max planck institut
informatik

Improving Robustness of Vision Transformers by Reducing Sensitivity to Patch Corruptions

CVPR 2023
Session and Poster ID: TUE-AM-392

Yong Guo, David Stutz, Bernt Schiele

Max Planck Institute for Informatics

Overview

Motivation:

- ViTs are often more robust than CNNs but still remain very vulnerable against corruptions and perturbations.
- We seek to understand the vulnerability of ViTs by investigating the stability of self-attention mechanism.
- ViTs are inherently patch-based models.

Idea & Method:

❖ We explicitly study the **sensitivity to patch corruptions/perturbations**.

❖ We propose a new method to improve robustness by **Reducing Sensitivity to Patch Corruptions (RSPC)**.

  - Finding particular vulnerable patches to introduce corruptions
  - Aligning the features between the clean and corrupted examples

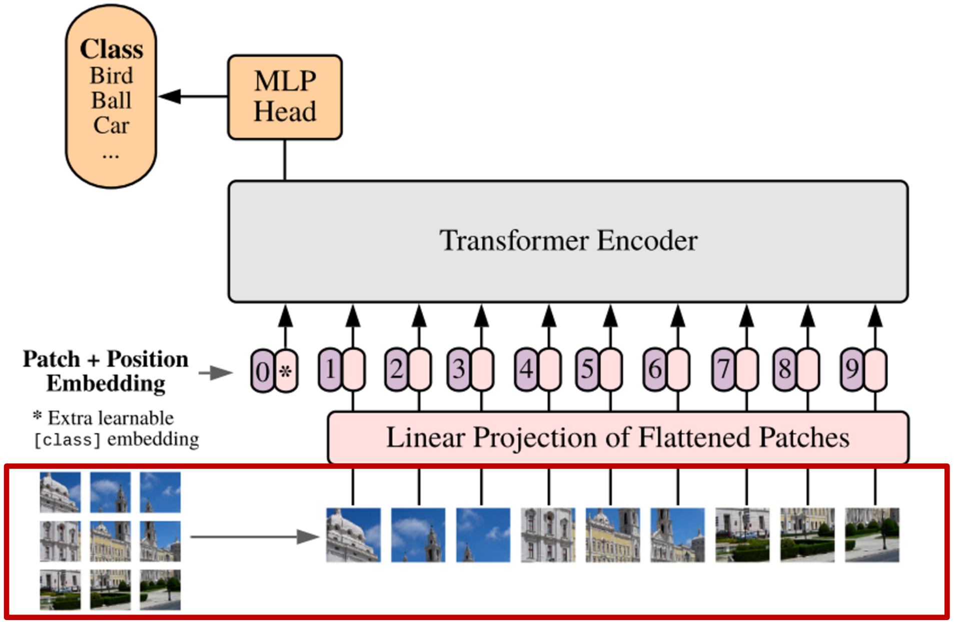
Results:

- The robustness improvement against patch corruptions can **generalize well to diverse architectures on various robustness benchmarks**.
- We can show, both qualitatively and quantitatively, that these improvements stem from the **more stable attention mechanism across layers**.

Background & Motivation

- ViTs are often more robust than CNNs but still remain very vulnerable against corruptions and perturbations.
- We seek to understand the vulnerability of ViTs by investigating the stability of self-attention mechanism.

Vision Transformer (ViT)

