

# OPTIMAL TRANSPORT BASED ADVERSARIAL PATCH TO LEVERAGE LARGE SCALE ATTACK TRANSFERABILITY

**Anonymous authors**

Paper under double-blind review

## CONTENTS

<b>A</b>	<b>Implementation details</b>	<b>1</b>
<b>B</b>	<b>Feature point method instability</b>	<b>2</b>
<b>C</b>	<b>Feature point method generalization</b>	<b>6</b>
<b>D</b>	<b>Benefits of Optimal Transport</b>	<b>6</b>
<b>E</b>	<b>Model robustness and patch position</b>	<b>7</b>
<b>F</b>	<b>Ensemble methods</b>	<b>8</b>
<b>G</b>	<b>Transferability on adversarially trained models</b>	<b>8</b>
<b>H</b>	<b>Robustness according to physical transformations</b>	<b>9</b>
<b>I</b>	<b>Ablation studies</b>	<b>10</b>
<b>J</b>	<b>Decision boundary-based methods overfitting</b>	<b>11</b>
<b>K</b>	<b>Complementary tables</b>	<b>14</b>
<b>L</b>	<b>Printable patches</b>	<b>15</b>

## A IMPLEMENTATION DETAILS

Our method training routine uses the PyTorch library (Paszke et al., 2019). For the training of each patch on medium and large models we consider a single NVIDIA V100-32G or a single NVIDIA A100 respectively. To train patches on smaller NVIDIA cards we should reduce the batch size.

When not specified, patches are designed to target one the following classes: salamander, starfish, bird house, bullfrog, pinwheel, mongoose, brown bear, accordion and common iguana.

We use Expectation over Transformations (EoT (Eykholt et al., 2018)) to obtain a more physically realizable patch, similarly to prior work on APAs (Brown et al., 2017; Lee & Kolter, 2019; Casper et al., 2022). For all the methods (GAP (Brown et al., 2017), LaVAN (Karmon et al., 2018), L2 (Inkawhich et al., 2019) and ours), during training, we randomly rotate the patch up to five degrees for the x and y-axis and up to 10 degrees for the z-axis. We also randomly scale the patch between  $70 \times 70$  to  $110 \times 110$  pixels, adjust patch brightness between  $[-0.1, 0.1]$  and patch blur between  $[0.8, 1.2]$ , and apply normal noise of magnitude 0.1 on the patch. Patches are randomly translated in the image but not in the center.

To control the balance between the adversarial loss and the total variation loss, the gradient of each

loss is computed individually, normalized, and combined using a weighted sum. Following Nesti et al. (2022) we choose  $w_{adv} = 1$  and  $w_{TV} = 0.1$  where  $w_{adv}$  is the weight for the adversarial loss and  $w_{TV}$  is the weight for the TV loss.

**Computation time.** We measure and report the computation time of each method in Table 1. This Table reports the averaged computational time for the different methods. Our method has a similar computational time as other methods. This result may be counterintuitive as OT losses are known to be slow, but in our setting, the number of samples is low. The M3D method (Zhao et al., 2023) is much slower than other methods. It is coherent, this method trains alternatively the patch and two models in a min-max game.

Table 1: Computational time of the different methods to obtain a fully optimized patch (minutes). Times are averaged over ten optimization runs. Each run is launched on the same setup composed by a single NVIDIA A100.

Method	Time
GAP (Brown et al., 2017)	20
LaVAN (Karmon et al., 2018)	30
L2 (Inkawhich et al., 2019)	20
TnT (Doan et al., 2022)	30
Casper et al. (2022)	35
TTP (Naseer et al., 2021)	30
M3D (Zhao et al., 2023)	66
<b>Ours</b> ( $\mathbf{SW}_2^{(1)}_{500}$ )	19
<b>Ours</b> ( $\mathbf{W}_2^{(1)}$ )	20

## B FEATURE POINT METHOD INSTABILITY

To measure the stability of the L2 method (Inkawhich et al., 2019), we launch the optimization for three randomly selected target points. Patches are designed to sway ResNet50-v1 or Swin-T to output the class Australian terrier. Figure 1 plots the learning curves and the resulting patches for our distribution-based approach for Resnet50-v1 and Swin-T, respectively. Figure 2 and 3 plot the learning curves and the resulted patches of the L2 method for Resnet50-v1 and Swin-T, respectively. These four graphs show that our method is the easiest to optimize and is more robust to optimization artifacts. For the Swin-T model, the optimization for the L2 method becomes noisy. Table 2 reports the transfer results of the obtained patches from previous figures. Although the optimization has converged for the first target of the L2 method for ResNet50-v1, the obtained patch is harmless. Even if the APA works, its attacking capacity depends on the considered target point. For example, the mean transferability on Swin-T can be decreased by a factor four. In general, our distribution-oriented approach outperforms the L2 method.

Table 2: Transfer results between categories of models (tSuc (%)) for the L2 method and for our distribution-oriented method. Three different target points are evaluated for the L2 method. Results are for the source model ResNet50-V1 and Swin-T, for the class Australian terrier and for patches of size  $60 \times 60$ . Patches are placed randomly in the image but not at the center of images.

Source	Method		Target							mean / std
			CNNs-v1	CNNs-v2	ENet	CNext	DeiT	Swin	AT	
ResNet50-v1	L2 (Inkawhich et al., 2019)	Target 1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1 / 0
		Target 2	36.64	2.52	9.35	0.52	3.59	0.5	3.71	8.12 / 12
		Target 3	43.83	4.18	8.82	0.75	4.98	0.58	6.09	9.89 / 14.1
	<b>Ours</b>		43.34	4.76	8.75	0.92	6.46	0.63	4.68	<b>9.94 / 13.9</b>
Swin-T	L2 (Inkawhich et al., 2019)	Target 1	4.12	1.18	2.41	0.23	1.83	1.9	0.39	1.72 / 7.8
		Target 2	26.97	7.36	4.65	3.9	7.2	6.13	1.92	8.3 / 7.8
		Target 3	0.17	0.11	0.12	0.1	0.1	0.07	0.1	0.11 / 0.02
	<b>Ours</b>		50.77	12.54	14.2	7.08	13.64	8.19	5.94	<b>16.05 / 14.5</b>

