Supplementary Material for Random Entangled Tokens for Adversarially Robust Vision Transformer

1. Deduction Details of Optimizing r_2 via Eq. (12)

Firstly, let us explicitly define the dimensions of certain variables: X, R_1, and R_2 have dimensions of $d \times n$, $d \times m$, and $d \times m$, respectively. For each element $x_{(s,t)}$ in X, we can calculate the gradient with respect to $x_{(s,t)}$ of R_1 as follow:

$$\frac{\partial R_1}{\partial x_{(s,t)}} = \frac{1}{dm} \sum_{i}^{d} \sum_{j}^{m} \frac{\partial R_{1(i,j)}}{\partial x_{(s,t)}}, \tag{S1}$$

where $s = 1, 2, \cdots, d$; $t = 1, 2, \cdots, m$. Note that we use the average gradient here to reduce the dimension of the matrix derivative, thereby multiplied by an average coefficient $1/dm$. Thus, the full gradient with respect to X of R_1 is

$$\frac{\partial R_1}{\partial X} = \left[\frac{1}{dm} \sum_{i}^{d} \sum_{j}^{m} \frac{\partial R_{1(i,j)}}{\partial x_{(s,t)}} \right]_{d \times n}. \tag{S2}$$

Similarly, the full gradient with respect to X of R_2 is

$$\frac{\partial R_2}{\partial X} = \left[\frac{1}{dm} \sum_{i}^{d} \sum_{j}^{m} \frac{\partial R_{2(i,j)}}{\partial x_{(s,t)}} \right]_{d \times n}. \tag{S3}$$

For the multiplication in Eq. (12), we utilize the Hadamard product for computing the element-wise product to keep the dimension of the object $f(r_2)$ constant with the gradients:

$$\begin{aligned} f(r_2) = \frac{\partial R_1}{\partial X} \frac{\partial R_2}{\partial X} &= \left[\frac{1}{dm} \sum_{i}^{d} \sum_{j}^{m} \frac{\partial R_{1(i,j)}}{\partial x_{(s,t)}} \right] \odot \left[\frac{1}{dm} \sum_{i}^{d} \sum_{j}^{m} \frac{\partial R_{2(i,j)}}{\partial x_{(s,t)}} \right] \\ &= \left[\frac{1}{d^2 m^2} \left(\sum_{i}^{d} \sum_{j}^{m} \frac{\partial R_{1(i,j)}}{\partial x_{(s,t)}} \right) \left(\sum_{i}^{d} \sum_{j}^{m} \frac{\partial R_{2(i,j)}}{\partial x_{(s,t)}} \right) \right]. \end{aligned} \tag{S4}$$

Therefore, the dimension of $f(r_2)$ is still $d \times n$. Here, we rewrite the object function as

$$\min_{r_2} \left[\frac{1}{d^2 m^2} \left(\sum_{i}^{d} \sum_{j}^{m} \frac{\partial R_{1(i,j)}}{\partial x_{(s,t)}} \right) \left(\sum_{i}^{d} \sum_{j}^{m} \frac{\partial R_{2(i,j)}}{\partial x_{(s,t)}} \right) \right]_{d \times n}. \tag{S5}$$

Similar to Eq. (S1), we use the average gradient for each element $r_{2(1,k)}$ ($k = 1, 2, \cdots, n$) of r_2. Thus, we obtain the gradient of $f(r_2)$ as

$$\begin{aligned}\frac{\partial f(r_2)}{\partial r_{2(1,k)}} &= \frac{1}{dn}\sum_{s}^{d}\sum_{t}^{n}\frac{\partial\left(\frac{1}{d^2m^2}\left(\sum_i^d\sum_j^m\frac{\partial R_{1(i,j)}}{\partial x_{(s,t)}}\right)\left(\sum_i^d\sum_j^m\frac{\partial R_{2(i,j)}}{\partial x_{(s,t)}}\right)\right)}{\partial r_{2(1,k)}} \\ &= \frac{1}{d^3m^2n}\sum_{s}^{d}\sum_{t}^{n}\frac{\partial\left(\left(\sum_i^d\sum_j^m\frac{\partial R_{1(i,j)}}{\partial x_{(s,t)}}\right)\left(\sum_i^d\sum_j^m\frac{\partial R_{2(i,j)}}{\partial x_{(s,t)}}\right)\right)}{\partial r_{2(1,k)}}.\end{aligned} \tag{S6}$$