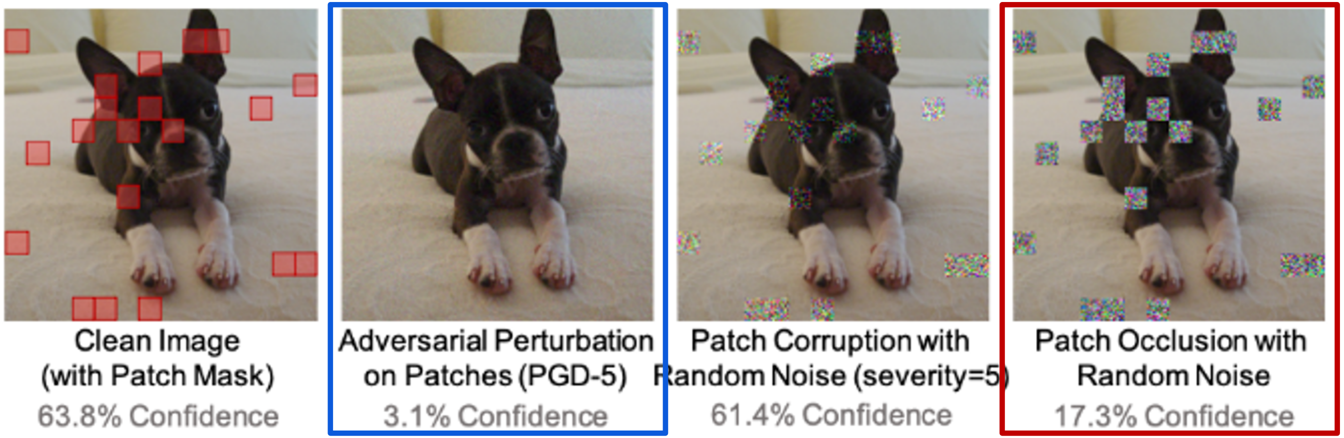

Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
0
*
1
2
3
4
5
6
7
8
9
* Extra learnable [class] embedding
Linear Projection of Flattened Patches

❖ Since ViTs are inherently patch-based, we explicitly study the **sensitivity to patch corruptions/perturbations**.

Sensitivity to Patch Perturbations/Corruptions

Experimental settings:

- Randomly sample a small number of patches to be perturbed/corrupted (10%, keeping the mask fixed)
- Introduce different perturbations and corruptions into the selected patches



Clean Image (with Patch Mask)
63.8% Confidence

Adversarial Perturbation on Patches (PGD-5)
3.1% Confidence

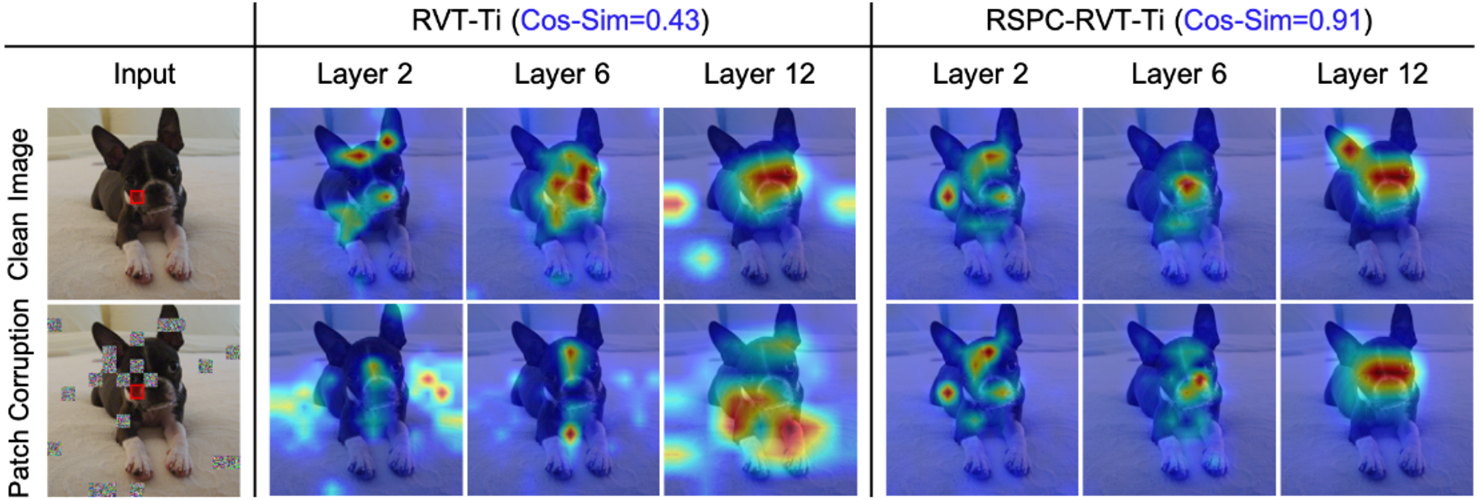
Patch Corruption with Random Noise (severity=5)
61.4% Confidence

