Supplementary Material: Bias-based Universal Adversarial Patch Attack for Automatic Check-out

Aishan Liu[1*], Jiakai Wang[1*], Xianglong Liu[1,2†], Bowen Cao[1], Chongzhi Zhang[1], and Hang Yu[1]

[1] State Key Lab of Software Development Environment, Beihang University, Beijing, China
[2] Beijing Advanced Innovation Center for Big Data-Based Precision Medicine, Beihang University, Beijing, China

1 The Algorithm of The Proposed Framework

To sum up, we first exploit the perceptual bias of models and extract a textural prior from hard examples by adopting the style similarities. To further alleviate the heavy dependency on large amounts of data in training universal attacks, we further exploit the semantic bias. As the class-wise preference, prototypes are introduced and pursued by maximizing the multi-class margin. Using the textural prior as initialization, we train our adversarial patches using the prototypes as training data. The illustration of our two-staged adversarial patch attack algorithm can be found in Algorithm 1.

Algorithm 1: Bias-based Universal Adversarial Patch Attack

```
Input: hard example set $\mathcal{X}^h = \{x_i^h | i = 1, ..., r\}$, target model $F$
Output: bias-based patch $\delta^{adv}$
Stage1 : Perceptually Biased Prior Generation
initial $x^*$ by randomly select a hard example from $\mathcal{X}^h$;
for the number of fusion epochs do
    for $m = r/batchsize$ steps do
        sample a minibatch of hard examples from $\mathcal{X}^h$;
        optimize $x^*$ to minimize $\mathcal{L}_f$;
obtain the prior patch $\delta^*$ through attention by Eqn (6);
Stage2: Training with Semantically Biased Prototype
get class prototypes set $\mathbf{I} = \{I_1, I_2, ...I_n\}$ by Eqn (8);
for the number of training epochs do
    for $k = n/batchsize$ steps do
        sample a minibatch of prototypes from $\mathbf{I}$;
        optimize the adversarial patch $\delta^{adv}$ to minimize $\mathcal{L}_t$ with prototypes;
```

* These authors contributed equally to this work.
† Corresponding author.

2 Ablations on Prototypes and Transformation Module

2.1 The Effectiveness of Class Prototypes

In this section, we report the improvement brought by prototypes and evaluate the accuracy of the prototypes using ResNet-152. We first study it by using different amounts of prototypes. Specifically, we mix class prototypes and item images from the RPC dataset in different ratios. We use them to train different adversarial patches and assess their attacking ability. Note that the total number of the training data is fixed as 1000. As shown in Table 1, with the increasing number of prototypes, adversarial patches becoming stronger (*i.e.*, lower accuracy).

Table 1. Training with a different mixture of class prototypes and original item images using the same amount of training data. Obviously, training with class prototypes give adversarial patches with the strongest attack ability.

Mixture settings (#Prototypes : #Item images)	*top*-1
1000 : 0	**6.51**
750 : 250	7.81
500 : 500	10.03
250 : 750	11.55
0 : 1000	12.43

2.2 Transformation Module

Studies have shown that adversarial examples are ineffective to environmental conditions, *e.g.*, different rotations, illuminations, *etc.* In the ACO scenario, items are often scanned from different views with different lighting conditions. Thus, we introduce a transformation module to reduce the impact of environmental conditions to the the attack ability. Here, we study the effectiveness of different transformation types we used in the module. Specifically, we employ ResNet-152 as the target model and execute only one kind of transformation in each experiment. The results show that enabling transformations can increase attacking ability in ACO scenario with lower accuracy (*i.e.*,**11.98%** to 13.01% in rotation setting, **22.26%** to 30.70% in distortion setting, **16.38%** to 21.10% in affine setting).