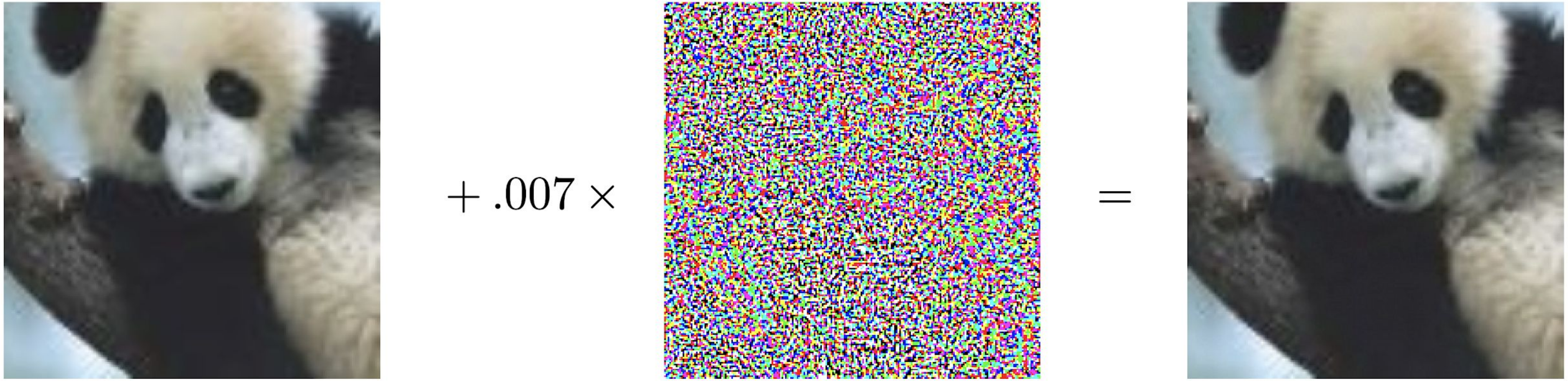
NEURAL INFORMATION PROCESSING SYSTEMS

Adversarial Attack Generation Empowered by Min-Max Optimization

Jingkang Wang*[1,2], Tianyun Zhang*[3], Sijia Liu[4,5], Pin-Yu Chen[5], Jiacen Xu[6], Makan Fardad[7], Bo Li[8]

University of Toronto[1], Vector Institute[2], Cleveland State University[3]
Michigan State University[4], MIT-IBM Watson AI Lab, IBM Research[5]
University of California, Irvine[6], Syracuse University[7]
University of Illinois at Urbana-Champaign[8]

Neural networks are susceptible to adversarial attacks



Classified as
Panda

Imperceptible
Perturbation

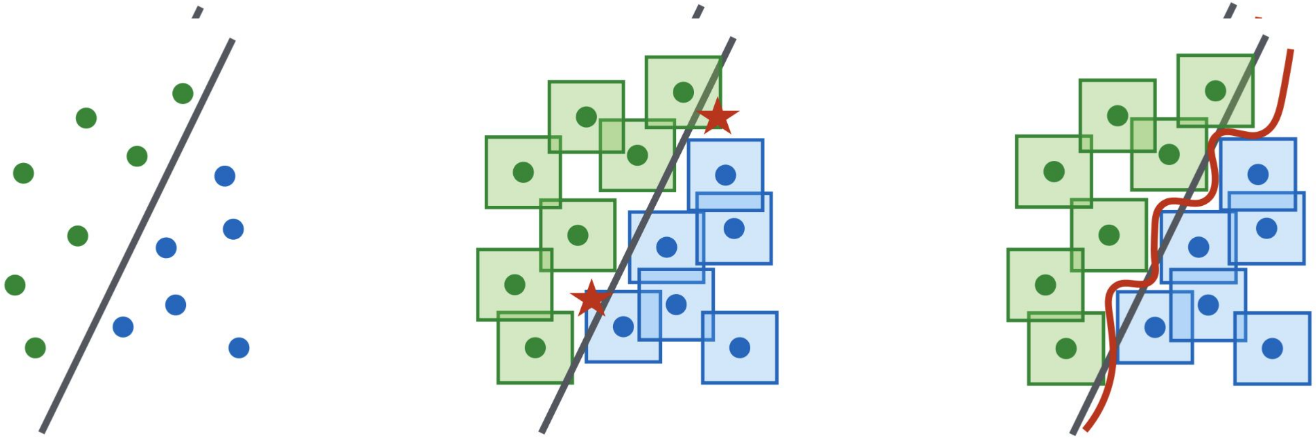
Classified as
Gibbon

Image: Goodfellow et al., Explaining and harnessing adversarial examples, ICLR 2015

Adversarial training: worst-case training principle

- Adversarial training (Madry et al, 2018):

$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$



- *Beyond adversarial training, can other types of **min-max** formulation and optimization techniques advance the research in adversarial attack generation?*

Min-Max Across Domains: Robust Optimization

- Robust optimization over K risk domains (optimize the worst-case performance):

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{i\in[K]}{\text{maximize}}\quad F_i(\mathbf{v})$$

Qi Qian, Shenghuo Zhu, Jiasheng Tang, Rong Jin, Baigui Sun, Hao Li. Robust Optimization over Multiple Domains, 2019.

Min-Max Across Domains: Robust Optimization

- Robust optimization over K risk domains (optimize the worst-case performance):

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{i\in[K]}{\text{maximize}}\quad F_i(\mathbf{v})$$

Equivalent to

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{\mathbf{w}\in\mathcal{P}}{\text{maximize}}\quad \sum_{i=1}^{K} w_i F_i(\mathbf{v})$$

where $\mathcal{P} = \{\mathbf{w} \,|\, \mathbf{1}^T\mathbf{w} = 1, w_i \in [0,1], \forall i\}$

Qi Qian, Shenghuo Zhu, Jiasheng Tang, Rong Jin, Baigui Sun, Hao Li. Robust Optimization over Multiple Domains, 2019.

Min-Max Across Domains: Robust Optimization

- Robust optimization over K risk domains (optimize the worst-case performance):

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{i\in[K]}{\text{maximize}}\quad F_i(\mathbf{v})$$

Equivalent to

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{\mathbf{w}\in\mathcal{P}}{\text{maximize}}\quad \sum_{i=1}^{K} w_i F_i(\mathbf{v})$$

where $\mathcal{P} = \{\mathbf{w} \,|\, \mathbf{1}^T\mathbf{w} = 1, w_i \in [0,1], \forall i\}$

reduces the generalizability to other domains and induces instability of the learning procedure

Qi Qian, Shenghuo Zhu, Jiasheng Tang, Rong Jin, Baigui Sun, Hao Li. Robust Optimization over Multiple Domains, 2019.

Min-Max Across Domains: Robust Optimization

- Robust optimization over K risk domains (optimize the worst-case performance):

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{i\in[K]}{\text{maximize}}\quad F_i(\mathbf{v})$$

Equivalent to

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{\mathbf{w}\in\mathcal{P}}{\text{maximize}}\quad \sum_{i=1}^{K} w_i F_i(\mathbf{v})$$

where $\mathcal{P} = \{\mathbf{w}\,|\,\mathbf{1}^T\mathbf{w} = 1, w_i \in [0,1], \forall i\}$

reduces the generalizability to other domains and induces instability of the learning procedure

- Regularized Formulation (strike a balance between the average and the worst-case performance):

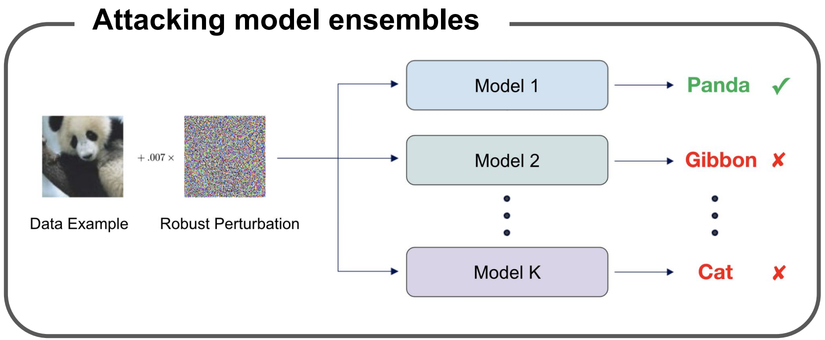
$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{\mathbf{w}\in\mathcal{P}}{\text{maximize}}\quad \sum_{i=1}^{K} w_i F_i(\mathbf{v}) - \frac{\gamma}{2}\|\mathbf{w} - \mathbf{1}/K\|_2^2$$

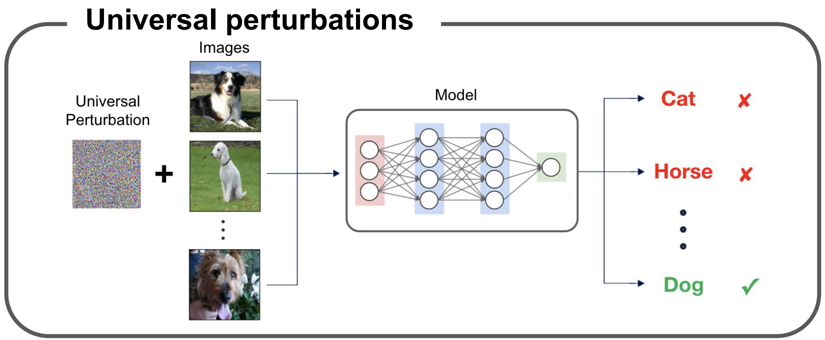
Domain weights (for w_i) **strongly concave regularizer** (for $-\frac{\gamma}{2}\|\mathbf{w} - \mathbf{1}/K\|_2^2$)

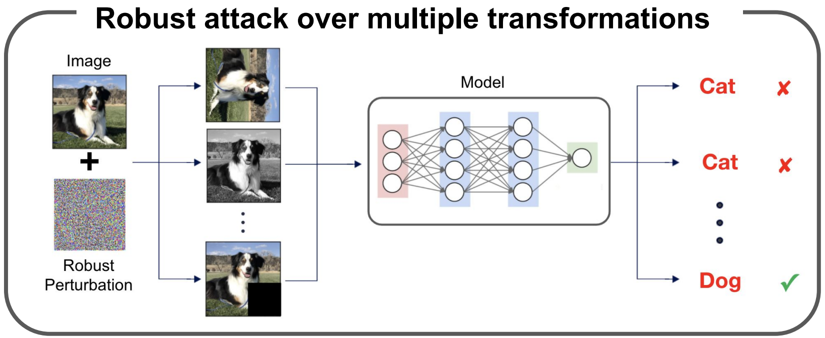
Qi Qian, Shenghuo Zhu, Jiasheng Tang, Rong Jin, Baigui Sun, Hao Li. Robust Optimization over Multiple Domains, 2019.

