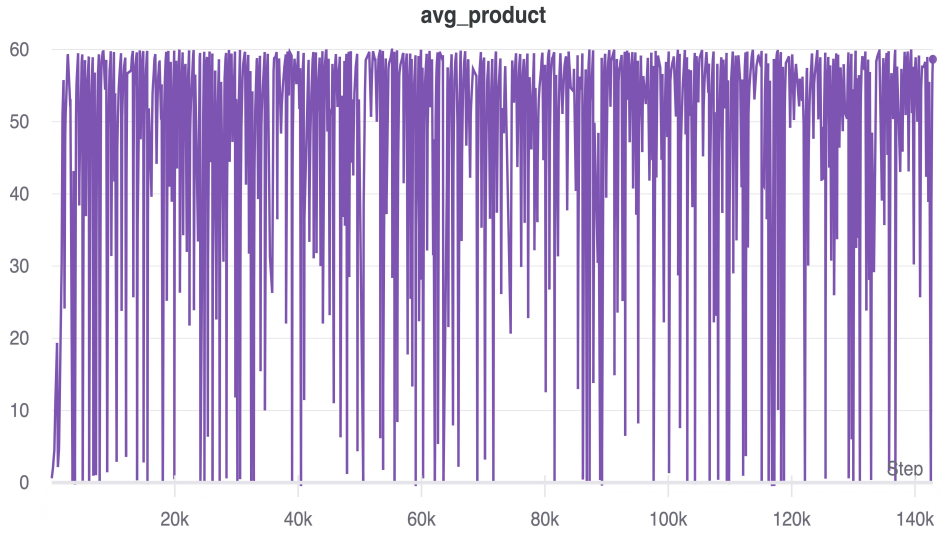
General Remedy:
Use a threshold condition to break the training of current mini-batch, thus enforce generalization. However, this makes the attack inefficient.

Patch Averaging

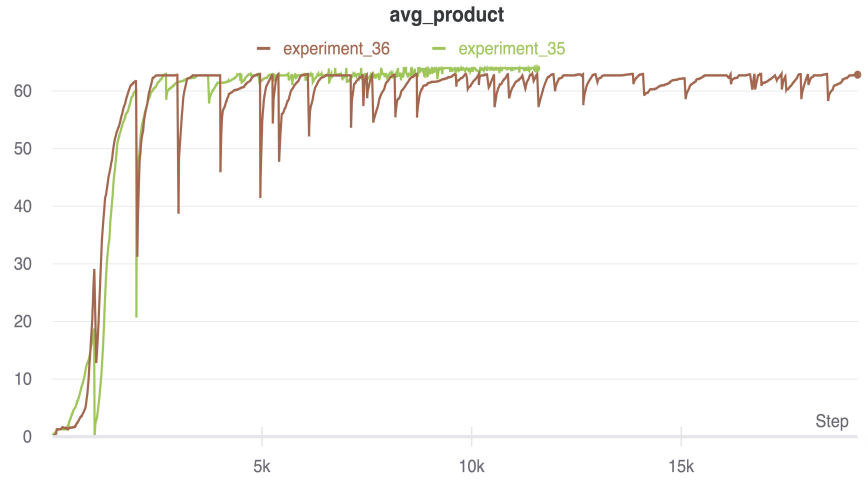
Our solution

Averaging a few patches trained from the previous mini-batches as the initial patch for current mini-batch X'_b.

$$\delta_b^0 = \frac{1}{p}(\delta_{b-p} + \delta_{b-p+1} + \ldots + \delta_{b-1})$$



Use Our Method



Evaluation Setup

Dataset: ImageNet and NUS-WIDE

Target model :

- CNN backbone: ResNet50 and VGG16
- Deep Hashing Method: CSQ and HashNet
- Hash bit : 64bit and 32bit

Mapping: We choose **10** source class to target class mappings for above scenarios.

Query number: Each adversarial patch is paste on 500 unseen images for test.