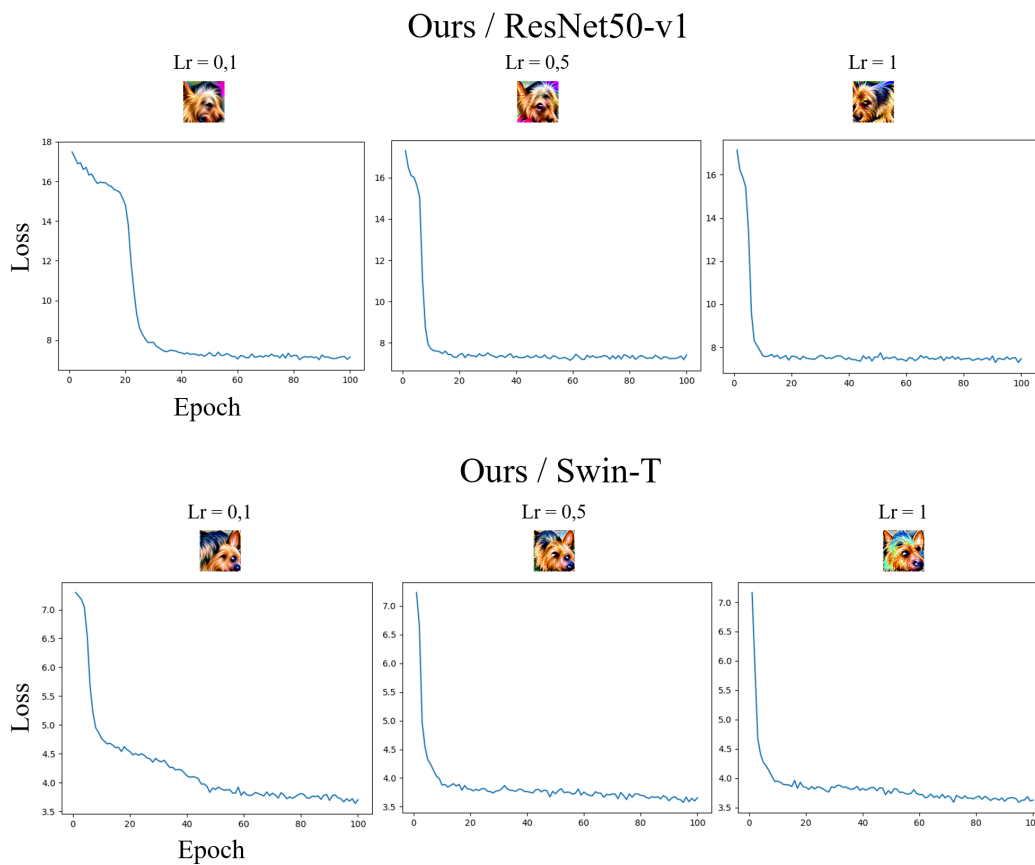


Figure 1: Learning curves and resulted patches of our distribution-oriented method. The optimization is run for three different learning rate. The source model is ResNet50-v1 or Swin-T and the targeted class is Australian terrier.

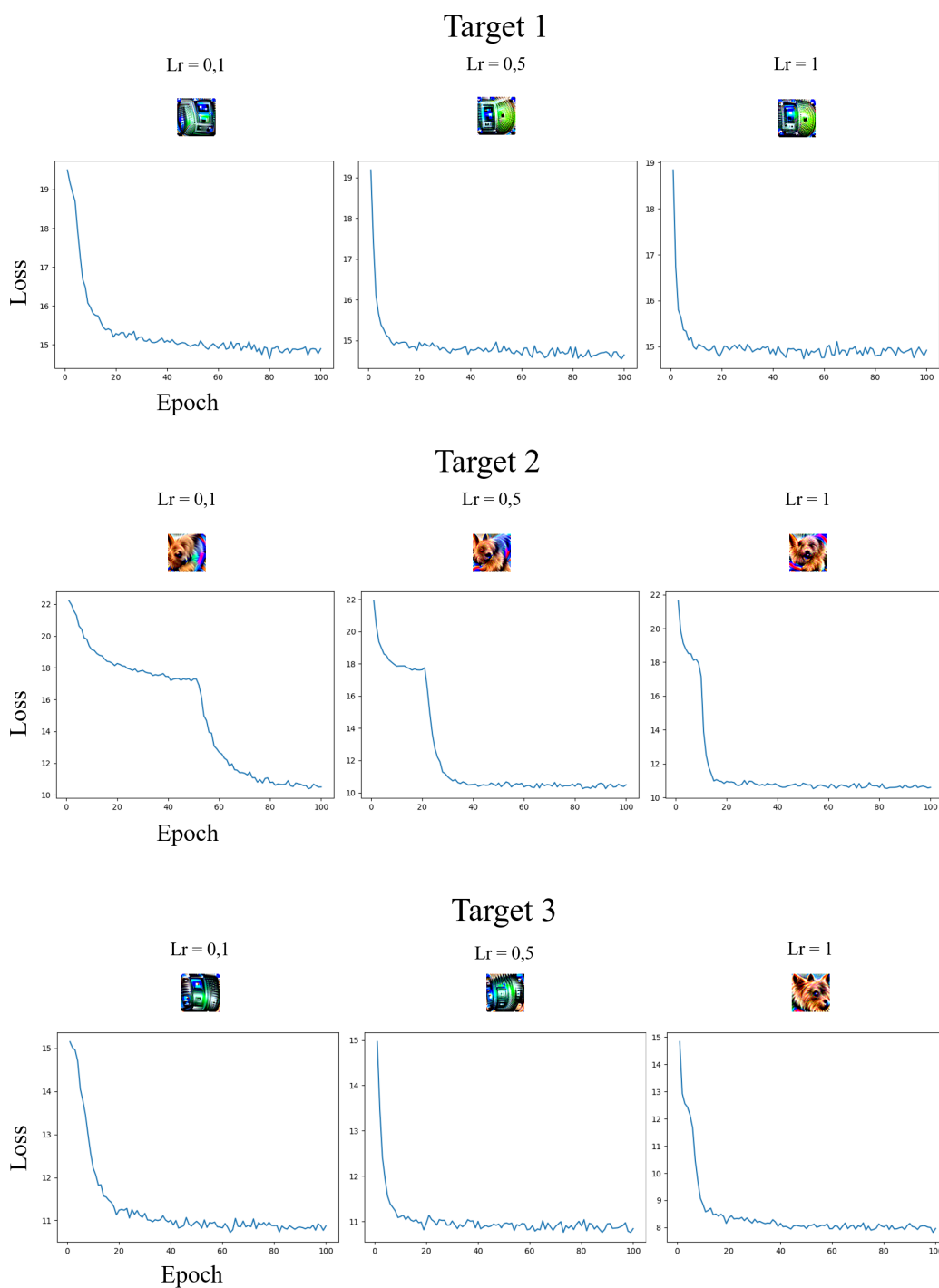


Figure 2: Learning curves and resulted patches of the L2 method for different targeted points. For each targeted point the optimization is run for three different learning rate. The source model is ResNet50-v1 and the targeted class is Australian terrier.