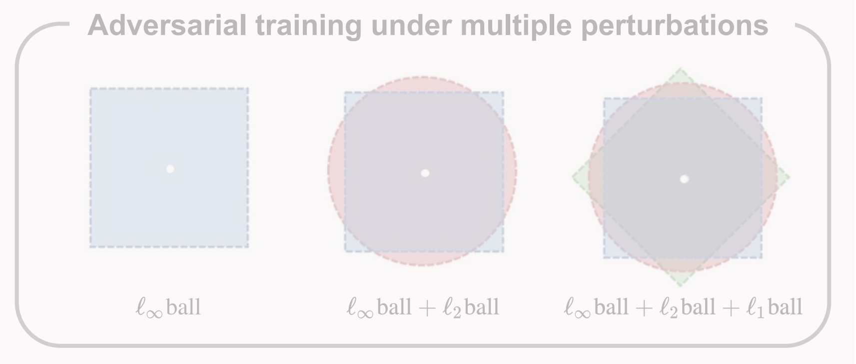
Min-Max Power in Attack Design

- The unified min-max framework actually fits into **various attack settings**!







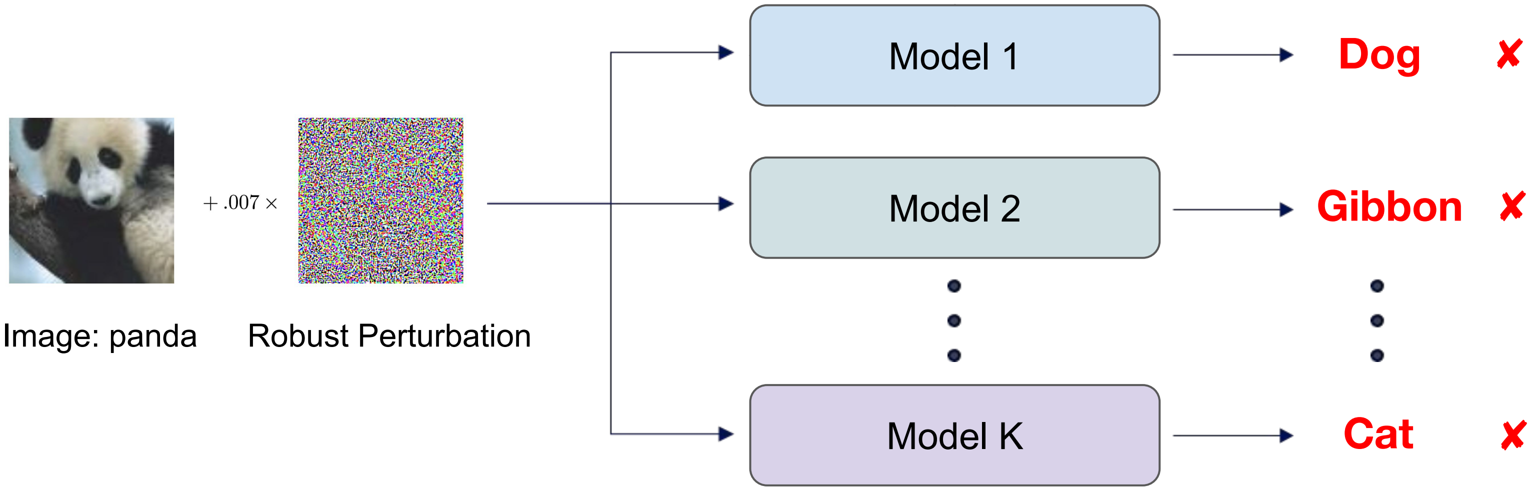


Ensemble Attack over Multiple Models

- Consider K ML/DL models $\{\mathcal{M}_i\}_{i=1}^K$, the goal is to find robust adversarial examples that can fool all K models simultaneously

$$\underset{\boldsymbol{\delta} \in \mathcal{X}}{\text{minimize}} \ \underset{\mathbf{w} \in \mathcal{P}}{\text{maximize}} \quad \sum_{i=1}^K w_i f(\boldsymbol{\delta}; \mathbf{x}_0, y_0, \mathcal{M}_i) - \frac{\gamma}{2} \|\mathbf{w} - \mathbf{1}/K\|_2^2$$

- $\mathbf{w}$ encodes the difficulty level of attacking each model

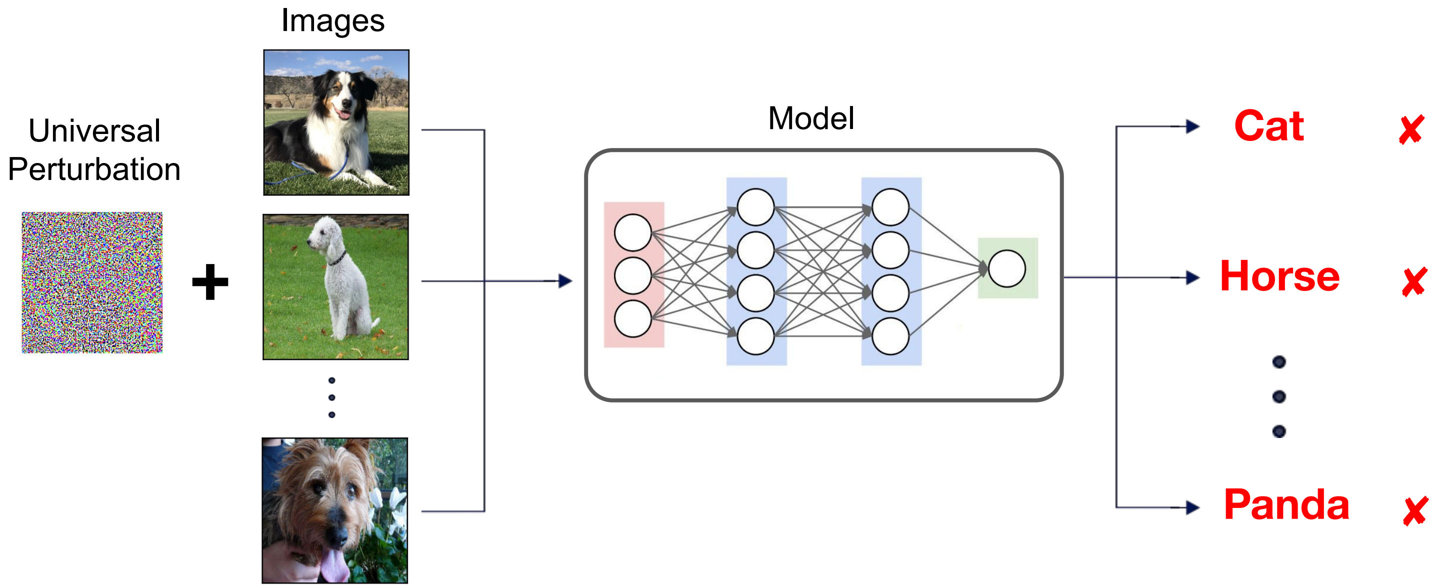


Universal perturbation over multiple examples

- Consider K natural examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{K}$ and a single model $\mathcal{M}$, the goal is to find the universal perturbation δ so that all the corrupted K examples can fool $\mathcal{M}$

$$\underset{\boldsymbol{\delta} \in \mathcal{X}}{\text{minimize}} \ \underset{\mathbf{w} \in \mathcal{P}}{\text{maximize}} \quad \sum_{i=1}^{K} w_i f(\boldsymbol{\delta}; \mathbf{x}_i, y_i, \mathcal{M}) - \frac{\gamma}{2} \|\mathbf{w} - \mathbf{1}/K\|_2^2$$

