Technical Appendix

Algorithms and Parameters

We present the details of the preprocessing functions used in the GYM. Each method is modified or upgrade based on proposed methods with the aid of randomness. The hyperparameters we adopted in the preprocessing procedures for both the intensive preprocessing and the lightweight preprocessing are listed in Table 1.

Notation	Meaning	Value
δ	distortion limit of the Optical Distortion	0.5
γ_1	Gamma Compression's gamma value	0.6
γ_2	Gamma Extension's gamma value	2.6
σ	scale limit of the RSDP	1.3
T	translation limit of the SAT	0.16
S	sacaling limit of the SAT	0.16
R	rotation limit of the SAT	4

Table 1: Hyperparameters' settings during the Preprocessing.

ALGORITHM 1: *Random Pincushion Distortion*

Input: original image $I \in \mathbb{R}^{h \times w}$
Output: distorted image $I' \in \mathbb{R}^{h \times w}$
Parameters: distortion limit δ;

/* 1.Acquire distortion parameter δ_k */
$\delta_k \sim \mathcal{U}(-\delta, 0)$;
/* 2.Acquire Distortion Maps */
$c_x = \lfloor (w/2) \rfloor, c_y = \lfloor (h/2) \rfloor$;
$P_{set} = \{(m, n) \in \{(0, ..., w) \times (0, ..., h)\}\}$;
for (u, v) *in* $P_{set} \backslash \{(m, n)\}$ **do**
  $map_x(u, v) = ((u - c_x) \times (1 + k)) + c_x$;
  $map_y(u, v) = ((v - c_y) \times (1 + k)) + c_y$;
end
/* 3.Remapping I to I' */
for (u, v) *in* $P_{set} \backslash \{(m, n)\}$ **do**
  $I'(u, v) = I(map_x(u, v), map_y(u, v))$;
end
return I';

Optical Distortion

Different from (Liu, Malcolm, and Xu 2010), the Optical Distortion we upgraded and utilized in the GYM is based on assigning a random distortion value chosen from a uniform distribution of the distortion limit. This random process can distort each sample on a different scale for a different time, thus better help the infected model better adapt to the remapping distortions. The details of the Random Pincushion Distortion we proposed and improved in the GYM are explained in Algorithm 1. The random pincushion distortion can be interpreted into three phases. For starters, we acquire a random distortion value, δ_k, from a uniform distribution between $-\delta$ to 0. Using this randomly sampled δ_k, we can acquire two pincushion maps for horizontal and vertical indexes, respectively. Finally, by broadcasting those two maps for each pixel, we can output the result. During the experiment, we set the δ as 0.5 based on experimental analysis.

Gamma Compression and Extension

Inspired by the previous work (Kumari, Thomas, and Sahoo 2014), the Gamma Compression and the Gamma Extension are fine-tuned and used in the median filters set to merging pixels' values and enhance the effects of the median filters. The Gamma value of the Gamma Compression procedure is set to 0.6, which acquires a Look-Up Table shown in the middle of Figure 1. As demonstrated that larger values from the original pixels range (the left part of 1) are mapping with a larger value close to the maximum value (255), thus helps larger values to bend in. As a result, the median filter can work more efficiently to smoothen pixels of low value. Vice versa, with a Gamma value set to 2.6, we can use the help of the Gamma Extension to merge small values, thus better smoothen large pixels. We summarize the Gamma Compression and Extension as a single function shown in Algorithm 2. As demonstrated, the Gamma Transformation we used here in the experiment can be interpreted as two functional parts. First, we acquire the LUT based on the Gamma value, γ. Then, the output image can be obtained by using the value of the corresponding position in the LUT to replace the original pixel value. The function with a Gamma value larger than 1 conducts extension, and a Gamma value smaller than 1 performs compression. We chose 0.6 and 2.6 as the Gamma values for the compression and extension based on experimental results, as they can achieve better results during the inference after GYM fine-tuning.