Evaluation metrics:

- org-mAP(O): take clean images as input and original label as referenced label;
- t-mAP (T): take clean images as input and target label as the referenced label;
- adv-org-mAP (AO): take perturbed images as input and original label as the referenced label;
- adv-t-mAP (AT): take perturbed images as input and target label as the referenced label;

Overall Attack Performance

Table 3: Targeted attack performance for each mapping on CSQ [44] and HashNet [6]

Dataset	Mapping	CSQ															
		ResNet50								VGG16							
		32 bits				64 bits				32 bits				64 bits			
		O	T	AO	AT	O	T	AO	AT	O	T	AO	AT	O	T	AO	AT
ImageNet	$\mathcal{M}_1$	98.57	0.00	2.06	93.21	98.75	0.00	10.17	88.99	97.97	0.00	1.99	92.34	98.57	0.00	2.06	93.21
	$\mathcal{M}_2$	92.85	0.00	0.35	98.34	91.16	0.00	7.85	91.38	89.18	0.01	2.05	95.76	90.98	0.00	0.79	98.76
	$\mathcal{M}_3$	92.60	0.06	0.98	99.04	95.06	0.27	4.06	95.39	90.51	0.27	0.38	98.57	92.42	0.20	0.19	98.77
	$\mathcal{M}_4$	92.51	0.00	0.00	99.33	94.74	0.00	4.11	95.97	89.87	0.00	0.16	97.61	91.40	0.00	0.00	98.02
	$\mathcal{M}_5$	87.46	0.00	0.85	99.04	88.06	0.01	2.55	96.90	78.66	0.01	18.94	60.60	81.70	0.00	0.55	99.13
	$\mathcal{M}_6$	84.26	0.05	0.96	97.22	86.74	0.05	11.03	90.40	78.72	0.04	0.21	98.81	82.70	0.02	0.56	98.96
	$\mathcal{M}_7$	94.44	0.20	2.76	95.56	95.98	0.00	9.86	89.79	90.24	0.00	0.53	99.01	91.05	0.00	1.37	97.74
	$\mathcal{M}_8$	91.24	0.00	0.39	99.36	90.39	0.00	6.72	93.18	89.57	0.16	5.73	50.12	88.00	0.16	23.72	5.57
	$\mathcal{M}_9$	95.28	0.00	0.22	95.04	97.31	0.00	9.34	87.21	96.11	0.00	0.39	97.40	96.68	0.00	0.98	95.32
	$\mathcal{M}_{10}$	94.23	0.00	0.00	99.66	95.10	0.00	0.00	99.43	91.77	0.01	0.00	96.71	93.10	0.00	0.19	98.29
	AVG	**92.34**	**0.03**	**0.86**	**97.58**	**93.33**	**0.03**	**6.57**	**92.86**	**89.26**	**0.05**	**3.04**	**88.69**	**90.66**	**0.04**	**3.04**	**88.38**
NUS-WIDE	$\mathcal{M}_{11}$	42.08	20.58	3.29	88.40	41.48	19.24	3.03	87.67	39.49	18.27	2.71	81.13	40.20	17.63	1.77	89.35
	$\mathcal{M}_{12}$	77.18	8.89	11.14	92.19	77.73	8.09	3.79	97.67	76.15	8.62	3.07	98.37	73.76	8.99	2.12	98.32
	$\mathcal{M}_{13}$	92.65	12.86	52.58	77.45	93.10	11.52	49.28	77.97	90.54	13.68	21.22	74.19	92.04	12.98	15.61	76.22
	$\mathcal{M}_{14}$	96.50	2.61	6.50	98.72	97.13	2.52	7.25	99.18	96.18	2.95	9.91	98.18	96.83	3.09	7.57	98.10