- $\mathbf{w}$ encodes the difficulty level of attacking each image

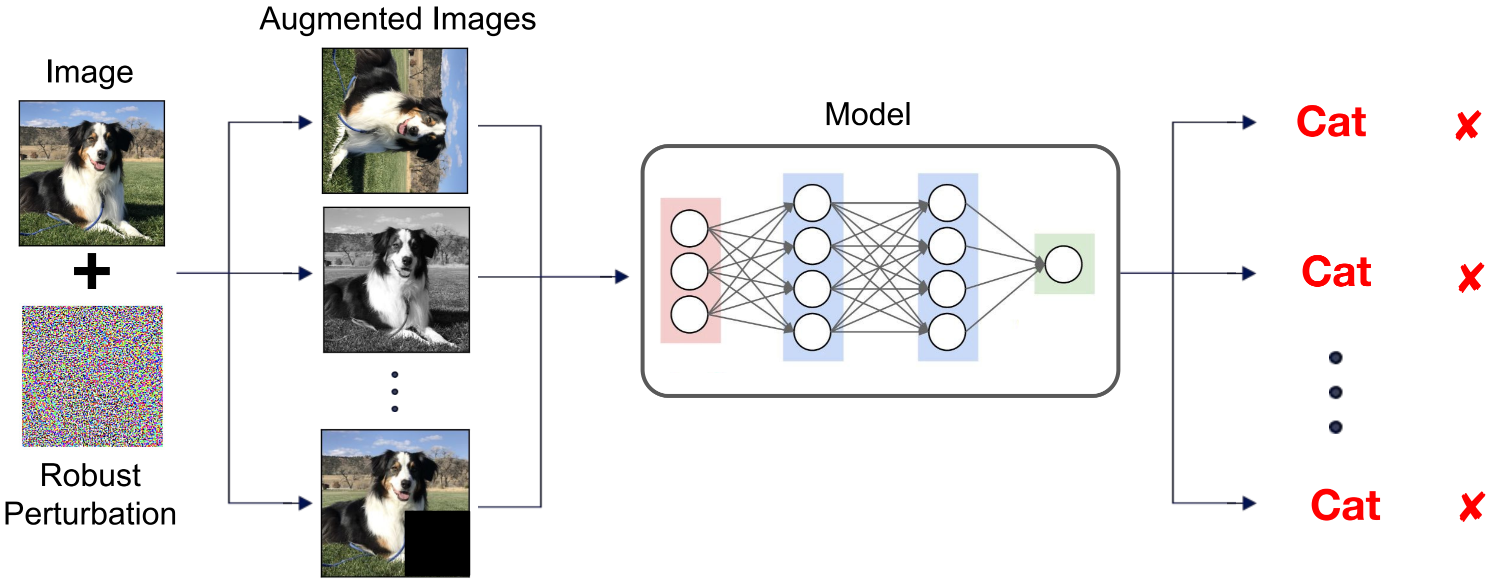


Robust attack over data transformations

- Consider K categories of data transformation $\{p_i\}$ e.g., rotation, lightening, and translation. The goal to find the adversarial attack that is robust to data trans$\mathcal{M}$mations

$$\underset{\boldsymbol{\delta} \in \mathcal{X}}{\text{minimize}} \ \underset{\mathbf{w} \in \mathcal{P}}{\text{maximize}} \quad \sum_{i=1}^{K} w_i \mathbb{E}_{t \sim p_i} [f(t(\mathbf{x}_0 + \boldsymbol{\delta}); y_0, \mathcal{M})] - \frac{\gamma}{2} \|\mathbf{w} - \mathbf{1}/K\|_2^2$$

- $\mathbf{w}$ encodes the difficulty level of attacking each type of transformed example



Min-Max Algorithm for Adversarial Attack Generation

- **Al**ternating **p**rojected **g**radient **d**escent-**a**scent (APGDA) to solve

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{\mathbf{w}\in\mathcal{P}}{\text{maximize}}\quad \sum_{i=1}^{K} w_i F_i(\mathbf{v}) - \frac{\gamma}{2}\|\mathbf{w} - \mathbf{1}/K\|_2^2$$

- APGDA takes only **one-step PGD** for outer minimization and **one-step projected gradient ascent** for inner maximization

APGDA

Input: given $\mathbf{w}^{(0)}$ and $\boldsymbol{\delta}^{(0)}$.
for $t = 1, 2, \ldots, T$ **do**
outer min.: fixing $\mathbf{w} = \mathbf{w}^{(t-1)}$, update $\boldsymbol{\delta}^{(t)}$ via $\boldsymbol{\delta}^{(t)} = \text{proj}_{\mathcal{V}}\left(\boldsymbol{\delta}^{(t-1)} - \alpha \nabla_{\boldsymbol{\delta}} F(\boldsymbol{\delta}^{(t-1)})\right)$
inner max.: fixing $\boldsymbol{\delta} = \boldsymbol{\delta}^{(t)}$, update $\mathbf{w}^{(t)}$ via $\mathbf{w}^{(t)} = \text{proj}_{\mathcal{P}}\left(\mathbf{w}^{(t-1)} + \beta \nabla_{\mathbf{w}} \psi(\mathbf{w}^{(t-1)})\right)$
end for

⇨

Theorem 1. *Suppose that $F_i(\boldsymbol{\delta})$ has L-Lipschitz continuous gradients, and $\mathcal{V}$ is a convex compact set. Given learning rates $\alpha \leq \frac{1}{L}$ and $\beta < \frac{1}{\gamma}$, then the sequence $\{\boldsymbol{\delta}^{(t)}, \mathbf{w}^{(t)}\}_{t=1}^{T}$ generated by Algorithm 1 converges to a first-order stationary point in rate $\mathcal{O}\left(\frac{1}{T}\right)$.*

Min-Max Algorithm for Adversarial Attack Generation

- **A**lternating **p**rojected **g**radient **d**escent-**a**scent (APGDA) to solve

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{\mathbf{w}\in\mathcal{P}}{\text{maximize}} \quad \sum_{i=1}^{K} w_i F_i(\mathbf{v}) - \frac{\gamma}{2}\|\mathbf{w} - \mathbf{1}/K\|_2^2$$

- APGDA takes only **one-step PGD** for outer minimization and **one-step projected gradient ascent** for inner maximization

APGDA

Input: given $\mathbf{w}^{(0)}$ and $\boldsymbol{\delta}^{(0)}$.
for $t = 1, 2, \ldots, T$ **do**
outer min.: fixing $\mathbf{w} = \mathbf{w}^{(t-1)}$, update $\boldsymbol{\delta}^{(t)}$ via $\boldsymbol{\delta}^{(t)} = \text{proj}_{\mathcal{V}}\left(\boldsymbol{\delta}^{(t-1)} - \alpha\nabla_{\boldsymbol{\delta}}F(\boldsymbol{\delta}^{(t-1)})\right)$
inner max.: fixing $\boldsymbol{\delta} = \boldsymbol{\delta}^{(t)}$, update $\mathbf{w}^{(t)}$ via $\mathbf{w}^{(t)} = \text{proj}_{\mathcal{P}}\left(\mathbf{w}^{(t-1)} + \beta\nabla_{\mathbf{w}}\psi(\mathbf{w}^{(t-1)})\right)$
end for

Theorem 1. *Suppose that $F_i(\boldsymbol{\delta})$ has L-Lipschitz continuous gradients, and $\mathcal{V}$ is a convex compact set. Given learning rates $\alpha \leq \frac{1}{L}$ and $\beta < \frac{1}{\gamma}$, then the sequence $\{\boldsymbol{\delta}^{(t)}, \mathbf{w}^{(t)}\}_{t=1}^{T}$ generated by Algorithm 1 converges to a first-order stationary point in rate $\mathcal{O}\left(\frac{1}{T}\right)$.*

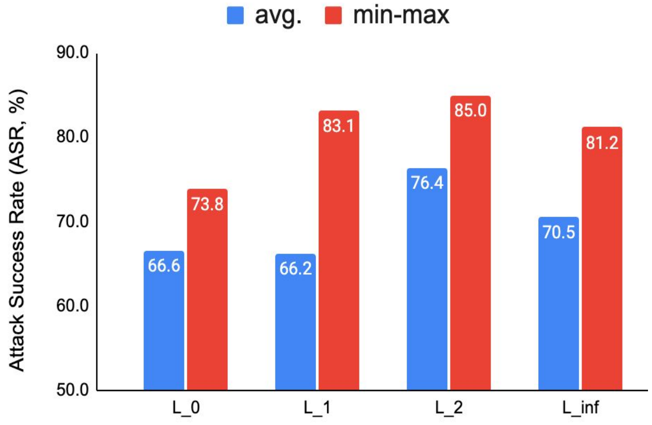
APGDA is efficient!
(linear convergence rate)

AMGDA produces more robust adversarial attacks

- Significant improvements over average strategy on three robust adversarial attacks

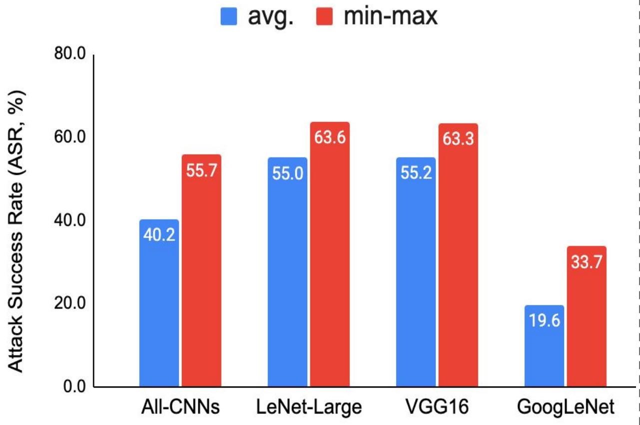
CIFAR-10 (L_inf Attack)

Model sets :
{MLP, VGG16, ResNet50, GoogLeNet}



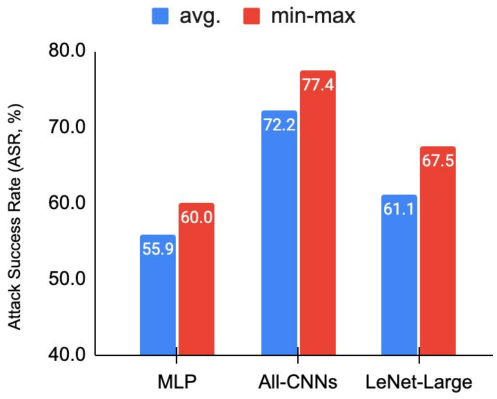
Attacking model ensembles

5 random images in one group (K=5)