Patch Occlusion with Random Noise
17.3% Confidence

- **Transformers are very sensitive to patch perturbations**
  - Transformers can be easily misled by the adversarial perturbations only on very few patches
  - Nevertheless, generating adversarial perturbations and training against them is very expensive.
- **Directly introducing corruptions only yields marginal degradation in terms of confidence score**
  - Introducing corruptions is much more efficient but not very effective
- **Occluding patches with noise can significantly hamper the prediction**
  - A good proxy of adversarial patch perturbations

Sensitivity of ViT to Patch-based Corruptions

- We construct the patch-based corruptions (by occluding a small number of patches with noise, e.g., 10%) and study how the attention maps would change in each layer.

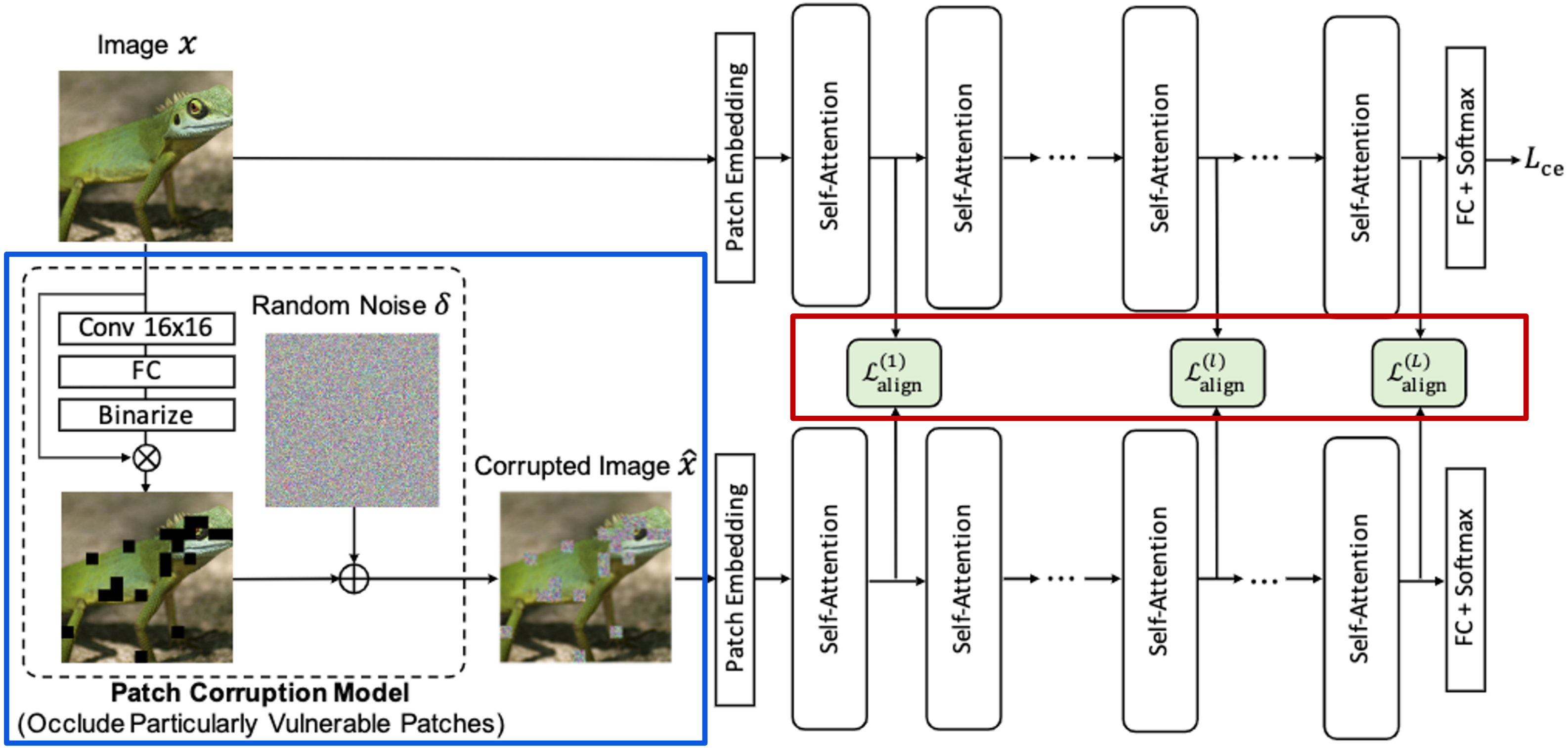


❖ The self-attention mechanism is very sensitive to patch-based corruptions, which could be a major reason for the lack of robustness.

Proposed Method

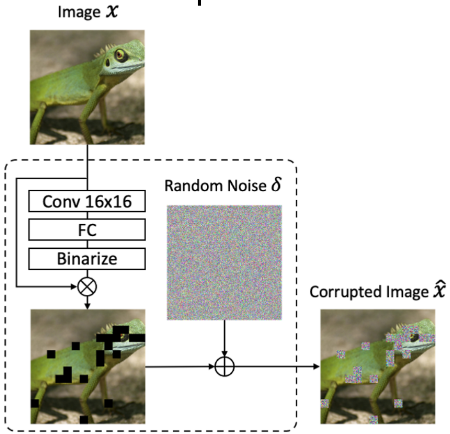