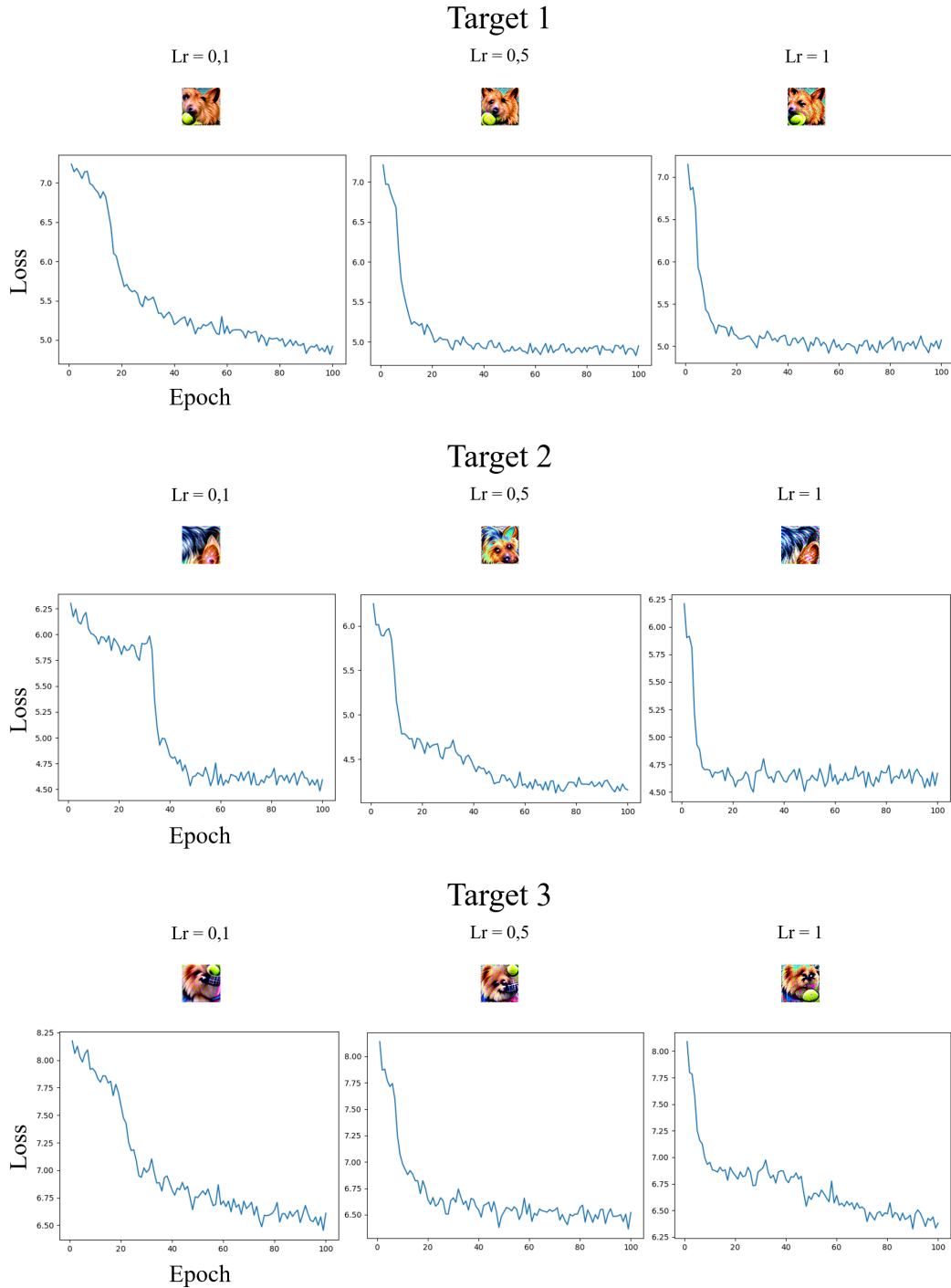


Figure 3: Learning curves and resulted patches of the L2 method for different targeted points. For each targeted point the optimization is run for three different learning rate. The source model is Swin-T and the targeted class is Australian terrier.

## C FEATURE POINT METHOD GENERALIZATION

We provide in this section a proof that the exact 2-Wasserstein distance coincide with the L2-based method Inkawhich et al. (2019) when the source distribution is uniformly distributed and the targeted distribution is supported by a unique point. We recall that

$$\mathbf{W}_2^2(\mu, \nu) = \inf_{\pi \in \Pi(\mu, \nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\|^2 d\pi(x, y), \quad (1)$$

defines the 2-Wasserstein distance. This distance can be interpreted through a probabilistic point of view. If we name  $(X, Y)$  a couple of random variables over  $\mathbb{R}^d \times \mathbb{R}^d$  with  $X \sim \mu$ ,  $Y \sim \nu$  and  $(X, Y) \sim \pi \in \Pi(\mu, \nu)$ , we can write

$$\mathbf{W}_2^2(\mu, \nu) = \min_{(X, Y)} \mathbb{E}_{(X, Y)} [\|X - Y\|^2]. \quad (2)$$

If we suppose that the target distribution is composed by a unique point, *i.e.*,  $\nu = \delta_y$ , then we have

$$\mathbf{W}_2^2(\mu, \nu) = \min_X \mathbb{E}_X [\|X - y\|^2]. \quad (3)$$

In our problem we have empirical distribution based on samples, we name  $\hat{\mu}_n$  and  $\hat{\nu}_m$  the empirical distributions of  $\mu$  based on  $n$  samples and  $\nu$  based on  $m$  samples respectively. We suppose that each sample from each distribution is uniformly distributed, *i.e.*,  $\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n \delta_{x_i}$  and  $\hat{\nu}_m = \frac{1}{m} \sum_{j=1}^m \delta_{y_j}$ , where  $\delta$  is the Kronecker symbol. The estimated 2-Wasserstein distance is

$$\mathbf{W}_2^2(\hat{\mu}_n, \hat{\nu}_m) = \min_{\pi \in \Pi(\hat{\mu}_n, \hat{\nu}_m)} \sum_{i=1}^n \sum_{j=1}^m \pi_{ij} \|x_i - y_j\|^2. \quad (4)$$

If we suppose that the target distribution is composed by a unique point, *i.e.*,  $\hat{\nu} = \delta_y$ , then we have

$$\mathbf{W}_2^2(\hat{\mu}_n, \hat{\nu}) = \frac{1}{n} \sum_{i=1}^n \|x_i - y\|^2. \quad (5)$$

which is equal to the L2-based criterion. Minimizing with respect to the 2-Wasserstein is equivalent to consider the L2-based criterion (Inkawhich et al., 2019). As a result, our method includes and generalizes the L2-based method.

## D BENEFITS OF OPTIMAL TRANSPORT

Optimal transport-based losses (both exact and sliced) has the following advantages:

- OT losses take into account the underlying metric space (through the cost matrix) on which the probability distributions are defined,
- for non-overlapping distributions such as ours, the Kullback-Leibler divergence is infinite.

To illustrate the first point, we consider the toy example shown in Figure 4. We define four different one-dimensional distributions supported here by five points. We compute the 1-Wasserstein distance and the KL divergence between the red and the blue distributions for each column (results are shown between graphs). The blue mass has been moved near the first point from right to left. The 1-Wasserstein distance captures this mass shift, while the KL divergence does not and remains constant. This toy example highlights that OT losses capture the underlying geometry on which distributions are defined. More details concerning the advantages of OT over other methods can be found in (Arjovsky et al., 2017) (Part 2: Different Distances).