Universal perturbations

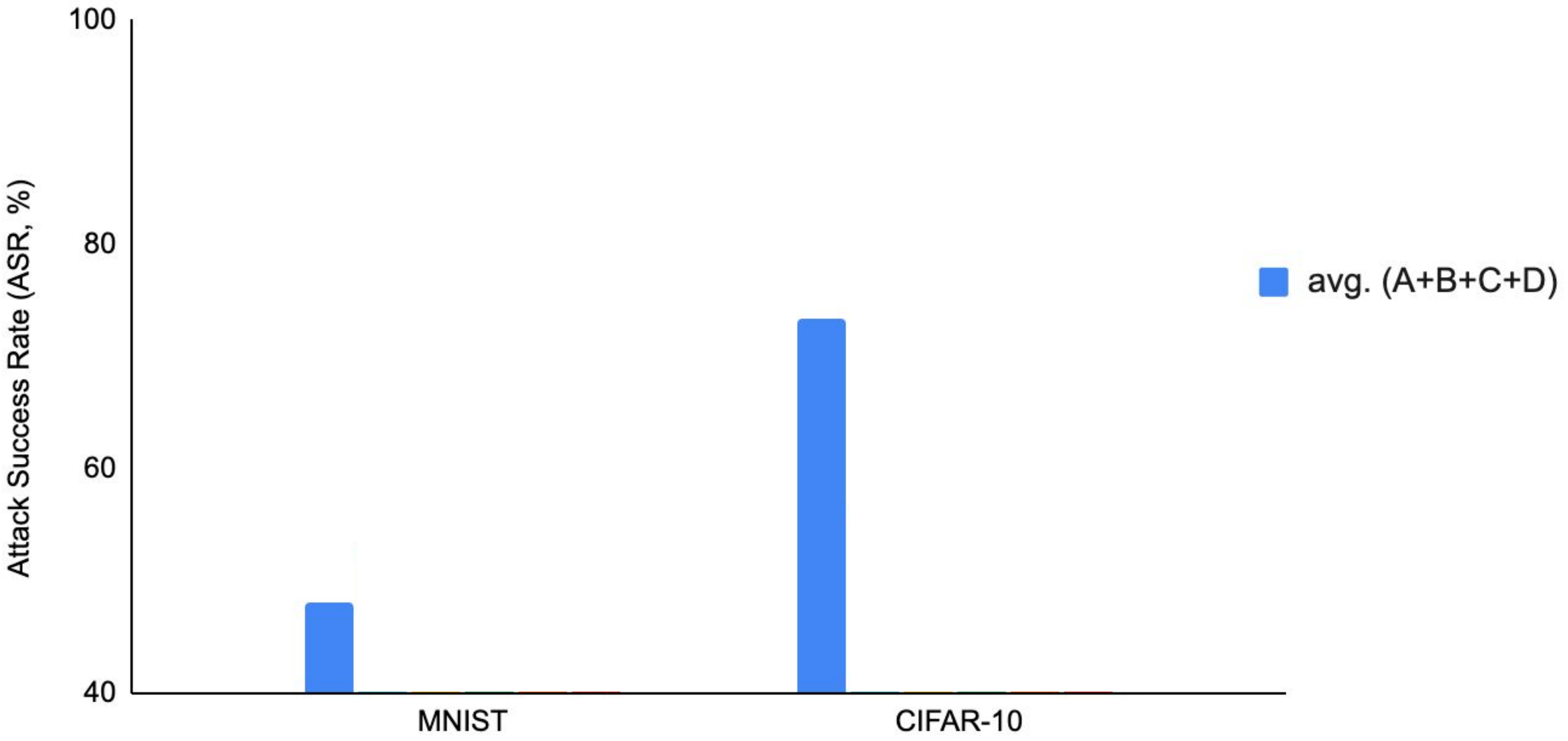
6 data transformations (K=6)



Robust perturbations over data transformations

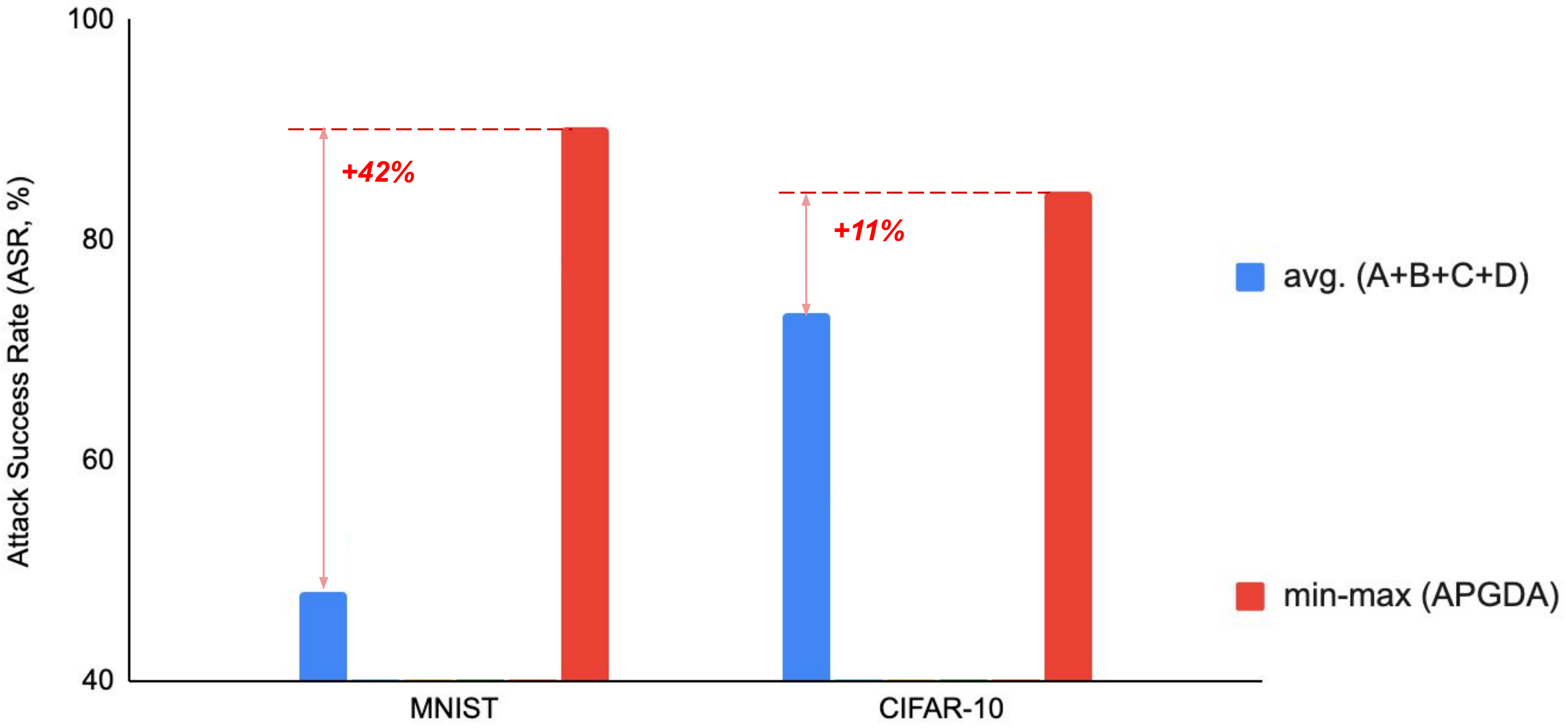
Min-max outperforms heuristic strategies

- ℓ_∞ ensemble attack over four models: Model A (MLP), B (All-CNNs), C (LeNet), D (LeNet-Large)



Min-max outperforms heuristic strategies

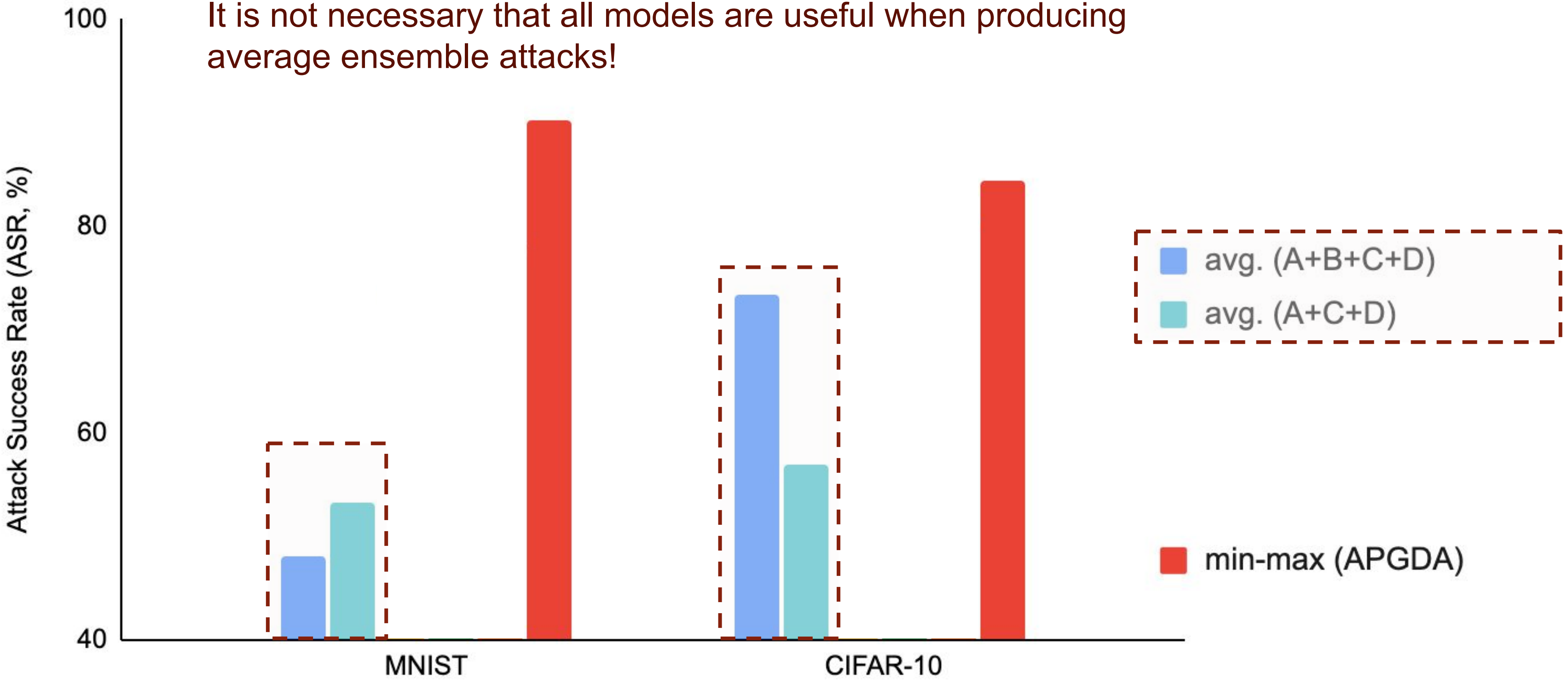
- AMGDA outperforms the average PGD (Liu et al., 2018) by a large margin