	$\mathcal{M}_{15}$	98.85	0.23	37.95	62.17	99.14	0.28	20.53	81.50	98.88	0.28	16.10	76.82	99.19	0.25	15.69	73.76
	AVG	**81.45**	**9.03**	**22.29**	**83.79**	**81.72**	**8.33**	**16.78**	**88.80**	**80.25**	**8.76**	**10.60**	**85.74**	**80.40**	**8.59**	**8.55**	**87.15**
Dataset	Mapping	HashNet															
ImageNet	$\mathcal{M}_1$	97.36	0.00	0.27	70.27	48.89	0.00	0.20	64.18	94.64	0.07	0.07	74.87	98.07	0.00	0.12	36.02
	$\mathcal{M}_2$	90.03	0.00	1.44	76.41	47.69	0.00	0.66	97.82	85.72	0.00	2.08	93.00	86.03	0.00	0.19	99.24
	$\mathcal{M}_3$	93.43	0.22	1.38	97.77	92.20	0.21	1.54	92.15	82.60	0.26	1.12	98.35	91.58	0.01	0.18	99.40
	$\mathcal{M}_4$	92.35	0.00	0.00	99.65	86.25	0.00	0.00	99.54	93.01	0.00	0.00	97.87	89.48	0.00	0.00	99.49
	$\mathcal{M}_5$	80.11	0.03	1.54	84.25	83.19	0.00	0.49	60.53	67.53	0.23	0.19	96.66	80.91	0.04	0.00	99.48
	$\mathcal{M}_6$	79.19	0.23	8.49	66.97	86.43	0.00	26.84	43.97	38.09	0.03	0.32	94.57	72.57	0.02	0.01	99.33
	$\mathcal{M}_7$	95.59	0.00	4.73	94.87	96.46	0.00	3.24	80.97	87.15	0.01	0.19	98.07	93.33	0.00	0.19	99.57
	$\mathcal{M}_8$	74.33	0.00	0.00	87.73	76.31	0.00	0.00	99.05	86.85	0.04	0.66	95.78	87.66	0.00	0.00	99.89
	$\mathcal{M}_9$	25.76	0.01	0.23	74.16	86.54	0.00	0.98	45.71	73.12	0.00	1.25	32.80	96.12	0.00	0.00	19.11
	$\mathcal{M}_{10}$	47.08	0.01	0.00	71.74	65.06	0.00	0.00	40.87	93.14	0.00	0.00	97.26	95.02	0.01	0.00	99.55
	AVG	**77.52**	**0.05**	**1.81**	**82.38**	**76.90**	**0.02**	**3.40**	**72.48**	**80.19**	**0.06**	**0.59**	**87.92**	**89.08**	**0.01**	**0.07**	**85.11**
NUS-WIDE	$\mathcal{M}_{11}$	39.41	15.06	3.53	84.70	38.93	14.59	4.24	83.25	40.06	15.57	1.10	82.95	39.40	15.38	0.86	87.92
	$\mathcal{M}_{12}$	68.16	9.37	4.75	96.09	76.46	8.08	2.41	98.04	68.37	10.40	2.34	98.45	69.82	7.55	2.49	99.34
	$\mathcal{M}_{13}$	89.40	16.08	37.41	80.95	91.04	15.23	32.19	86.02	90.14	16.12	15.88	82.84	93.39	13.89	11.95	86.15
	$\mathcal{M}_{14}$	95.41	4.65	14.21	98.76	97.08	2.72	15.62	98.87	95.65	3.90	7.33	97.89	97.65	3.01	5.61	99.13
	$\mathcal{M}_{15}$	98.64	0.16	23.78	69.17	99.23	0.28	40.01	64.34	98.07	0.21	9.94	77.14	99.43	0.17	6.94	87.38
	AVG	**78.20**	**9.06**	**16.74**	**85.93**	**80.55**	**8.18**	**18.89**	**86.10**	**78.46**	**9.24**	**7.32**	**87.85**	**79.94**	**8.00**	**5.57**	**91.98**