We seek to reduce the sensitivity of self-attention layers against patch corruptions.

- Finding particular vulnerable patches to introduce corruptions
- Aligning the features between the clean and corrupted examples



Finding Vulnerable Patches to be Corrupted

- ❖ Patch Corruption Model



Patch Corruption Model
(Occlude Particularly Vulnerable Patches)

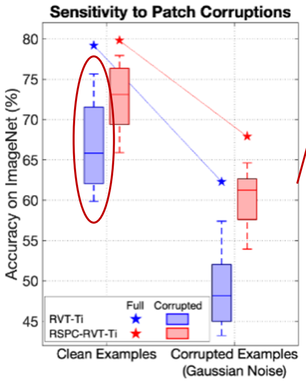
Notations: x: clearn sample
$\hat{x}$: occluded sample
$\mathcal{C}$: occlusion model
ρ: occlusion ratio

$$\hat{x} = \mathcal{C}(x;\rho)\cdot x + (1 - \mathcal{C}(x;\rho))\cdot\delta$$

- Conv: extract features for each patch (patch size=16x16)
- Binarize: select the top ρ% patches and produce a binary map

Making it differentiable with the Straight Through Estimator (STE)

- ❖ Find the patches that changes the intermediate features most:



Vary large variance: some patches greatly affect the performance while the others may not

$\mathcal{F}_l(*)$: features of the l-th layer

$$\max_{\mathcal{C}}\ \mathbb{E}_{x\sim\mathcal{D}}\,\mathcal{L}_{\text{align}}(x,\hat{x}),$$