Min-max outperforms heuristic strategies

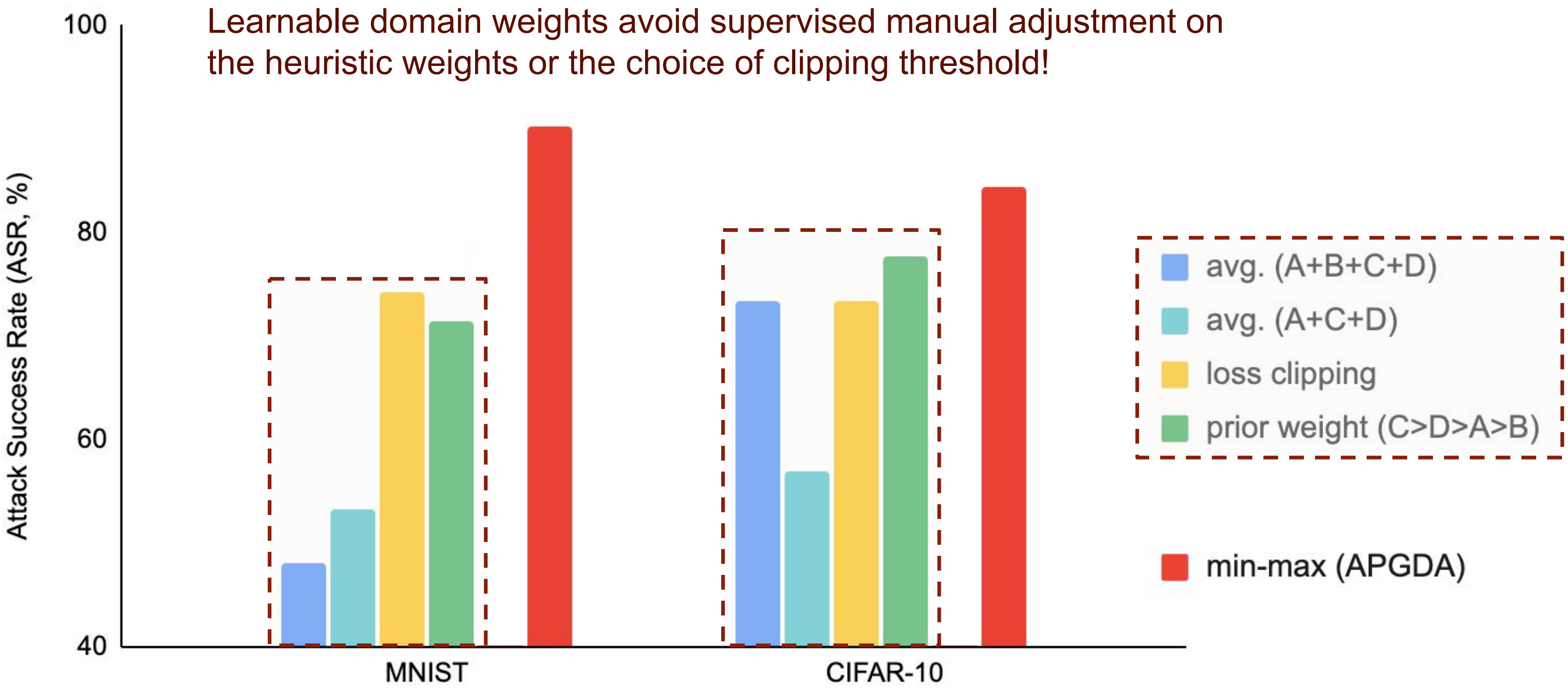
- Robustness of four models (C > D > A > B) ⇐ FGSM Attack $\text{Acc}_C > \text{Acc}_D > \text{Acc}_A > \text{Acc}_B$

It is not necessary that all models are useful when producing average ensemble attacks!



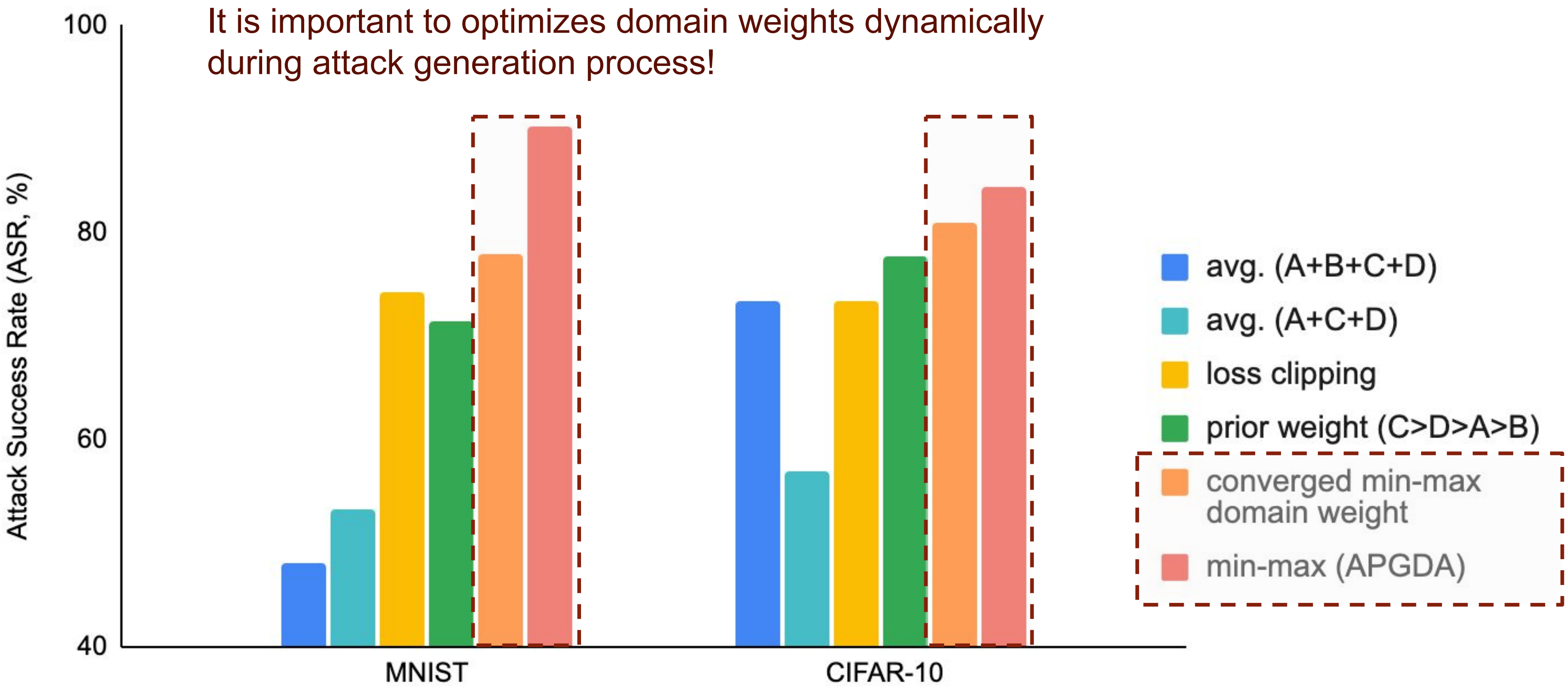
Min-max outperforms heuristic strategies

- With the prior knowledge of robustness (C>D>A>B), we are able to design stronger heuristic strategies!



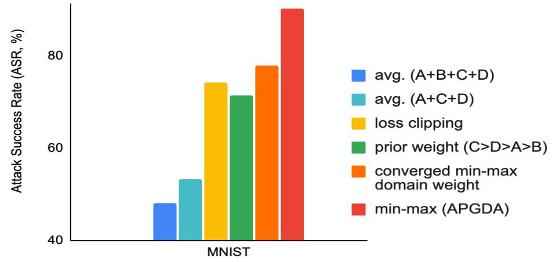
Min-max outperforms heuristic strategies

- Adopting converged min-max weights statically leads to a huge performance drop

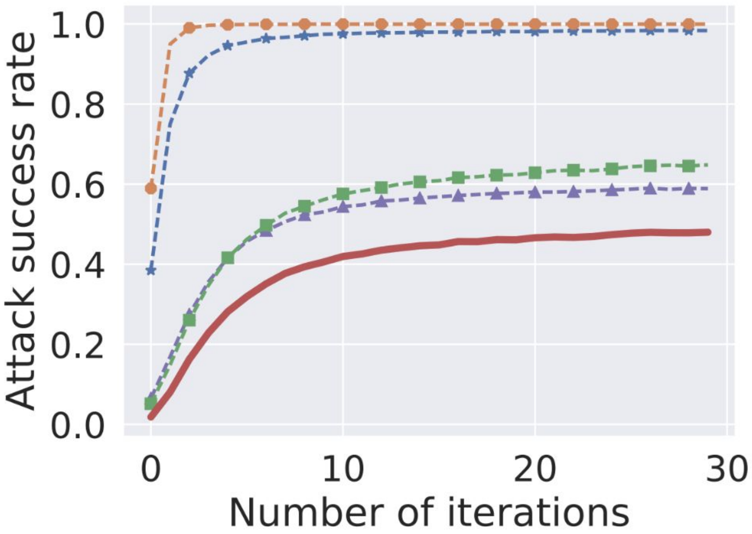


How does APGDA work?