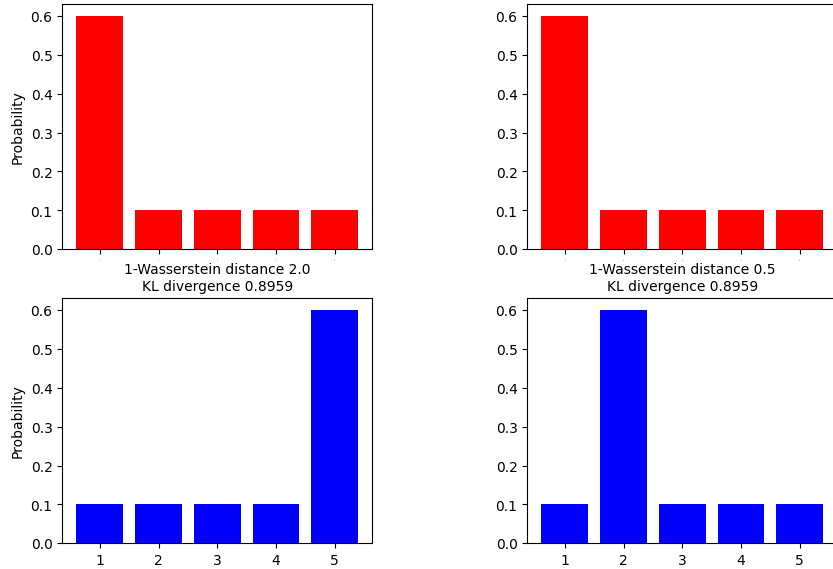


Figure 4: Example of distributions defined on five points with different mass values. The 1-Wasserstein distance and the KL divergence is computed between the red and the blue distribution for each column.

## E MODEL ROBUSTNESS AND PATCH POSITION

In this section, we evaluate the robustness of models according to the patch position in images. We consider the same families of models as before. We define nine patch positions and measure the patch transferability when the patch is fixed at one of these positions. Figure 5 represents the nine patch positions. We regroup these positions into three categories: Corner, Cross, and Center. We measure the patch transferability for a patch of size  $40 \times 40$  ( $\approx 3\%$  image size). Results are averaged over methods (GAP (Brown et al., 2017), LaVAN (Karmon et al., 2018), L2 (Inkawich et al., 2019) and ours), classes, and categories of patch position. Table 3 reports the patch transferability according to its position. CNNs-v1 models are much more biased by the center of images than other network families. The accuracy of CNNs-v1 drops by a factor of 14 % when the patch is moved to corners to the center of images. This effect is not entirely due to the occluding of the object of interest since the patch is very small. Very recent families of networks (CNext and Swin models) are the more balanced networks in using context in images. For these models, the accuracy is nearly the same when the patch is placed in either corners or the center. To measure the actual efficiency of patches and to not occlude the object of interest in the case of large patches, its patches may not be placed in the center of images.

Table 3: Transfer results according to the categories of patch position (Accuracy (%)). Results are averaged over methods, over classes, over patch positions and are for patches of size  $40 \times 40$ .

	Target						
	CNNs-v1	CNNs-v2	ENet	CNext	DeiT	Swin	AT
Clean	74.90	77.63	80.15	82.42	77.81	82.43	65.44
Position							
Corner	71.07	76.57	78.85	81.97	76.33	81.43	64.24
Cross	67.65	75.36	77.71	81.66	75.66	81.44	62.44
Center	61.52	72.01	74.23	80.72	73.94	80.71	57.06

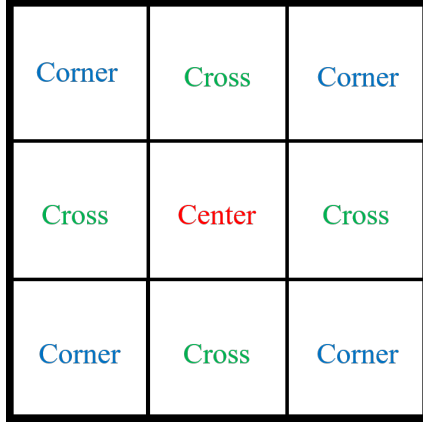


Figure 5: Illustration of the categories of patch positions.

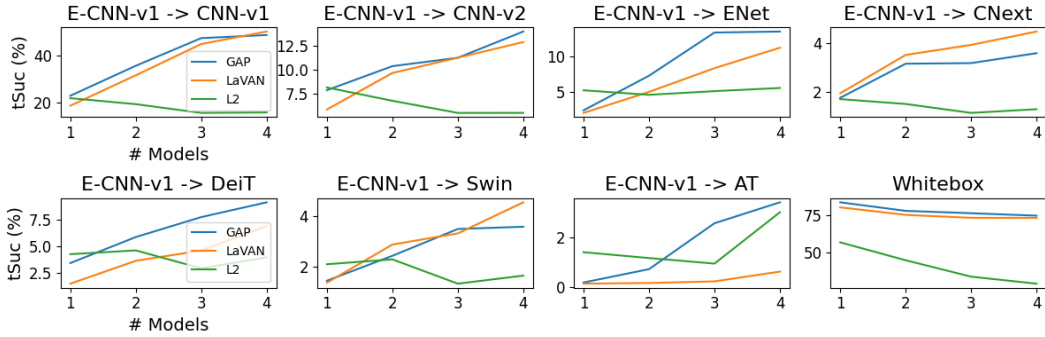


Figure 6: Transfer and whitebox results for patches built on an ensemble of models (tSuc (%)). Results are averaged over classes and over patch sizes. Patches are placed randomly in the image but not at the center of images.

## F ENSEMBLE METHODS

Ensemble methods train a single patch across an ensemble of models simultaneously. We determine if an attacker building his attack on an ensemble of CNN-v1 models can significantly increase its attacking performance on CNext or Swin models. We consider the following ordered list of models  $E\text{-CNN-v1} = \{\text{ResNet50/34/18-v1}, \text{DenseNet121}\}$  in which networks are added to the ensemble in this order. Figure 6 plots the targeted success rate (tSuc) as a function of the number of models in the ensemble. Even with the largest ensemble of four models, patches failed to significantly increase their transferability performances on CNext and Swin models. This result confirms that an attacker expecting to sway all the models uniformly should design his attack on Swin models using our methodology. Figure 6 also shows that the feature point method becomes unstable with the increased number of models in the ensemble.

## G TRANSFERABILITY ON ADVERSARIALY TRAINED MODELS

In this section, we study the robustness of Adversarially Trained (AT) models. We consider two scenarios: when the patch is learned on AT models and when not. To strongly transfer on an AT model, the patch must be designed on an AT model (Table 4). None of the other source models can show good transferability results when applied to AT models. These results suggest that AT models learn different representations than other networks. From Table 5, we see that the GAP method (Brown et al., 2017) and our method are the best procedures to design a patch to target an AT model.



Table 4: Transfer results between categories of models (tSuc (%)). Results are averaged over classes and over patch sizes. Patches are designed using our method ( $\mathbf{W}_2^2$ )<sup>(1)</sup>.