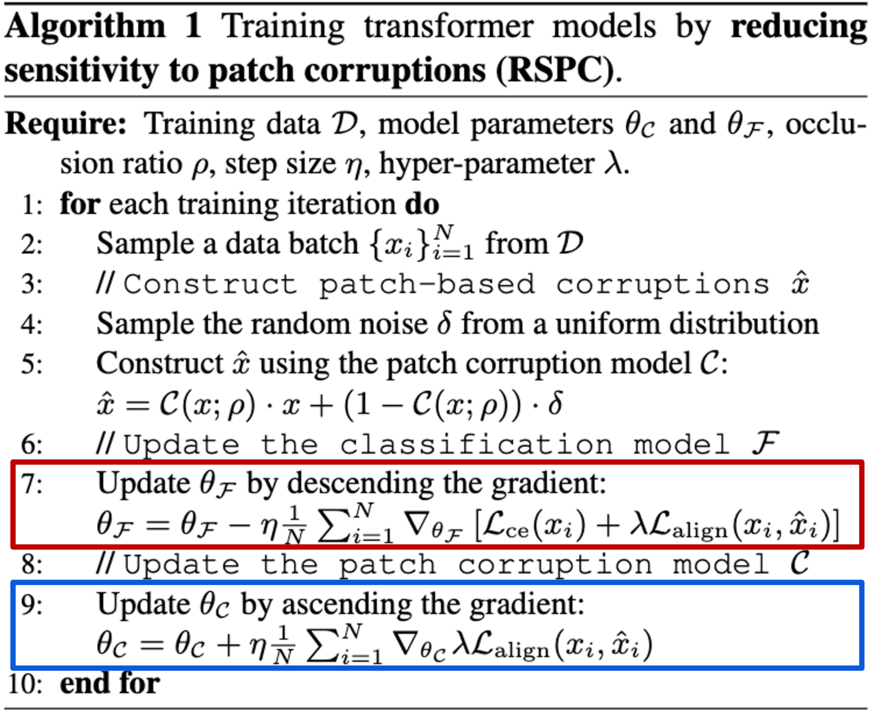
$$\text{where}\ \ \mathcal{L}_{\text{align}}(x,\hat{x}) = \frac{1}{L}\sum_{l=1}^{L}\|\mathcal{F}_l(x)-\mathcal{F}_l(\hat{x})\|^2$$

Reducing Patch Sensitivity via Feature Alignment

$$\min_{\mathcal{F}} \max_{\mathcal{C}} \ \mathbb{E}_{x\sim\mathcal{D}}[\mathcal{L}_{\text{ce}}(x) + \lambda\mathcal{L}_{\text{align}}(x, \hat{x})]$$

- **Adversarial objective**
- Maximize the loss to find vulnerable patches
- Minimize the loss to reduce patch sensitivity

- **Training both models using a single backpropagation**
- Descend the gradient for the classification model $\mathcal{F}$
- Ascend the gradient for the patch corruption model $\mathcal{C}$

