Boosting Adversarial Transferability by Achieving Flat Local Maxima

Zhijin Ge[1], Hongying Liu[2], Xiaosen Wang[3], Fanhua Shang[2], Yuanyuan Liu[1]

[1]Xidian University [2]Tianjin University [3]Huawei Singular Security Lab

Motivation

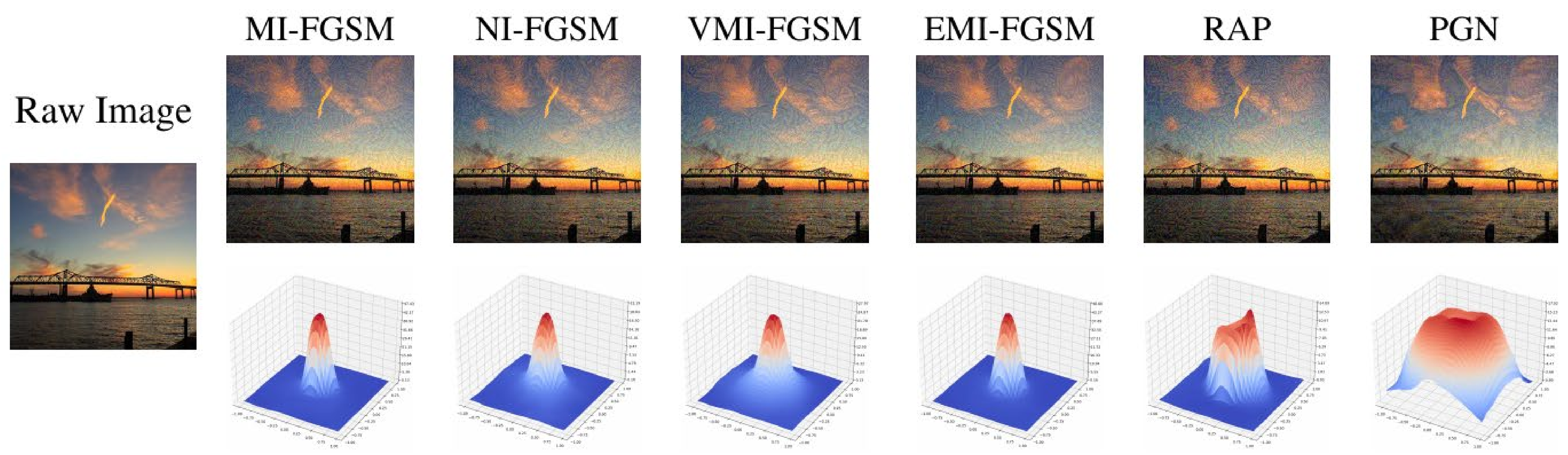


Figure 1: Adversarial examples generated by different methods are located in different regions on the surface of the loss function.

- Inspired by the observation that flat local minima are correlated with good generalization in deep learning, we are motivated to explore whether flat local optima can enhance adversarial transferability.

Contributions

- To the best of our knowledge, it is the first work that empirically validates that adversarial examples located in flat regions have good transferability.
- We propose a novel attack called Penalizing Gradient Norm (PGN), which can effectively generate adversarial examples at flat local regions with better transferability.
- Empirical evaluations show that PGN can significantly improve the attack transferability on both normally trained models and adversarially trained models, which can also be seamlessly combined with various previous attack methods for higher transferability.

Our Method

Assumption: Adversarial examples at flat local region w.r.t. the loss function tends to have better transferability.

- **Optimization problem:**

$$\max_{x^{adv}\in\mathcal{B}_\epsilon(x)}\left[J(x^{adv},y;\theta)-\lambda\cdot\max_{x'\in\mathcal{B}_\zeta(x^{adv})}\|\nabla_{x'}J(x',y;\theta)\|_2\right].$$