Clean Source	Target							mean / std
	CNNs-v1	CNNs-v2	ENet	CNext	DeiT	Swin	AT	
	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1 / 0
CNNs-v1	39.65	13.01	8.27	2.44	4.89	3.16	0.82	10.32 / 12.56
CNNs-v2	19.0	11.35	3.82	4.51	3.74	4.19	0.45	6.72 / 5.86
ENet	35.12	10.45	32.0	2.27	7.8	3.79	3.49	13.56 / 12.94
CNext	3.47	12.2	0.92	25.14	2.04	15.12	0.16	8.44 / 8.69
DeiT	22.26	11.43	10.18	5.29	39.51	9.25	5.08	14.72 / 11.43
Swin	20.55	17.89	8.09	17.7	13.55	49.1	0.72	18.23 / 14.09
AT	39.75	10.69	17.35	3.51	19.87	5.31	38.95	19.35 / 13.77

Table 5: Transfer results between categories of models (tSuc (%)). Results are averaged over classes and over patch sizes. Patches are placed randomly in the image but not at the center of images.

Clean Method	Target							mean / std
	CNNs-v1	CNNs-v2	ENet	CNext	DeiT	Swin	AT	
	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1 / 0
GAP (Brown et al., 2017)	43.05	11.67	16.7	3.35	20.09	5.23	39.17	19.98 / 14.51
LaVAN (Karmon et al., 2018)	37.27	10.94	14.08	3.43	18.18	5.21	29.96	17.018 / 11.64
L2 (Inkawhich et al., 2019)	6.78	1.86	2.23	0.59	4.39	1.1	8.35	3.618 / 2.77
TnT (Doan et al., 2022)	3.71	1.33	1.41	0.8	2.61	0.85	8.03	2.688 / 2.39
Casper et al. (2022)	5.83	1.38	2.74	0.54	7.55	0.97	13.91	4.78 / 4.48
TTP (Naseer et al., 2021)	35.25	9.45	13.75	2.91	17.92	4.69	35.17	17.028 / 12.43
M3D (Zhao et al., 2023)	6.24	5.61	3.45	0.82	1.82	1.12	2.53	3.088 / 1.98
<b>Ours</b> ( $\mathbf{SW}_2^2$ ) <sup>(1)</sup>	22.52	5.22	8.05	2.17	11.51	3.22	21.57	10.618 / 7.79
<b>Ours</b> ( $\mathbf{W}_2^2$ ) <sup>(1)</sup>	39.75	10.69	17.35	3.51	19.87	5.31	38.95	19.35 / 13.77

## H ROBUSTNESS ACCORDING TO PHYSICAL TRANSFORMATIONS

In this section, we measure the robustness of patches according to physical transformations. We evaluate the L2 (Inkawhich et al., 2019), our exact Wasserstein ( $\mathbf{W}_2^2$ )<sup>(1)</sup> and Sliced-Wasserstein ( $\mathbf{SW}_2^2$ )<sup>(1)</sup><sub>500</sub> patches as they are the only to transfer in the easiest scenario, *i.e.*, without patch rotation, medium brightness and small distance patch-camera (Section 4.3 of the main article). Patch transferability is measured according to z-axis rotations (rotations in the image plane), variation of light (low and high) and distance between camera and the object (the patch is placed near the object). Results are reported in Table 6 and Figure 7. Our patches transfer even in the worst-case scenario (far from the camera or when rotated), while other patches do not. This indicates that our patches may be critical in real-world scenarios. Globally, our method produces patches with better transferability than other methods.

Table 6: Transfer results according to rotations and variation of light (tSuc %). Patches are designed to sway networks to output the class bird house. Patches are printed and placed in the real-world near a cup. Results are averaged over video frames and over all the networks.

Method	z-axis rotations			Variation of light	
	-45°	0°	45°	Low	High
L2 (Inkawhich et al., 2019)	0.8	5.7	0.23	4.4	5.7
<b>Ours</b> ( $\mathbf{SW}_2^2$ ) <sup>(1)</sup> <sub>500</sub>	6.1	11.5	6.53	12	11.5
<b>Ours</b> ( $\mathbf{W}_2^2$ ) <sup>(1)</sup>	<b>7.1</b>	<b>14.8</b>	<b>7.05</b>	<b>12.6</b>	<b>14.8</b>

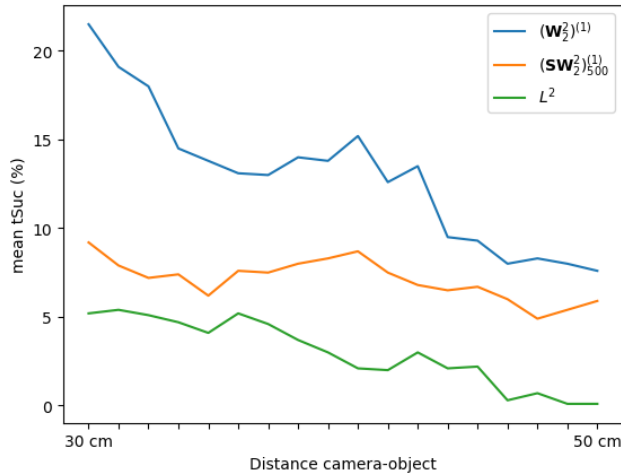


Figure 7: Transfer results as a function of the distance camera-object. Patches are designed to sway networks to output the class bird house. Patches are printed and placed in the real-world near a cup. Results are averaged over video frames and over all the networks.

## I ABLATION STUDIES

In this section, we study the effect of our method hyper-parameters. We solve the exact and the sliced Wasserstein distance for  $p \in \{1, 2\}$  and report the results in Table 7. This Table shows that both values of  $p$  lead to the same transferability. To penalize higher feature values, we set the value of  $p = 2$ .

We launch the Sliced-Wasserstein distance ( $\mathbf{SW}$ ) for the following number of projections:  $K \in \{500, 1000, 5000, 10000, 50000\}$ . There is no clear advantage to considering many projections (Table 10). The best transferability results are obtained with  $K = 500$ .

We now study the effect of the number of attacked layers ( $N$ ). In Table 8, we report the transferability results according to different numbers of targeted layers. We obtain better results for the exact Wasserstein distance when considering multiple layers. We observe that it helps the optimization to converge to a better local minimum, leading to stronger patches. For the Sliced-Wasserstein distance, targeting multiple layers seems counterproductive. Table 9 details the result presented in the article on the choice of the essential layer to target. The last layer of the encoder ( $l = l_J$ ) seems essential to model and close the gap between the two distributions and, particularly, for the Sliced-Wasserstein distance.

To evaluate the data dependency of our method, we create different targeted distributions by changing the number of points which compose it ( $m = 1, 2, 10, 100, 300, 600, 900$ ). We launch the optimization of patches for five different sampling seeds and three different classes. We consider the Swin-T model as the source model. We evaluate patches using the same procedure explained in the main article (Section 4). We report the results of the three runners-up baselines (GAP, LaVAN and TTP). As these methods do not consider distributions, they correspond to straight lines in the figure. From Figure 9 we see that the average targeted success rate (tSuc) increases with respect to the number of target samples. When considering multiple points, our method leads to better transfer results and is more stable than the L2-based method (see B). Our method performs better than decision-boundary-based methods (GAP, LaVAN and TTP). However, we would like to emphasize that our method requires multiple images of the target class to overcome the limitations of the L2-based approach (see Appendix B). This data dependency is a practical limitation of our method. This practical limitation may be simply leveraged by considering the training data of the source model when available.