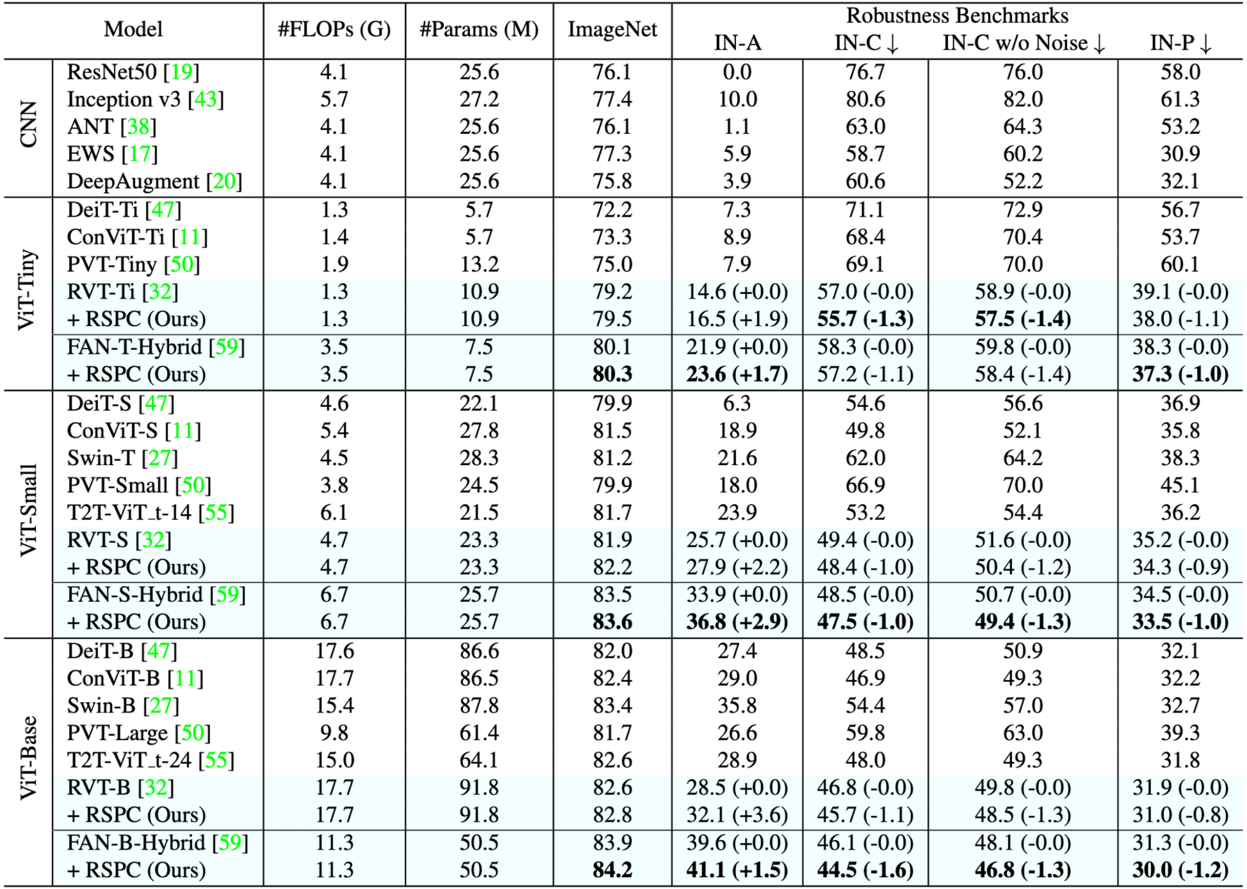


Algorithm 1 Training transformer models by **reducing sensitivity to patch corruptions (RSPC).**

```
Require: Training data $\mathcal{D}$, model parameters $\theta_{\mathcal{C}}$ and $\theta_{\mathcal{F}}$, occlusion ratio $\rho$, step size $\eta$, hyper-parameter $\lambda$.
for each training iteration do
    Sample a data batch $\{x_i\}_{i=1}^{N}$ from $\mathcal{D}$
    // Construct patch-based corruptions $\hat{x}$
    Sample the random noise $\delta$ from a uniform distribution
    Construct $\hat{x}$ using the patch corruption model $\mathcal{C}$:
    $\hat{x} = \mathcal{C}(x;\rho)\cdot x + (1-\mathcal{C}(x;\rho))\cdot\delta$
    // Update the classification model $\mathcal{F}$
    Update $\theta_{\mathcal{F}}$ by descending the gradient:
    $\theta_{\mathcal{F}} = \theta_{\mathcal{F}} - \eta\frac{1}{N}\sum_{i=1}^{N}\nabla_{\theta_{\mathcal{F}}}[\mathcal{L}_{\text{ce}}(x_i) + \lambda\mathcal{L}_{\text{align}}(x_i,\hat{x}_i)]$
    // Update the patch corruption model $\mathcal{C}$
    Update $\theta_{\mathcal{C}}$ by ascending the gradient:
    $\theta_{\mathcal{C}} = \theta_{\mathcal{C}} + \eta\frac{1}{N}\sum_{i=1}^{N}\nabla_{\theta_{\mathcal{C}}}\lambda\mathcal{L}_{\text{align}}(x_i,\hat{x}_i)$
end for
```