- **Empirical validation:**

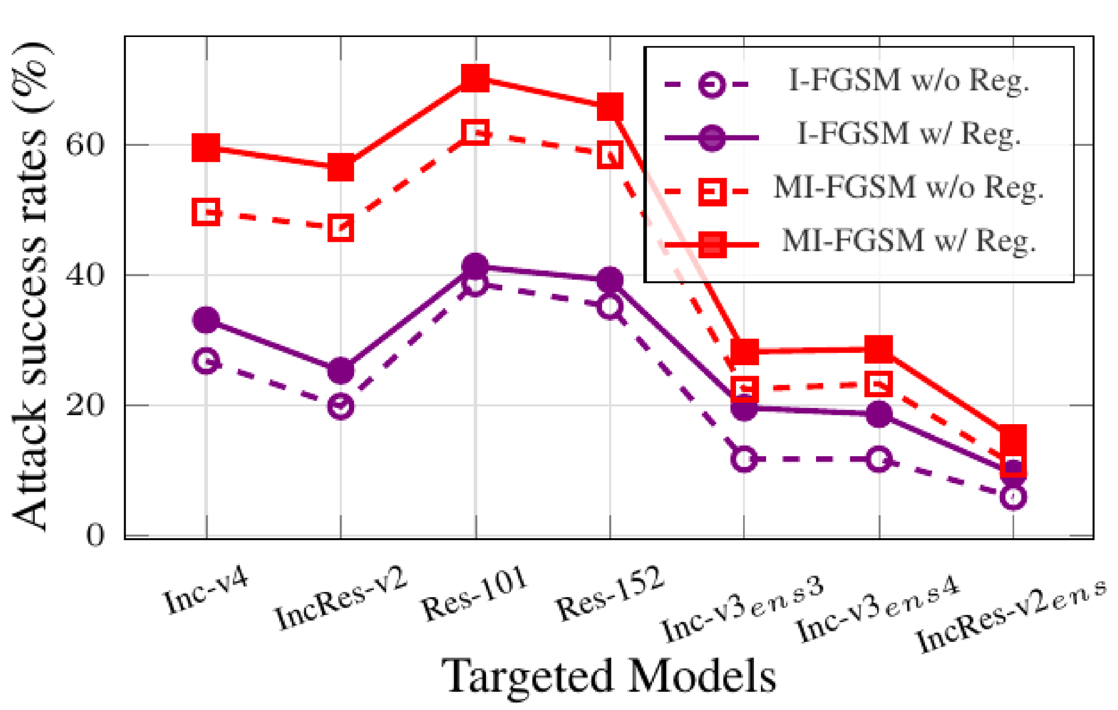


Figure 2: The average attack success rates (%) of I-FGSM and MI-FGSM w/wo the gradient regularization on seven black-box models. The adversarial examples are generated on Inc-v3.

- **Approximate solution:**

$$\max_{x^{adv}\in\mathcal{B}_\epsilon(x)}\mathcal{L}(x^{adv},y;\theta)\approx J(x',y;\theta)-\lambda\cdot\|\nabla_{x'}J(x',y;\theta)\|_2,\quad \text{s.t.}\quad x'\in\mathcal{B}_\zeta(x^{adv}).$$

$$\nabla_{x^{adv}}\mathcal{L}(x^{adv},y;\theta)\approx\nabla_{x'}J(x',y;\theta)-\lambda\cdot\nabla^2_{x'}J(x',y;\theta)\cdot\frac{\nabla_{x'}J(x',y;\theta)}{\|\nabla_{x'}J(x',y;\theta)\|_2}.$$

- **Finite Difference Method:**

$$\nabla^2_x J(x,y;\theta)\cdot v\approx\frac{\nabla_x J(x+\alpha\cdot v,y;\theta)-\nabla_x J(x,y;\theta)}{\alpha},\ v=-\frac{\nabla_x J(x,y;\theta)}{\|\nabla_x J(x,y;\theta)\|_2}.$$

- **Gradient update:**

$$\nabla_{x_t^{adv}}\mathcal{L}(x_t^{adv},y;\theta)\approx(1-\delta)\cdot\nabla_{x_t'}J(x_t',y;\theta)+\delta\cdot\nabla_{x_t'}J(x_t'+\alpha\cdot v,y;\theta),\quad \delta=\frac{\lambda}{\alpha}.$$