Table 7: Transfer results according to the power  $p$  (tSuc (%)). Results are averaged over classes and over patch sizes. Patches are designed on Swin-T.

		CNNs-v1	CNNs-v2	ENet	Target CNext	DeiT	Swin	mean / std
$p$	Clean	0.1	0.1	0.1	0.1	0.1	0.1	0.1 / 0
	$(\mathbf{W}_2^2)^{(1)}$	25.62	19.88	10.96	18.84	13.28	55.67	24.04 / 14.91
	$(\mathbf{W}_1^1)^{(1)}$	26.37	20.99	10.86	19.56	13.3	56.98	24.68 / 15.31
	$(\mathbf{SW}_2^2)^{(1)}_{10000}$	27.82	20.22	11.29	18.6	16.66	41.43	22.67 / 9.72
	$(\mathbf{SW}_1^1)^{(1)}_{10000}$	28.74	22.72	11.24	19.89	16.07	43.13	23.63 / 10.26

Table 8: Transfer results according to the number of targeted layers ( $N$ ) (tSuc (%)). Results are averaged over classes and over patch sizes. Patches are designed on the Swin family. Layers  $l_{J-8}$  and  $l_{J-2}$  correspond to the second and third block of Swin models (which are composed by four blocks in total).

		CNNs-v1	CNNs-v2	ENet	Target CNext	DeiT	Swin	mean / std
$(N)$	Clean	0.1	0.1	0.1	0.1	0.1	0.1	0.1 / 0
$(\mathbf{W}_2^2)^{(N)}$	$\{l_J\}$	20.55	17.89	8.09	17.7	13.55	49.1	21.14 / 13.12
	$\{l_{J-2}, l_J\}$	24.87	20.38	9.59	19.77	17.77	48.42	23.47 / 12.06
	$\{l_{J-8}, l_{J-2}, l_J\}$	19.14	12.45	7.52	10.56	13.55	24.75	14.66 / 14.66
$(\mathbf{SW}_2^2)^{(N)}$	$\{l_J\}$	25.26	18.7	9.19	17.27	15.32	44.11	21.64 / 11.11
	$\{l_{J-2}, l_J\}$	24.22	18.26	8.25	15.27	17.47	34.5	19.66 / 8.14
	$\{l_{J-8}, l_{J-2}, l_J\}$	15.94	10.23	6.35	8.6	12.43	17.35	11.82 / 3.89

Table 9: Transfer results according to targeted layer in the single targeted layer setting (tSuc (%)). Results are averaged over classes and over patch sizes. Patches are designed on the Swin family. Layers  $l_{J-8}$  and  $l_{J-2}$  correspond to the second and third block of Swin models (which are composed by four blocks in total).

		CNNs-v1	CNNs-v2	ENet	Target CNext	DeiT	Swin	mean / std
$\mathcal{L}$	Clean	0.1	0.1	0.1	0.1	0.1	0.1	0.1 / 0
$(\mathbf{W}_2^2)^{(1)}$	$l_{J-2}$	17.02	15.03	6.59	14.32	12.55	38.35	17.31 / 9.95
	$l_J$	20.55	17.89	8.09	17.7	13.55	49.1	21.14 / 13.12
$(\mathbf{SW}_2^2)^{(1)}$	$l_{J-8}$	0.3	0.19	0.19	0.14	0.17	0.2	0.2 / 0.05
	$l_{J-2}$	15.39	11.2	5.2	9.08	13.37	20.44	12.45 / 4.81
	$l_J$	25.26	18.7	9.19	17.27	15.32	44.11	21.64 / 11.11

## J DECISION BOUNDARY-BASED METHODS OVERFITTING

In this section, we conduct an additional experiment to support that decision boundary-based methods learn a patch that tends to overfit on the source model classifier. For this purpose, we consider the transfer not between 2 different models but between 2 models sharing the same encoder but different classifiers. We select from the different methods patches trained to attack the source model Swin-T (Liu et al., 2021). On top of this Swin-T encoder, we train a new linear classifier from scratch on the ImageNet train set (Deng et al., 2009). This new linear classifier reaches the same level of clean accuracy as the previous classifier (from Pytorch (Paszke et al., 2019)) while being different. We measured the patch performance when targeting this new network (same encoder, different linear classifier). As expected, the transferability of decision boundary-based patches drops drastically (nearly by half) while our patches transferability remains almost the same.

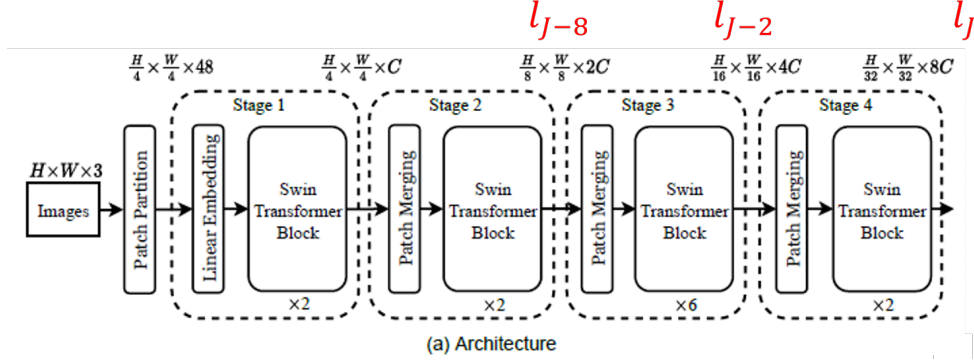


Figure 8: Figure from (Liu et al., 2021). In red are displayed the targeted layers consider in the article.

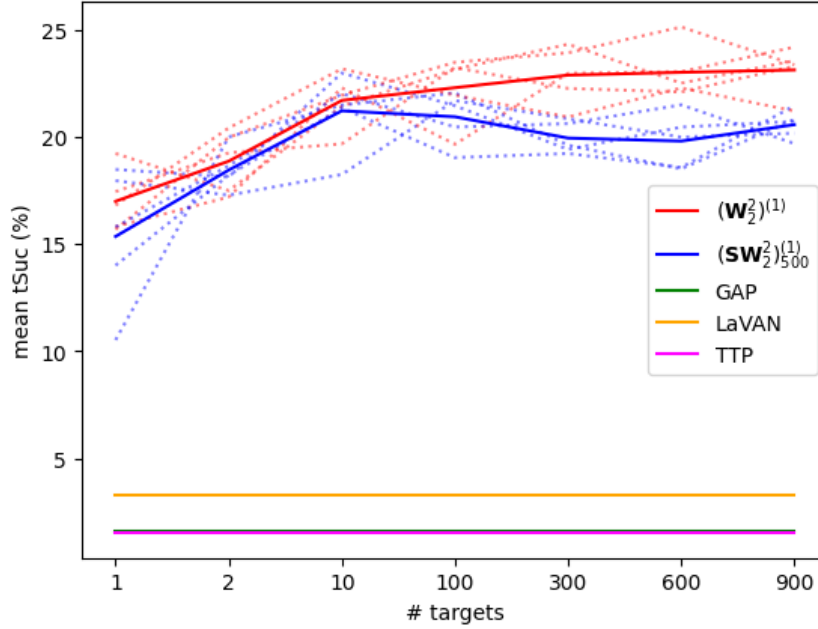


Figure 9: Transfer results as a function of the number of targets points supported in the target distribution (mean tSuc (%)). Each dotted line correspond to a different sampling of points to create the target distribution. The solid line is the average of the five dotted lines. Patches are designed on the Swin-T source model. Results are averaged over three classes, over patch sizes and over all the targeted networks.