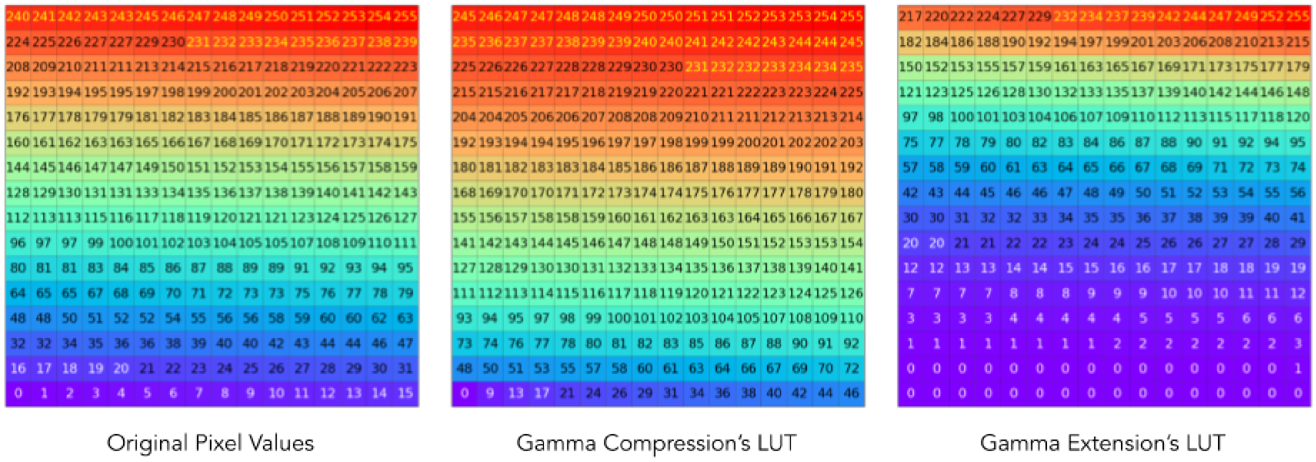

Figure 1: Different Gamma Look-Up Tables (LUTs) used in the Median Filters set: The left is the original pixel values, ranging from 0 to 255; the Gamma Compression uses a Gamma value of 0.6, which lead to the LUT shown in the middle; the Gamma Extention uses a Gamma value of 2.6, which lead the left LUT.

ALGORITHM 2: *Gamma Transformation*

Input: original image $I \in \mathbb{R}^{h \times w}$
Output: transformed image $I' \in \mathbb{R}^{h \times w}$
Parameters: Gamma Value γ;

```
/* 1.Acquire LUT */
```
$T = range(0:255)^{16\times16}$;
$LUT = (T/255)^{\gamma} \times 255$;
```
/* 2.Assigning New Values */
```
$P_{set} = \{(m,n) \in \{(0,...,w) \times (0,...,h)\}\}$;
for (u,v) *in* $P_{set} \backslash \{(m,n)\}$ **do**
    $(x,y) = where(T == I(u,v))$;
    $I'(u,v) = LUT(x,y)$;
end
return I';

ALGORITHM 3: *RSDP*

Input: original image $I \in \mathbb{R}^{l \times l}$
Output: distorted image $I' \in \mathbb{R}^{l \times l}$
Parameters: scale limit σ;

```
/* 1.Acquire random parameter */
```
$Len_{max} = \lfloor (l \times \sigma) \rfloor$;
$Len \sim \lfloor \mathcal{U}(l, Len_{max}) \rfloor$;
$l_{rem} = Len_{max} - Len$;
$x_1 \sim \lfloor \mathcal{U}(0, l_{rem}) \rfloor, y_1 \sim \lfloor \mathcal{U}(0, l_{rem}) \rfloor$;
$x_2 = l_{rem} - x_1, y_2 = l_{rem} - y_1$;
/* 2.Padding to Len_{max} */
$I' = reshape(I)$ s.t. $I' \in \mathbb{R}^{Len \times Len}$;
$I' = pad(I', ((x_1, x_2), (y_1, y_2)), value = 0)$
s.t. $I' \in \mathbb{R}^{Len_{max} \times Len_{max}}$;
/* 3.Reshape I' to the size of I */
$I' = \texttt{reshape}(I')$ s.t. $I' \in \mathbb{R}^{l \times l}$;
return I';