- **Generate adversarial examples:**

$$\bar{g}=\frac{1}{N}\cdot\sum_{i=0}^{N}\left[(1-\delta)\cdot\nabla_{x_t'}J(x_t',y;\theta)+\delta\cdot\nabla_{x_t'}J(x_t'+\alpha\cdot v,y;\theta)\right].$$

$$g_{t+1}=\mu\cdot g_t+\frac{\bar{g}}{\|\bar{g}\|_1},\quad g_0=0.$$

$$x_{t+1}^{adv}=\Pi_{\mathcal{B}_\epsilon(x)}\left[x_t^{adv}+\alpha\cdot\text{sign}(g_{t+1})\right],\quad x_0^{adv}=x.$$

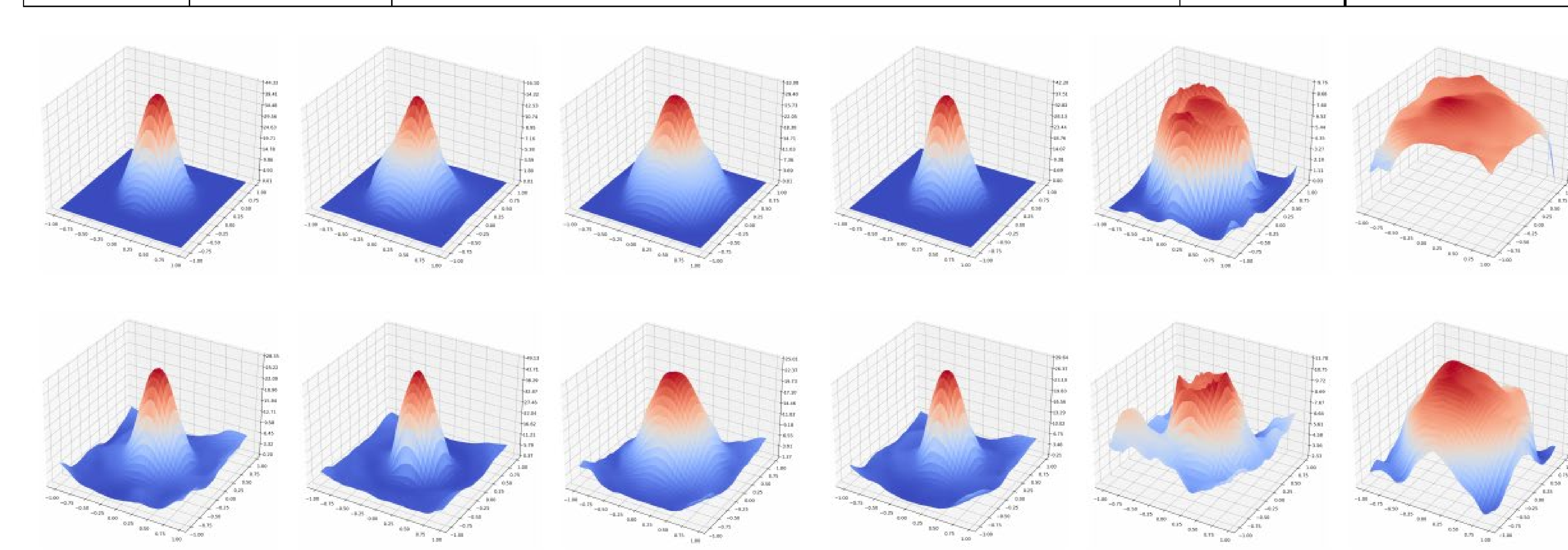
Experimental Results

Table 1: The untargeted attack success rates (%) of various gradient-based attacks in the single model setting. Here * indicates the white-box model.