Table 10: Transfer results (tSuc (%), higher is better attack) between categories of models. Results are averaged over classes and over patch sizes. Patches are placed randomly in the image without object overlapping. Physical transformations (*e.g.*, noise, rotations) are applied to patches. Control stands for inserting a real object of the corresponding class as a patch.

Method	Clean Source	Target						mean / std
		CNNs-v1	CNNs-v2	ENet	CNext	DeiT	Swin	
		0.1	0.1	0.1	0.1	0.1	0.1	0.1 / 0
Control		2.85	1.59	0.86	0.54	1.57	0.93	1.39 / 0.75
$(\text{SW}_2^{(1)})_{500}$	CNNs-v1	25.25	6.15	4.73	1.7	5.15	2.61	7.6 / 8.04
	CNNs-v2	16.93	8.67	4.02	4.08	5.77	3.56	7.17 / 4.69
	ENet	22.53	5.83	18.8	2.07	8.49	3.03	10.13 / 7.8
	CNext	3.97	11.62	1.1	29.97	3.14	14.75	10.76 / 9.86
	DeiT	23.65	12.16	7.27	5.21	32.39	9.35	15.01 / 9.77
	Swin	25.2	20.21	8.93	19.54	16.16	45.31	22.56 / 11.3
$(\text{SW}_2^{(1)})_{1000}$	CNNs-v1	26.38	6.13	5.59	1.96	5.85	2.8	8.12 / 8.32
	CNNs-v2	16.88	8.82	3.97	3.89	5.8	3.53	7.15 / 4.71
	ENet	23.56	6.45	19.18	2.25	8.55	3.01	10.5 / 8.07
	CNext	4.44	12.07	1.14	33.22	3.24	15.3	11.57 / 10.89
	DeiT	22.77	11.97	7.68	5.36	35.25	9.2	15.37 / 10.48
	Swin	24.2	19.01	8.94	17.73	15.89	44.53	21.72 / 11.16
$(\text{SW}_2^{(1)})_{5000}$	CNNs-v1	26.4	6.11	5.37	1.83	5.2	2.65	7.93 / 8.4
	CNNs-v2	14.49	8.35	3.73	3.64	5.89	3.24	6.55 / 3.96
	ENet	27.44	7.04	19.85	2.16	8.88	3.12	11.42 / 9.2
	CNext	4.52	13.79	1.18	31.54	3.18	16.4	11.77 / 10.45
	DeiT	24.14	12.89	8.37	5.02	36.29	9.17	15.98 / 10.9
	Swin	24.02	19.69	9.53	17.97	15.06	44.74	21.83 / 11.15
$(\text{SW}_2^{(1)})_{10000}$	CNNs-v1	25.73	6.25	5.51	1.86	5.75	2.67	7.96 / 8.11
	CNNs-v2	18.38	10.46	4.19	4.73	6.15	4.01	7.99 / 5.14
	ENet	24.49	6.6	20.26	2.14	8.64	2.98	10.85 / 8.52
	CNext	2.92	9.34	0.92	23.33	2.9	12.18	8.6 / 7.68
	DeiT	23.87	12.22	7.57	4.89	36.3	9.34	15.7 / 11.01
	Swin	23.68	18.08	8.92	17.95	15.42	44.61	21.44 / 11.24
$(\text{SW}_2^{(1)})_{50000}$	CNNs-v1	26.16	6.16	5.4	1.89	5.32	2.7	7.94 / 8.29
	CNNs-v2	13.67	8.71	3.09	4.0	4.67	3.4	6.26 / 3.8
	ENet	25.66	6.06	20.4	2.11	8.73	2.99	10.99 / 8.91
	CNext	3.06	10.97	0.95	27.34	3.34	16.73	10.4 / 9.31
	DeiT	23.95	11.84	8.65	4.6	35.72	8.58	15.56 / 10.86
	Swin	25.26	18.7	9.19	17.27	15.32	44.11	21.64 / 11.11
$(\text{W}_2^{(1)})$	CNNs-v1	39.65	13.01	8.27	2.44	4.89	3.16	11.9 / 12.91
	CNNs-v2	19.0	11.35	3.82	4.51	3.74	4.19	7.77 / 5.69
	ENet	35.12	10.45	32.0	2.27	7.8	3.79	15.24 / 13.25
	CNext	3.47	12.2	0.92	25.14	2.04	15.12	9.82 / 8.64
	DeiT	22.26	11.43	10.18	5.29	39.51	9.25	16.32 / 11.59
	Swin	20.55	17.89	8.09	17.7	13.55	49.1	21.14 / 13.12

Table 11: Transfer results when changing the linear classifier while the encoder remains fixed (variation of tSuc (%)). Patches are designed to fool the Swin-T model (Pytorch version, encoder and linear classifier). The transferability is measured when targeting a new network (same encoder, different linear classifier). Results are averaged over classes and over patch sizes.

Method	Variation of tSuc (%)
GAP (Brown et al., 2017)	- 61.4
LaVAN (Karmon et al., 2018)	- 42.6
TTP Naseer et al. (2021)	- 51.8
<b>Ours</b> $(\text{SW}_2^{(1)})_{500}$	<b>- 0.27</b>
<b>Ours</b> $(\text{W}_2^{(1)})$	<b>- 5.6</b>

## K COMPLEMENTARY TABLES

In this section, we provide additional tables. Table 12 is the same as Table 4 present in the main paper but results are presented for different values of smoothing factors  $\lambda$ .

Table 12: Transfer results on robustified models by LGS defense (Naseer et al., 2019) (tSuc (%)). Patches are designed on Swin models.