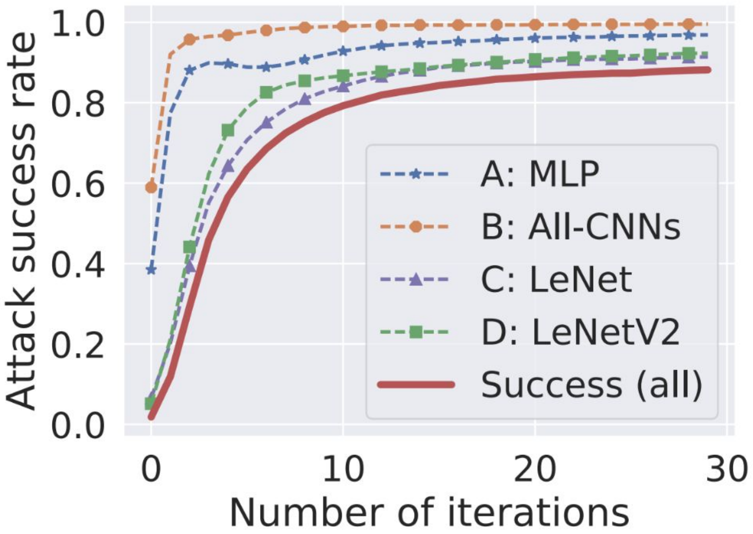
- Robustness of four models (C > D > A > B)
- Model C and D are attacked insufficiently, leading to relatively weak ensemble performance
- APGDA encodes the difficulty level to attack different models based on the current attack loss



avg.

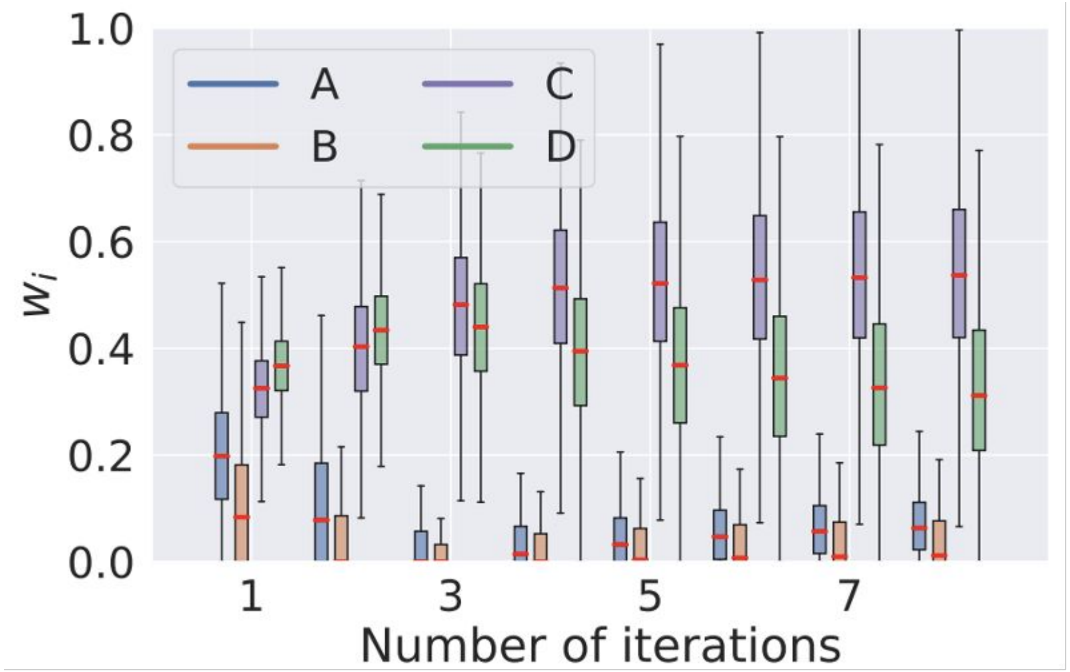


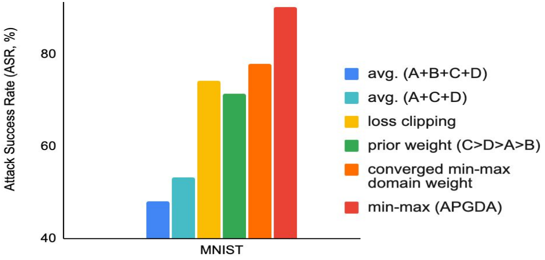
min-max



How does APGDA work?

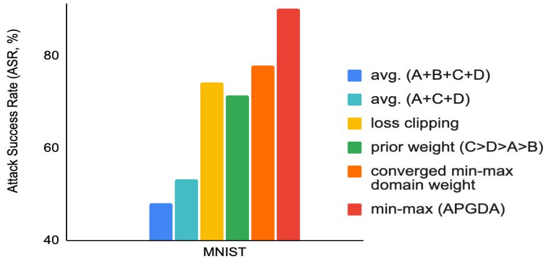
- APGDA dynamically adjusts the domain weights w_i
- w_D first raised to 0.45 then decreased to 0.3
- APGDA is efficient, w_i converges after a small number of iterations

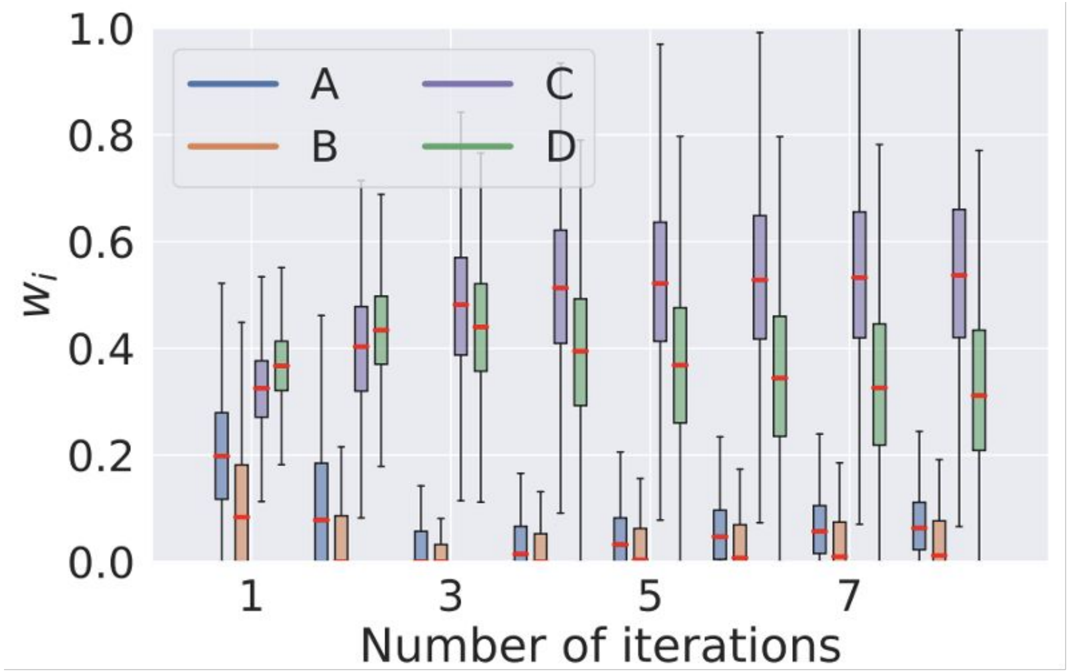




How does APGDA work?

- APGDA dynamically adjusts the domain weights w_i
- w_D first raised to 0.45 then decreased to 0.3
- APGDA is efficient, w_i converges after a small number of iterations





A holistic tool to interpret the risk of different domain sources!

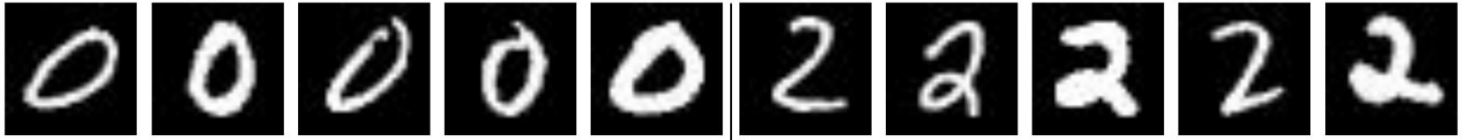
$$w_c > w_d > w_a > w_b$$

$$\text{Acc}_C > \text{Acc}_D > \text{Acc}_A > \text{Acc}_B$$

Interpreting "*image robustness*" with domain weights

- Domain weight w for different images under ℓ_p norm ($p = 0, 1, 2, \infty$)
- Associating domain weights with image visualization

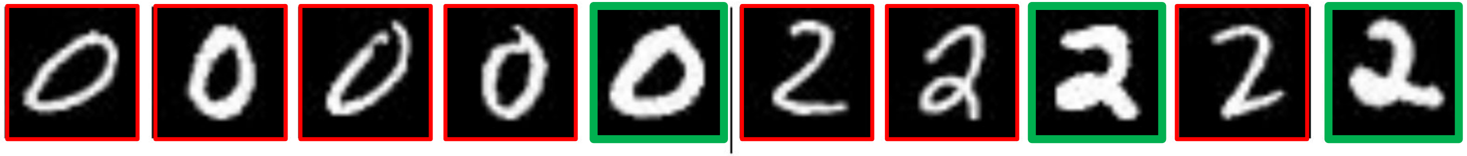
Which images are more robust?