Model	Attack	Inc-v3	Inc-v4	IncRes-v2	Res-101	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$
Inc-v3	MI	**100.0±0.0***	51.0±0.47	45.8±0.60	49.0±0.24	22.6±0.52	22.0±0.35	10.9±0.24
	NI	**100.0±0.0***	61.4±0.42	59.6±0.54	57.2±0.18	22.5±0.37	22.7±0.35	11.5±0.26
	VMI	**100.0±0.0***	74.8±0.58	69.9±0.92	65.5±0.67	41.6±0.54	41.6±0.54	25.0±0.34
	EMI	**100.0±0.0***	80.7±0.58	77.1±0.37	72.4±0.83	33.0±0.61	31.9±0.49	17.0±0.48
	RAP	99.9±0.10*	84.5±0.69	79.3±0.47	76.5±0.65	56.9±0.84	51.3±0.62	31.9±0.35
	PGN	**100.0±0.0***	**90.6±0.67**	**89.5±0.75**	**81.2±0.68**	**64.6±0.75**	**65.6±0.94**	**45.3±0.77**
Inc-v4	MI	57.2±0.36	**100.0±0.0***	46.1±0.14	51.5±0.33	19.1±0.46	18.4±0.23	10.2±0.36
	NI	62.8±0.43	**100.0±0.0***	52.7±0.34	56.7±0.19	19.2±0.25	18.3±0.37	11.7±0.29
	VMI	77.6±0.65	99.8±0.10*	69.8±0.41	66.7±0.33	41.1±0.87	41.2±0.54	27.0±0.24
	EMI	84.2±0.62	99.7±0.10*	75.0±0.70	74.4±0.64	31.5±0.44	28.0±0.65	16.2±0.36
	RAP	85.3±0.74	99.5±0.21*	79.5±0.62	77.2±0.42	45.2±0.69	46.8±0.48	29.3±0.51
	PGN	**91.2±0.58**	99.6±0.15*	**87.6±0.74**	**83.5±0.53**	**67.0±0.68**	**64.2±0.63**	**49.1±0.82**
IncRes-v2	MI	58.2±0.21	52.4±0.41	99.3±0.21*	50.7±0.26	22.0±0.37	22.0±0.31	13.8±0.43
	NI	60.3±0.35	57.1±0.17	99.5±0.17*	55.3±0.35	18.3±0.18	19.3±0.29	12.1±0.16
	VMI	78.2±0.64	77.0±0.57	99.1±0.36*	66.0±0.48	47.6±0.69	43.3±0.36	37.7±0.37
	EMI	85.2±0.78	83.3±0.29	99.7±0.18*	74.0±0.56	38.4±0.48	33.8±0.53	24.1±0.48
	RAP	87.1±0.75	84.2±0.45	99.4±0.28*	79.4±0.64	50.3±0.47	49.8±0.89	40.2±0.54
	PGN	**92.0±0.69**	**92.3±0.63**	**99.8±0.10***	**83.5±0.41**	**74.6±0.75**	**71.5±0.64**	**66.62±0.58**
Res-101	MI	51.5±0.26	42.2±0.35	36.3±0.24	**100.0±0.0***	18.7±0.32	16.6±0.14	9.0±0.22
	NI	55.6±0.35	46.9±0.41	40.8±0.28	**100.0±0.0***	17.5±0.57	17.6±0.42	9.2±0.24
	VMI	75.0±0.40	69.2±0.59	63.0±0.84	**100.0±0.0***	35.9±0.41	35.7±0.87	24.1±0.57
	EMI	74.3±0.65	71.7±0.47	62.6±0.29	**100.0±0.0***	25.7±0.74	24.6±0.98	13.3±0.68
	RAP	80.4±0.75	75.5±0.56	68.0±0.84	99.9±0.10*	40.3±0.47	39.9±0.73	30.4±1.03
	PGN	**86.2±0.84**	**83.3±0.66**	**77.8±0.69**	**100.0±0.0***	**63.1±1.32**	**62.9±0.74**	**50.8±0.88**

Table 2: Comparison of the approximation effect between directly optimizing the second-order Hessian matrix and using the Finite Difference Method (FDM) to approximate. "Time" represents the total running time on 1,000 images, and "Memory" represents the computing memory size.

Attack	H_m	FDM	Inc-v3	Inc-v4	IncRes-v2	Res-101	Res-152	Time (s)	Memory (MiB)
I-FGSM	✗	✗	**100.0***	27.8	19.1	38.1	35.2	52.31	1631
	✓	✗	**100.0***	**39.2**	**30.2**	**47.0**	**45.5**	**469.54**	**7887**
	✓	✓	**100.0***	37.9	28.6	45.7	44.6	96.42	1631



(a) MI-FGSM (b) NI-FGSM (c) VMI-FGSM (d) EMI-FGSM (e) RAP (f) PGN

Figure 3: Visualization of loss surfaces along two random directions for two randomly sampled adversarial examples on the surrogate model (Inc-v3).