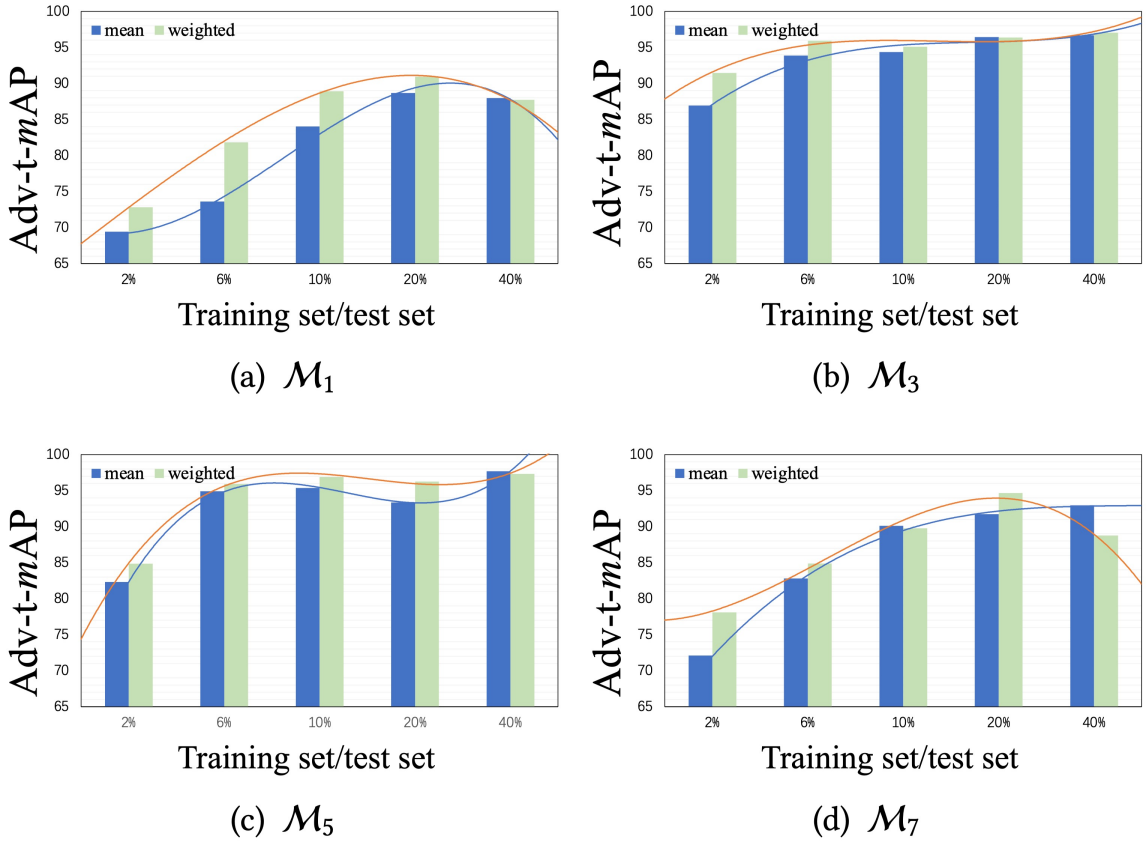
The sharp mAP drops (from O to AO) means the patched images successfully leave the original clusters.