Random Scale Down with Padding (RSDP)

We proposed Random Scale Down with Padding (RSDP) as a tool to help the infected model to better adapt to affine transformations. The details of the proposed preprocessing function are explained in 3. The σ we used in the experiment is set to 1.3 to downscale the input image in a range of range (0.8,1). The whole process of the proposed RSDP can be interpreted as three functional parts. First, the algorithm acquires random parameters for the scaling and the padding. This includes after-padding size, Len_{max}; resizing size, Len; the number of pixels to pad to reach the after-padding size, l_{rem}; and padding coordinates, (x_1, x_2) and (y_1, y_2). Padding the resized image using the padding coordinates to (Len_{max}, Len_{max}), we can acquire a black canvas patched with the resized original input. With resizing the image back to the original size, we can acquire the final result. Based on the experiment, we found that resizing the image from 0.8 to 1 times smaller can best help the infected model adapts to the affine transformation.

Stochastic Affine Transformation

We adopt the Stochastic Affine Transformation (SAT) (Zeng et al. 2020) to further help the infected model adapt the affine transformation in GYM Fine-tuning and invalidate potential triggers during inference. The details of the SAT are explained as follows in the Algorithm 4. We adopt the same settings from (Zeng et al. 2020), T, 0.16, S, 0.16, and R, 4. The whole process of the SAT can be functionally interpreted as three parts, namely translation, rotation, and scaling.

Attacking and Preprocessing Results

The attacking and preprocessing results over patched samples are shown in Figure 2,3,and 4. Please noted that the intensive preprocessed patched data are not used during the GYM fine-tuning procedure. We only adopt the intensive preprocessed patched data during the inference before and after the fine-tuning (Inference(I)).

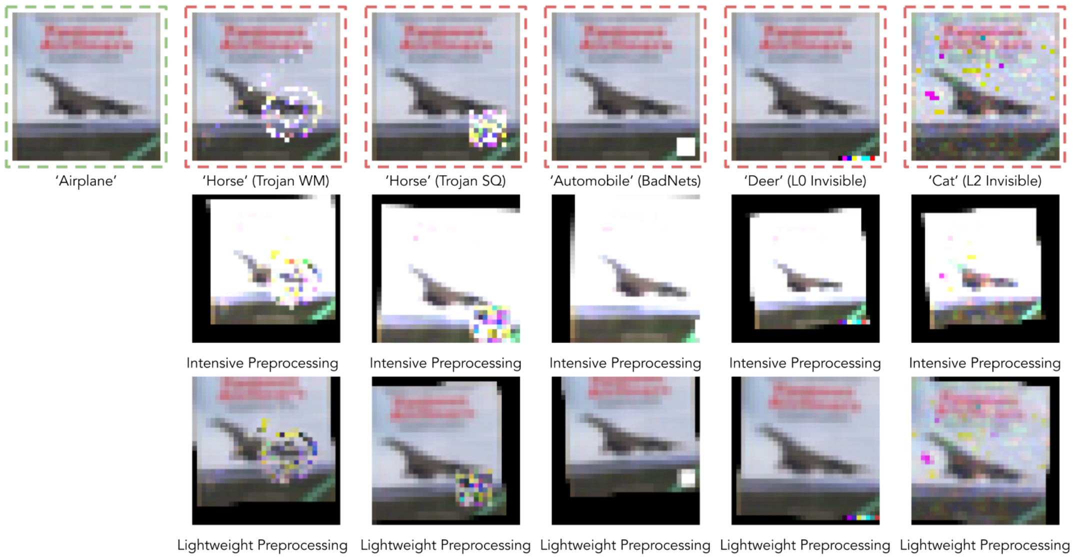

Figure 2: The attacking and preprocessing results over the Cifat10 dataset.

ALGORITHM 4: SAT

Input: original image $I \in \mathbb{R}^{h \times w}$
Output: transformed image $I' \in \mathbb{R}^{h \times w}$
Parameters: translation limit T; scaling limit S, rotation limit R.