Image			
Weight	ℓ_0 ℓ_1 ℓ_2 ℓ_∞		
Metric	dist.(C&W ℓ_2) $\epsilon_{\min}$ (ℓ_∞)		

Interpreting "*image robustness*" with domain weights

- Domain weight w for different images under ℓ_p norm ($p = 0, 1, 2, \infty$)
- Associating domain weights with image visualization

Which images are more robust?

Image											
Weight	ℓ_0	0.	0.	0.	0.	1.000	0.	0.	0.909	0.	0.091
	ℓ_1	0.	0.	0.	0.	1.000	0.	0.	0.843	0.	0.157
	ℓ_2	0.	0.	0.	0.	1.000	0.	0.	0.788	0.	0.112
	ℓ_∞	0.	0.	0.	0.	1.000	0.	0.	0.850	0.	0.150
Metric	dist.(C&W ℓ_2)										
	$\epsilon_{\min}$ (ℓ_∞)										

Interpreting "*image robustness*" with domain weights

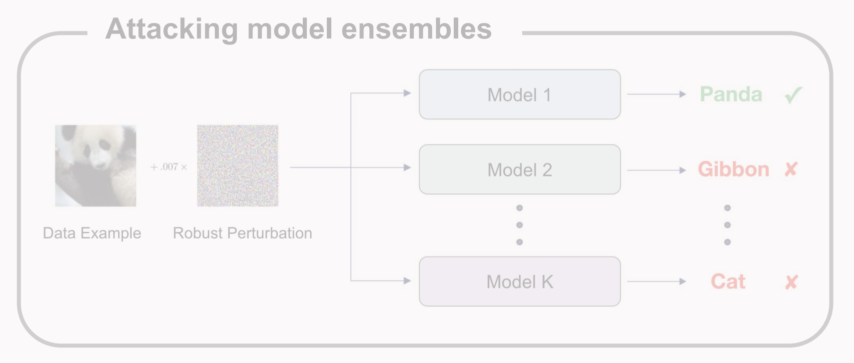
- Domain weight w for different images under ℓ_p norm ($p = 0, 1, 2, \infty$)
- Associating domain weights with image visualization
- **Letters with clear appearance** (e.g., bold letter) ⇔ **larger domain weights**

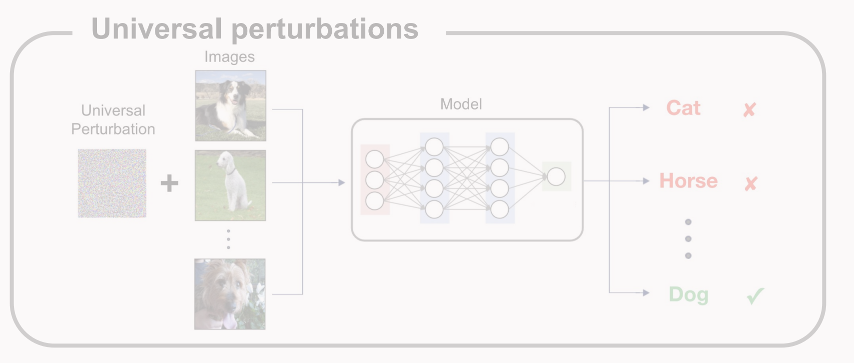
Which images are more robust?

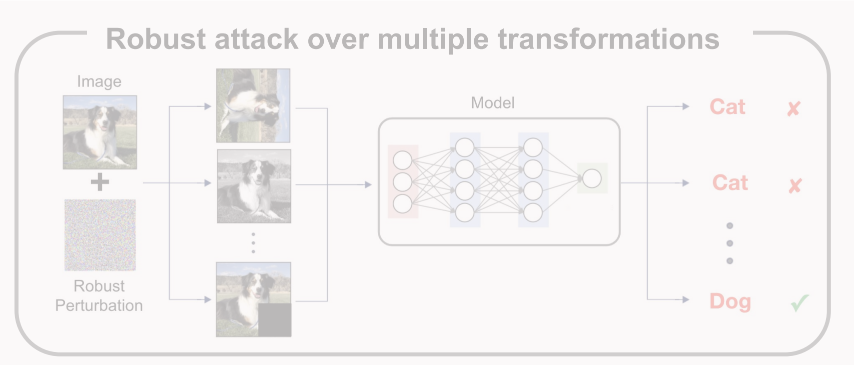
Image											
Weight	ℓ_0	0.	0.	0.	0.	1.000	0.	0.	0.909	0.	0.091
	ℓ_1	0.	0.	0.	0.	1.000	0.	0.	0.843	0.	0.157
	ℓ_2	0.	0.	0.	0.	1.000	0.	0.	0.788	0.	0.112
	ℓ_∞	0.	0.	0.	0.	1.000	0.	0.	0.850	0.	0.150
Metric	dist.(C&W ℓ_2)	1.839	1.954	1.347	1.698	**3.041**	1.928	1.439	2.312	1.521	**2.356**
	$\epsilon_{\min}$ (ℓ_∞)	0.113	0.167	0.073	0.121	**0.199**	0.082	0.106	**0.176**	0.072	0.171

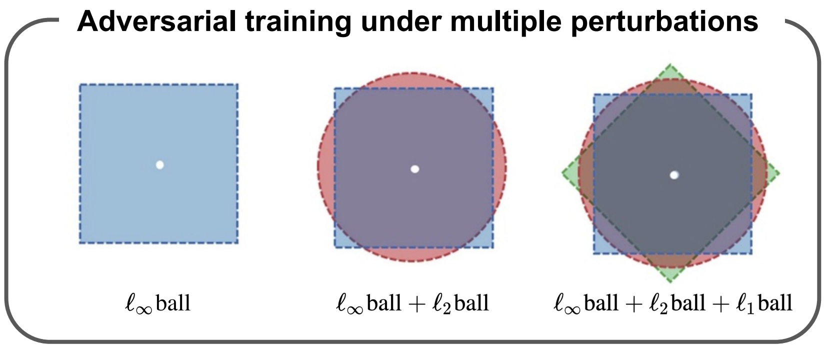
Min-Max Power in Attack Design, and more?

- The unified min-max framework also fits into **defense**!









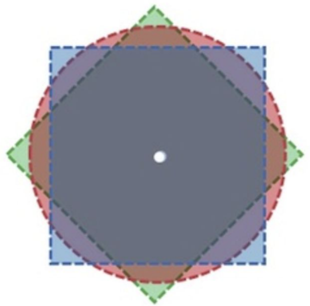
Understanding Defense over Multiple Perturbation Domains

- Conventional adversarial training

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{\|\boldsymbol{\delta}\|_\infty\leq\epsilon}{\text{maximize}}\ f_{\text{tr}}(\boldsymbol{\theta},\boldsymbol{\delta};\mathbf{x},y)$$

  - how to generalize under multiple ℓ_p-norm adversarial attacks?

ℓ_∞ ball + ℓ_2 ball + ℓ_1 ball



[Maini et al., 2020]

Understanding Defense over Multiple Perturbation Domains

- Conventional adversarial training

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{\|\boldsymbol{\delta}\|_\infty\leq\epsilon}{\text{maximize}}\ f_{\text{tr}}(\boldsymbol{\theta},\boldsymbol{\delta};\mathbf{x},y)$$

  - how to generalize under multiple ℓ_p-norm adversarial attacks?

- Treating "attack type" as "risk domain"
  - Defending against the strongest adversarial attack across K attack types in order to avoid blind attacking spots!

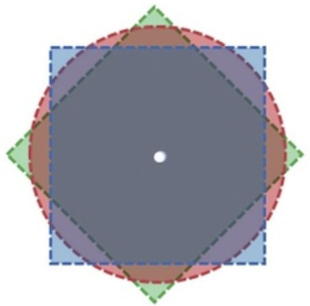
$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{i\in[K]}{\text{maximize}}\ F_i(\boldsymbol{\theta})$$

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{\mathbf{w}\in\mathcal{P},\{\boldsymbol{\delta}_i\in\mathcal{X}_i\}}{\text{maximize}}\ \sum_{i=1}^{K} w_i f_{\text{tr}}(\boldsymbol{\theta},\boldsymbol{\delta}_i;\mathbf{x},y)$$

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{\mathbf{w}\in\mathcal{P},\{\boldsymbol{\delta}_i\in\mathcal{X}_i\}}{\text{maximize}}\ \psi(\boldsymbol{\theta},\mathbf{w},\{\boldsymbol{\delta}_i\})$$

$$\psi(\boldsymbol{\theta},\mathbf{w},\{\boldsymbol{\delta}_i\}) := \sum_{i=1}^{K} w_i f_{\text{tr}}(\boldsymbol{\theta},\boldsymbol{\delta}_i;\mathbf{x},y) - \frac{\gamma}{2}\|\mathbf{w}-\mathbf{1}/K\|_2^2$$

ℓ_∞ ball + ℓ_2 ball + ℓ_1 ball



[Maini et al., 2020]

Understanding Defense over Multiple Perturbation Domains

- **A**lternating **m**ulti-step **p**rojected **g**radient **d**escent (AMPGD) to solve