The total gradient with respect to r_2 of $f(r_2)$ can be formulated as

$$\frac{\partial f(r_2)}{\partial r_2} = [\frac{\partial f(r_2)}{\partial r_{2(1,1)}}, \frac{\partial f(r_2)}{\partial r_{2(1,2)}}, \cdots, \frac{\partial f(r_2)}{\partial r_{2(1,n)}}]_{1\times n}. \tag{S7}$$

Thus, we can use Eq. (S7) to iteratively update r_2 with some optimizer (like Adam [3]). Besides, Equation (S7) is decoupled from the input token X.

2. More Experimental Results

In this section, we provide more experimental results to evaluate the effectiveness of our proposed ReiT. Expectation Over Transformations (EoT) [2] is an effective way to generate robust adversarial examples that are generated by multiple perturbations or transformations. Here, we leveraged EoT plus PGD with 10 and 20 steps (EoT PGD-10 and EoT PGD-20) to further evaluate our proposed method. The basic configurations are the same as the PGD attack and the hyperparameter of EoT iteration is set as 2. Besides, we also used two black-box attacks, e.g., one-pixel attack (OnePixel) [4] and square attack (Square) [1], to evaluate the black-box performance of our proposed method. For the configurations for the one-pixel attack, the number of pixels to change is set as 1; the number of steps is set as 10; the population size, i.e. the number of candidate agents or "parents" in differential evolution, is set as 10; the maximum batch size during inference is set as 128. For the configurations for the square attack, the norm of the attack is set as ℓ_∞; the maximum perturbation is set as $8/255$; the max number of queries (each restart) is set as 5,000; the number of random restarts is set as 1; the parameter to control size of squares is set as 0.8; the loss function optimized is set as "margin"; the adapt schedule of norm to the max number of queries is set as *True*. The results are shown in Tab. S1, which showcases that our proposed method achieves non-trial improvement under EoT and black-box attacks.

Model	Method	EoT PGD-10	EoT PGD-20	OnePixel	Square
DeiT-T	vanilla	49.21	49.13	78.77	72.71
	ReiT	**49.87**	**49.74**	**80.02**	**79.69**

Table S1. Results (%) on CIFAR-10 dataset under EoT PGD and two black-box attacks. The best results are stressed in **BOLD**.

Besides, as shown in Sec. 2, we use the metric of floating point operations (FLOPs) to evaluate the complexity and computational cost in the inference stage for the vanilla method and the proposed method. The proposed module leads to a very small amount of extra computation cost. Besides, on big-scale datasets, the proportion of the extra cost can be almost negligible (less than 0.5%). Thus, the proposed module hardly affects scalability and practical deployment on large-scale datasets or in real-time applications.

Dataset	Method	ViT-S/DeiT-S	ViT-T/DeiT-T	Swin-S	Swin-T
CIFAR-10/100	vanilla	1.3839G	0.3469G	0.6960G	0.3556G
	ReiT	1.3892G (+0.38%)	0.3482G (+0.38%)	0.7093G (+1.91%)	0.3635G (+2.24%)
ImageNet(te)	vanilla	4.2488G	1.0786G	8.5450G	4.3719G
	ReiT	4.2541G (+0.12%)	1.0800G (+0.12%)	8.5875G (+0.50%)	4.3931G (+0.49%)

Table S2. FLOPs of different models with different methods on different datasets in the inference stage.

References

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *ECCV*, 2020. 2

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *ICML*, 2018. 2

[3] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 2

[4] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019. 2