Comparisons on ImageNet

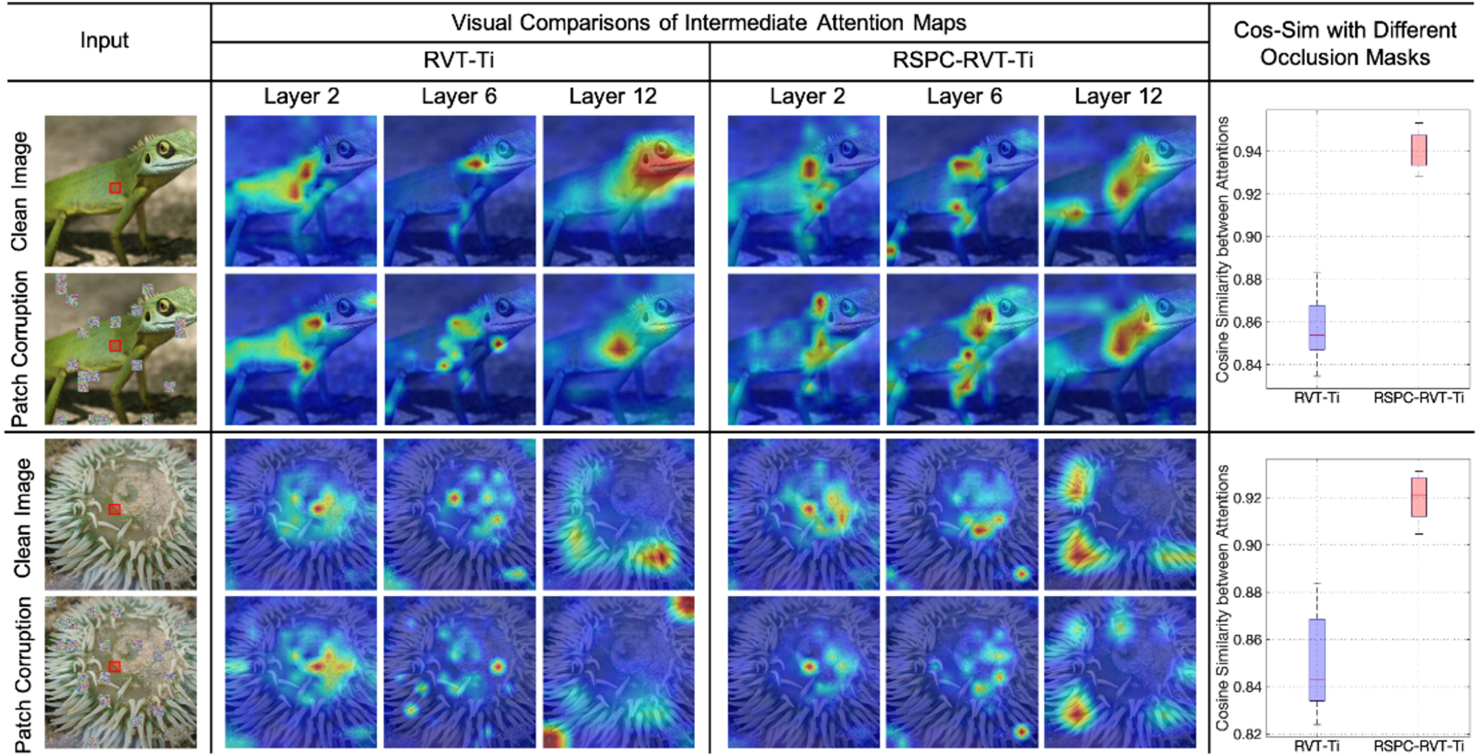
❖ Our RSPC models consistently improve the robustness across different model sizes on ImageNet.