		CNNs-v1	CNNs-v2	Target				mean / std
				ENet	CNext	DeiT	Swin	
$\lambda = 1.5$	Clean	0.1	0.1	0.1	0.1	0.1	0.1	0.1 / 0
	GAP (Brown et al., 2017)	0.72	0.87	0.35	0.78	1.13	2.34	1.03 / 0.63
	LaVAN (Karmon et al., 2018)	0.56	0.69	0.3	0.69	0.82	2.65	0.95 / 0.78
	L2 (Inkawhich et al., 2019)	4.79	6.44	1.72	7.79	4.79	13.85	6.56 / 3.75
	TnT (Doan et al., 2022)	0.84	0.59	0.52	0.53	0.7	0.85	0.67 / 0.13
	Casper et al. (2022)	0.37	0.4	0.2	0.32	0.25	0.59	0.36 / 0.13
	TTP (Naseer et al., 2021)	0.68	0.77	0.28	0.68	0.76	1.98	0.86 / 0.53
	M3D (Zhao et al., 2023)	0.83	0.81	0.36	0.77	1.17	1.17	0.85 / 0.27
	<b>Ours (<math>SW_{2/500}^{2(1)}</math>)</b>	<b>10.56</b>	<b>11.86</b>	<b>3.81</b>	<b>18.9</b>	<b>11.67</b>	<b>31.68</b>	<b>14.75 / 8.75</b>
	<b>Ours (<math>W_2^{(1)}</math>)</b>	<b>13.23</b>	<b>13.4</b>	<b>4.37</b>	<b>21.42</b>	<b>13.84</b>	<b>32.08</b>	<b>16.39 / 8.58</b>
$\lambda = 1.9$	Clean	0.1	0.1	0.1	0.1	0.1	0.1	0.1 / 0
	GAP (Brown et al., 2017)	0.61	0.81	0.32	0.68	1.	1.76	0.86 / 0.45
	LaVAN (Karmon et al., 2018)	0.45	0.61	0.26	0.55	0.72	1.72	0.72 / 0.47
	L2 (Inkawhich et al., 2019)	4.05	5.72	1.53	6.6	4.27	11.26	5.57 / 2.99
	TnT (Doan et al., 2022)	0.82	0.61	0.51	0.52	0.62	0.81	0.65 / 0.12
	Casper et al. (2022)	0.32	0.33	0.19	0.25	0.23	0.5	0.3 / 0.1
	TTP (Naseer et al., 2021)	0.56	0.73	0.24	0.59	0.66	1.34	0.69 / 0.33
	M3D (Zhao et al., 2023)	0.68	0.7	0.31	0.7	1.03	1.01	0.74 / 0.24
	<b>Ours (<math>SW_{2/500}^{2(1)}</math>)</b>	<b>8.56</b>	<b>10.49</b>	<b>3.27</b>	<b>15.93</b>	<b>10.39</b>	<b>25.96</b>	<b>12.43 / 7.1</b>
	<b>Ours (<math>W_2^{(1)}</math>)</b>	<b>10.95</b>	<b>11.98</b>	<b>3.78</b>	<b>18.37</b>	<b>12.35</b>	<b>27.07</b>	<b>14.08 / 7.19</b>
$\lambda = 2.3$	Clean	0.1	0.1	0.1	0.1	0.1	0.1	0.1 / 0
	GAP (Brown et al., 2017)	0.52	0.74	0.29	0.58	0.87	1.34	0.72 / 0.33
	LaVAN (Karmon et al., 2018)	0.38	0.55	0.24	0.47	0.64	1.19	0.58 / 0.3
	L2 (Inkawhich et al., 2019)	3.35	4.95	1.35	5.46	3.74	8.93	4.63 / 2.32
	TnT (Doan et al., 2022)	0.8	0.64	0.52	0.52	0.58	0.8	0.64 / 0.12
	Casper et al. (2022)	0.47	0.69	0.22	0.53	0.57	0.92	0.26 / 0.08
	TTP (Naseer et al., 2021)	0.47	0.69	0.22	0.53	0.57	0.92	0.57 / 0.21
	M3D (Zhao et al., 2023)	0.55	0.59	0.27	0.64	0.9	0.9	0.64 / 0.22
	<b>Ours (<math>SW_{2/500}^{2(1)}</math>)</b>	<b>6.76</b>	<b>9.16</b>	<b>2.81</b>	<b>13.1</b>	<b>9.13</b>	<b>20.69</b>	<b>10.28 / 5.59</b>
	<b>Ours (<math>W_2^{(1)}</math>)</b>	<b>8.85</b>	<b>10.61</b>	<b>3.27</b>	<b>15.28</b>	<b>10.86</b>	<b>22.28</b>	<b>11.86 / 5.85</b>

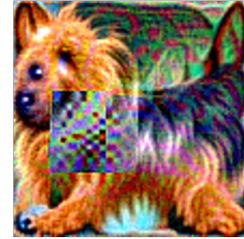
## L PRINTABLE PATCHES

Swin-T

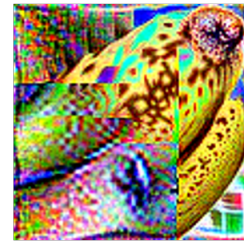
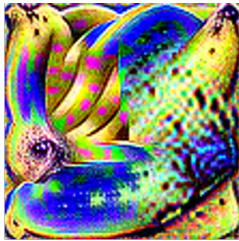
Swin-S

Swin-B

Australian terrier



Banana



Golden



Toaster



Volcano

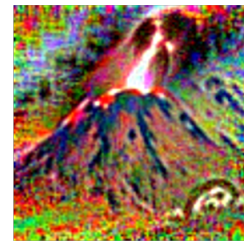
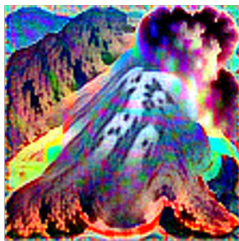


Figure 10: Printable patches designed on Swin models with our distribution-oriented method.



## REFERENCES

- Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.
- Tom B Brown et al. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.
- Stephen Casper, Max Nadeau, Dylan Hadfield-Menell, and Gabriel Kreiman. Robust feature-level adversaries are interpretability tools. *Advances in Neural Information Processing Systems*, 35: 33093–33106, 2022.
- Jia Deng et al. Imagenet: A large-scale hierarchical image database. In *2009 IEEE CVPR*, 2009.
- Bao Gia Doan, Minhui Xue, Shiqing Ma, Ehsan Abbasnejad, and Damith C Ranasinghe. Tnt attacks! universal naturalistic adversarial patches against deep neural network systems. *IEEE Transactions on Information Forensics and Security*, 17:3816–3830, 2022.
- Kevin Eykholt, Ivan Evtimov, Earlene Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1625–1634, 2018.
- Nathan Inkawich et al. Feature space perturbations yield more transferable adversarial examples. In *Proceedings of the IEEE/CVF CVPR*, 2019.
- Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. In *International Conference on Machine Learning*, pp. 2507–2515. PMLR, 2018.
- Mark Lee and Zico Kolter. On physical adversarial patches for object detection. *arXiv preprint arXiv:1906.11897*, 2019.
- Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.
- Muzammal Naseer, Salman Khan, and Fatih Porikli. Local gradients smoothing: Defense against localized adversarial attacks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1300–1307. IEEE, 2019.
- Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. On generating transferable targeted perturbations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7708–7717, 2021.
- Federico Nesti et al. Evaluating the robustness of semantic segmentation for autonomous driving against real-world adversarial patch attacks. In *Proceedings of the IEEE/CVF WACV*, pp. 2280–2289, 2022.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- Anqi Zhao, Tong Chu, Yahao Liu, Wen Li, Jingjing Li, and Lixin Duan. Minimizing maximum model discrepancy for transferable black-box targeted attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8153–8162, 2023.