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{\mathbf{w}\in\mathcal{P},\{\boldsymbol{\delta}_i\in\mathcal{X}_i\}}{\text{maximize}}\ \psi(\boldsymbol{\theta},\mathbf{w},\{\boldsymbol{\delta}_i\})$$

$$\psi(\boldsymbol{\theta},\mathbf{w},\{\boldsymbol{\delta}_i\}) := \sum_{i=1}^{K} w_i f_{\mathrm{tr}}(\boldsymbol{\theta},\boldsymbol{\delta}_i;\mathbf{x},y) - \frac{\gamma}{2}\|\mathbf{w}-\mathbf{1}/K\|_2^2$$

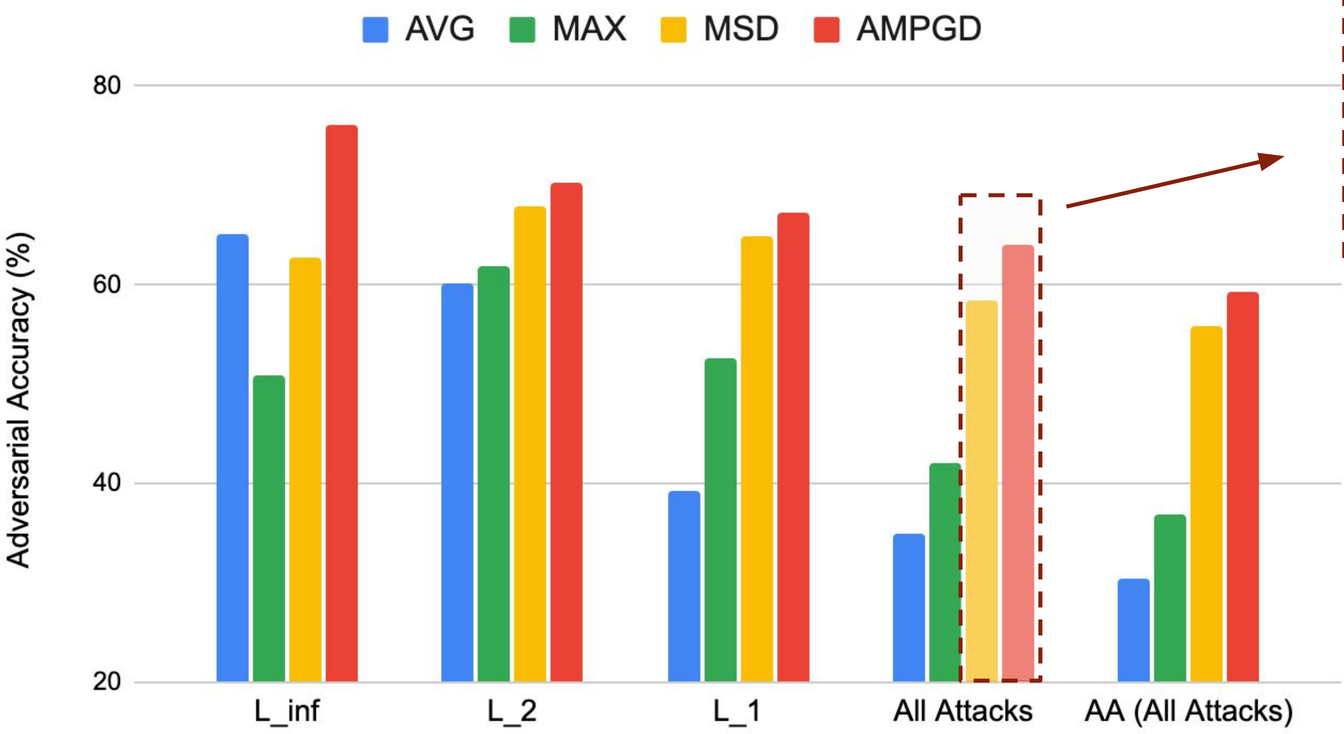
- AMPGD performs SGD for outer minimization and **multi-step PGD** for inner maximization (update perturbation and domain weights)

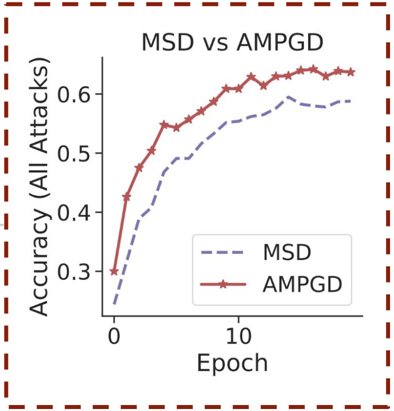
AMPGD

Input: given $\boldsymbol{\theta}^{(0)}$, $\mathbf{w}^{(0)}$, $\boldsymbol{\delta}^{(0)}$ and $K > 0$.
for $t = 1, 2, \ldots, T$ **do**
  given $\mathbf{w}^{(t-1)}$ and $\boldsymbol{\delta}^{(t-1)}$, perform SGD to update $\boldsymbol{\theta}^{(t)}$
  given $\boldsymbol{\theta}^{(t)}$, perform R-step PGD to update $\mathbf{w}^{(t)}$ and $\boldsymbol{\delta}^{(t)}$
end for

AMPGD improves over previous baselines

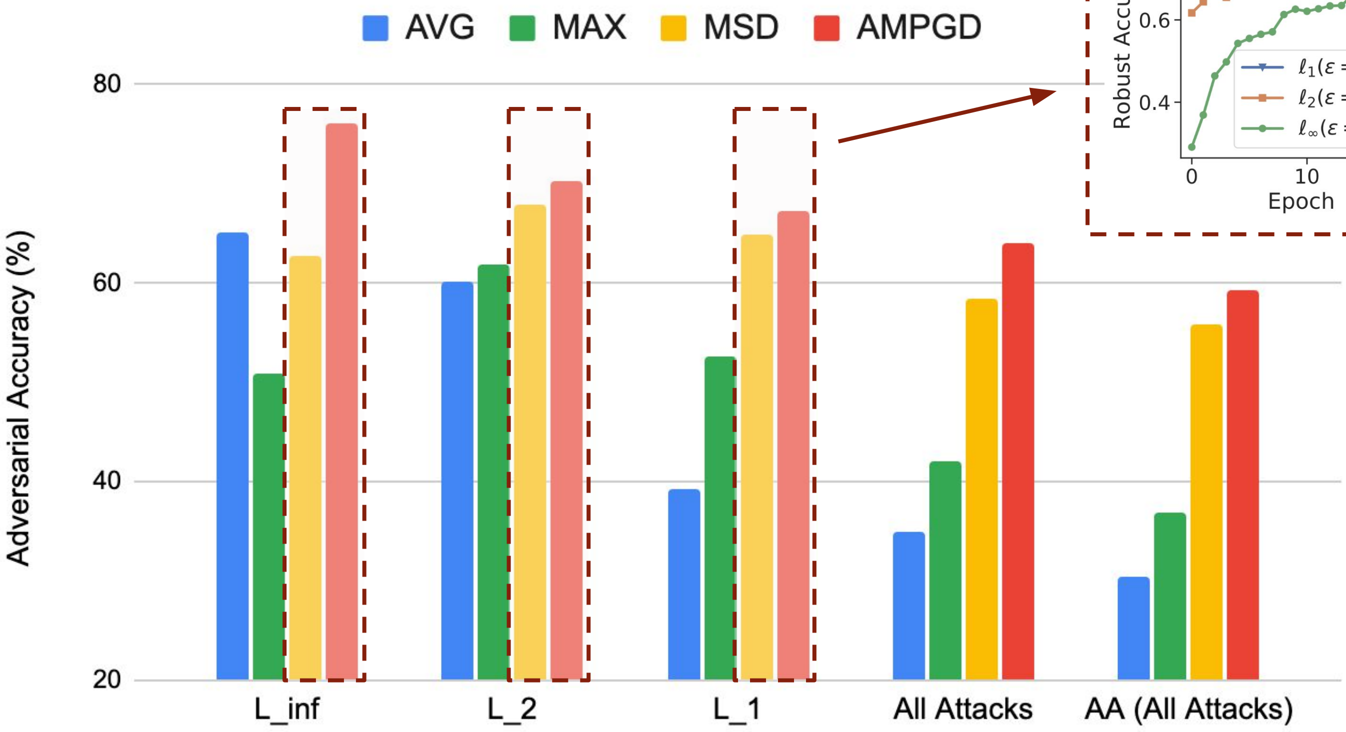
- Results on MNIST (Following Maini et. al., 2020)

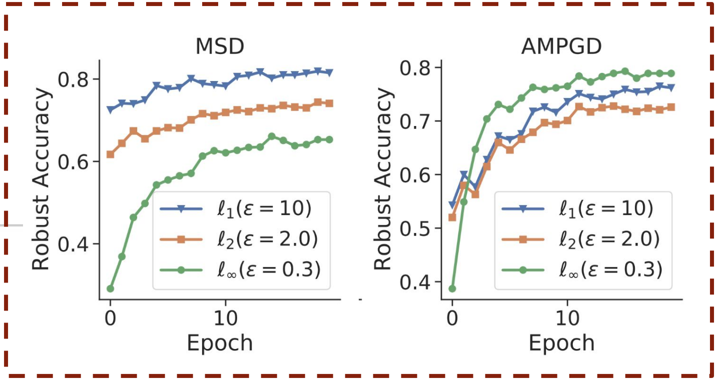




How does AMPGD work?

- Results on MNIST (Following Maini et. al., 2020)





Conclusion

- We revisit the strength of **min-max optimization** in the context of **adversarial attack**
- Beyond adversarial training, we show that many attack generation or defense problems can be re-formulated in our unified min-max framework
- Our approach results in **superior performance** as well as **interpretability**
- Our code is publicly available here: https://github.com/wangjksjtu/minmax-adv

SCAN ME

Our method has been used in the **following applications**:

Adversarial T-shirt (Xu et al., 2020), *black-box attack* (Liu et al, 2020)!