	Model	#FLOPs (G)	#Params (M)	ImageNet	Robustness Benchmarks			
					IN-A	IN-C ↓	IN-C w/o Noise ↓	IN-P ↓
CNN	ResNet50 [19]	4.1	25.6	76.1	0.0	76.7	76.0	58.0
	Inception v3 [43]	5.7	27.2	77.4	10.0	80.6	82.0	61.3
	ANT [38]	4.1	25.6	76.1	1.1	63.0	64.3	53.2
	EWS [17]	4.1	25.6	77.3	5.9	58.7	60.2	30.9
	DeepAugment [20]	4.1	25.6	75.8	3.9	60.6	52.2	32.1
ViT-Tiny	DeiT-Ti [47]	1.3	5.7	72.2	7.3	71.1	72.9	56.7
	ConViT-Ti [11]	1.4	5.7	73.3	8.9	68.4	70.4	53.7
	PVT-Tiny [50]	1.9	13.2	75.0	7.9	69.1	70.0	60.1
	RVT-Ti [32]	1.3	10.9	79.2	14.6 (+0.0)	57.0 (-0.0)	58.9 (-0.0)	39.1 (-0.0)
	+ RSPC (Ours)	1.3	10.9	79.5	16.5 (+1.9)	**55.7 (-1.3)**	**57.5 (-1.4)**	38.0 (-1.1)
	FAN-T-Hybrid [59]	3.5	7.5	80.1	21.9 (+0.0)	58.3 (-0.0)	59.8 (-0.0)	38.3 (-0.0)
	+ RSPC (Ours)	3.5	7.5	**80.3**	**23.6 (+1.7)**	57.2 (-1.1)	58.4 (-1.4)	**37.3 (-1.0)**
ViT-Small	DeiT-S [47]	4.6	22.1	79.9	6.3	54.6	56.6	36.9
	ConViT-S [11]	5.4	27.8	81.5	18.9	49.8	52.1	35.8
	Swin-T [27]	4.5	28.3	81.2	21.6	62.0	64.2	38.3
	PVT-Small [50]	3.8	24.5	79.9	18.0	66.9	70.0	45.1
	T2T-ViT$_t$-14 [55]	6.1	21.5	81.7	23.9	53.2	54.4	36.2
	RVT-S [32]	4.7	23.3	81.9	25.7 (+0.0)	49.4 (-0.0)	51.6 (-0.0)	35.2 (-0.0)
	+ RSPC (Ours)	4.7	23.3	82.2	27.9 (+2.2)	48.4 (-1.0)	50.4 (-1.2)	34.3 (-0.9)
	FAN-S-Hybrid [59]	6.7	25.7	83.5	33.9 (+0.0)	48.5 (-0.0)	50.7 (-0.0)	34.5 (-0.0)
	+ RSPC (Ours)	6.7	25.7	**83.6**	**36.8 (+2.9)**	**47.5 (-1.0)**	**49.4 (-1.3)**	**33.5 (-1.0)**
ViT-Base	DeiT-B [47]	17.6	86.6	82.0	27.4	48.5	50.9	32.1
	ConViT-B [11]	17.7	86.5	82.4	29.0	46.9	49.3	32.2
	Swin-B [27]	15.4	87.8	83.4	35.8	54.4	57.0	32.7
	PVT-Large [50]	9.8	61.4	81.7	26.6	59.8	63.0	39.3
	T2T-ViT$_t$-24 [55]	15.0	64.1	82.6	28.9	48.0	49.3	31.8
	RVT-B [32]	17.7	91.8	82.6	28.5 (+0.0)	46.8 (-0.0)	49.8 (-0.0)	31.9 (-0.0)
	+ RSPC (Ours)	17.7	91.8	82.8	32.1 (+3.6)	45.7 (-1.1)	48.5 (-1.3)	31.0 (-0.8)
	FAN-B-Hybrid [59]	11.3	50.5	83.9	39.6 (+0.0)	46.1 (-0.0)	48.1 (-0.0)	31.3 (-0.0)
	+ RSPC (Ours)	11.3	50.5	**84.2**	**41.1 (+1.5)**	**44.5 (-1.6)**	**46.8 (-1.3)**	**30.0 (-1.2)**

Stability of Intermediate Attention Maps

❖ Our RSPC models obtain much more stable attention maps when facing patch corruptions.



Thanks for your attention !