The sharp t-mAP rises (from T to AT) means the patched images successfully enter the target clusters.

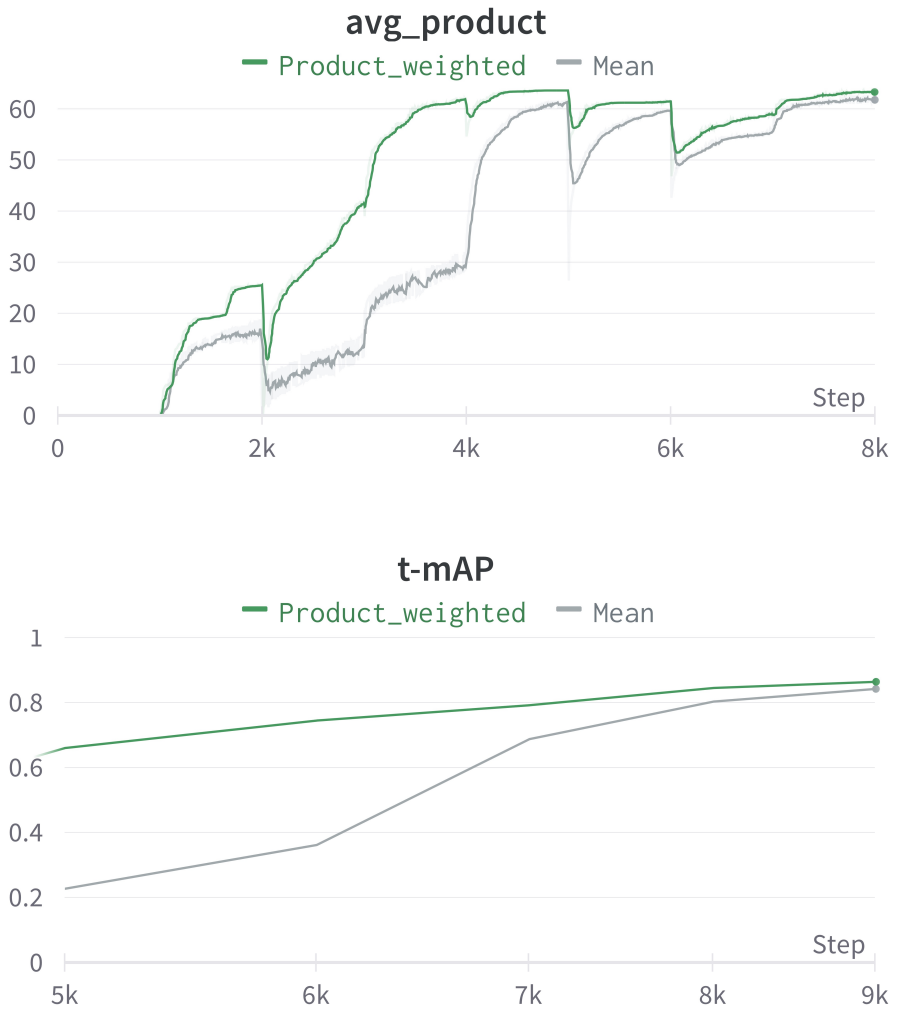
Among the 120 attack settings, 60.90% of them can achieve impressive attack performance with AT above 90%, nearly all of them have an AT above 60%.

Attack is strong, generalization is achieved.

Ablation Study: Weighted V.S Mean



Efficiency



Comparison Study: AdvHash V.S DHTA

Table 4: Comparison with DHTA [2]

Mapping	Samples	CSQ				HashNet			
		ResNet50		VGG16		ResNet50		VGG16	
		32	64	32	64	32	64	32	64
$\mathcal{M}_4$	DHTA	99.49	99.77	80.94	36.82	99.68	99.25	6.36	36.82
	AdvHash-train	98.33	89.3	97.8	34.97	99.63	95.85	95.86	99.5
	AdvHash-test	97.14	94.28	97.21	29.72	99.04	95.61	97.42	99.5
$\mathcal{M}_6$	DHTA	97.73	99.29	78.71	76.36	92.87	84.09	96.16	97.55
	AdvHash-train	97.79	81.3	99.03	99.54	43.38	31.06	94.62	99.6
	AdvHash-test	95.87	83.27	98.64	99.15	33.15	36.56	92.64	98.73
$\mathcal{M}_{10}$	DHTA	99.66	99.44	82.5	54.68	69.06	28.94	97.2	99.58
	AdvHash-train	97.66	97.41	80.59	98.46	71.32	69.99	97.06	99.59
	AdvHash-test	98.67	99.38	79.94	97.33	70.6	70.8	96.85	99.57

DHTA is the current SOTA targeted attack

DHTA is an image-specific attack method, which has not considered generalization.

The adversarial noise generated by AdvHash generalizes well and still can compete or even outperform DHTA.

Conclusions

- We propose the first targeted attack with strong generalization on image retrieval system, which is also the first patch-based attack on deep hashing.

- We convert the set-to-set problem to a set-to-point optimization, and we propose a gradient aggregation strategy and a patch averaging method to better solve this problem.

- We conduct extensive experiments considering different CNN backbones, different deep hashing methods, and different hash bits to evaluate the attack performance.