$I' = O^{h \times w}$;
`/* 1.Translation */`
$\delta_x \sim \mathcal{U}(-T, T)$;
$\delta_y \sim \mathcal{U}(-T, T)$;
$\Delta_x = \delta_x \times w$;
$\Delta_y = \delta_y \times h$;
if $(x + \Delta_x \in (0, w)) \wedge (y + \Delta_y \in (0, h))$ **then**
    $I'(x, y) = I(x + \Delta_x, y + \Delta_y)$;
end
`/* 2.Rotation */`
$\delta_r \sim \mathcal{U}(-R, R)$;
$\Delta_r = \delta_r \times \pi/180$;
for (x_i, y_j) *in* $\{(x, y) | x \in (0, w), y \in (0, h)\}$ **do**
    $x_i' = -(x_i - \lfloor w/2 \rfloor) \times sin(\Delta_r) + (y_j - \lfloor h/2 \rfloor) \times cos(\Delta_r)$;
    $y_j' = (x_i - \lfloor w/2 \rfloor) \times cos(\Delta_r) + (y_j - \lfloor h/2 \rfloor) \times sin(\Delta_r)$;
    $x_i' = \left\lfloor x_i' + \lfloor w/2 \rfloor \right\rfloor$;
    $y_j' = \left\lfloor y_j' + \lfloor h/2 \rfloor \right\rfloor$;
    if $(x_i' \in (0, w)) \wedge (y_j' \in (0, h))$ **then**
        $I'(x_i, y_j) = I(x_i', y_j')$;
    end
end
`/* 3.Scaling */`
$\delta_s \sim \mathcal{U}(1 - S, 1 + S)$;
$h_{new} = \delta_s \times h$;
$w_{new} = \delta_s \times w$;
$I' = \mathrm{reshape}(I', (h_{new}, w_{new}))$;
if $\delta_s > 1$ **then**
    $I'(x, y) = cropping(I', (h, w))$;
end
if $\delta_s < 1$ **then**
    $I'(x, y) = padding(I', (h, w))$;
end
return I';

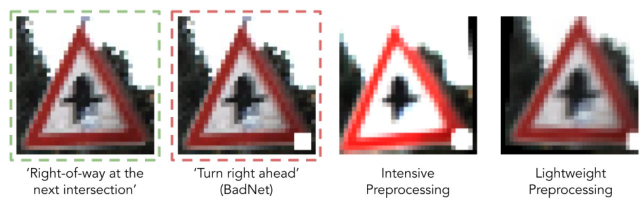

Figure 3: The attacking and preprocessing results over the GTSRB dataset.

Figure 4: The attacking and preprocessing results over the PubFig dataset.

As shown in each figure, the triggers in each sample preprocessed by the intensive preprocess become hard to recognize by human eyes. Thus, it is intuitive that such intensive preprocessing can help the infected models revise their decision boundaries to encompass those patch data to their original classes. With the decision boundary being shifted with the intensive preprocessed data, the lightweight preprocessing can help the fine-tuned model achieve a more accurate result, with less preprocessing procedures being adopted.

Please noted that the lightweight preprocessing is actually proposed based on the intensive preprocessing, which simplified some steps in the intensive preprocessing. The intensive preprocessing is developed from a functional perspective to help the infected model better and faster adapt to the lightweight preprocessing, thus achieving high accuracy for an efficient deployment.

References

Kumari, A.; Thomas, P. J.; and Sahoo, S. 2014. Single image fog removal using gamma transformation and median filtering. In *2014 annual IEEE India conference (INDICON)*, 1–5. IEEE.

Liu, T.; Malcolm, A.; and Xu, J. 2010. Pincushion distortion correction in x-ray imaging with an image intensifier. In *Fourth International Conference on Experimental Mechanics*, volume 7522, 75223T. International Society for Optics and Photonics.

Zeng, Y.; Qiu, H.; Memmi, G.; and Qiu, M. 2020. A Data Augmentation-based Defense Method Against Adversarial Attacks in Neural Networks. *arXiv preprint arXiv:2007.15290* .