Provable Defense Against Geometric Transformations

Rem Yang, Jacob Laurel, Sasa Misailovic, Gagandeep Singh




UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

vmware

Vulnerability of Deep Neural Networks

Vulnerability of Deep Neural Networks

Correctly classified

x

Vulnerability of Deep Neural Networks

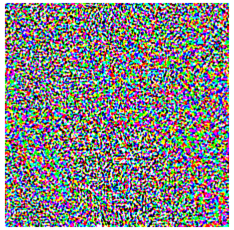
Perturbed: misclassified

ℓ_p perturbations

Correctly classified

x

$+$



δ where $\|\delta\|_p < \epsilon$

$=$

$x + \delta$

Vulnerability of Deep Neural Networks

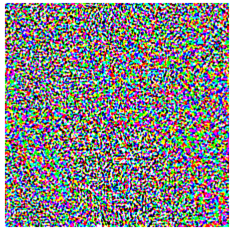
Correctly classified

x

Perturbed: misclassified

ℓ_p perturbations

$+$



$=$

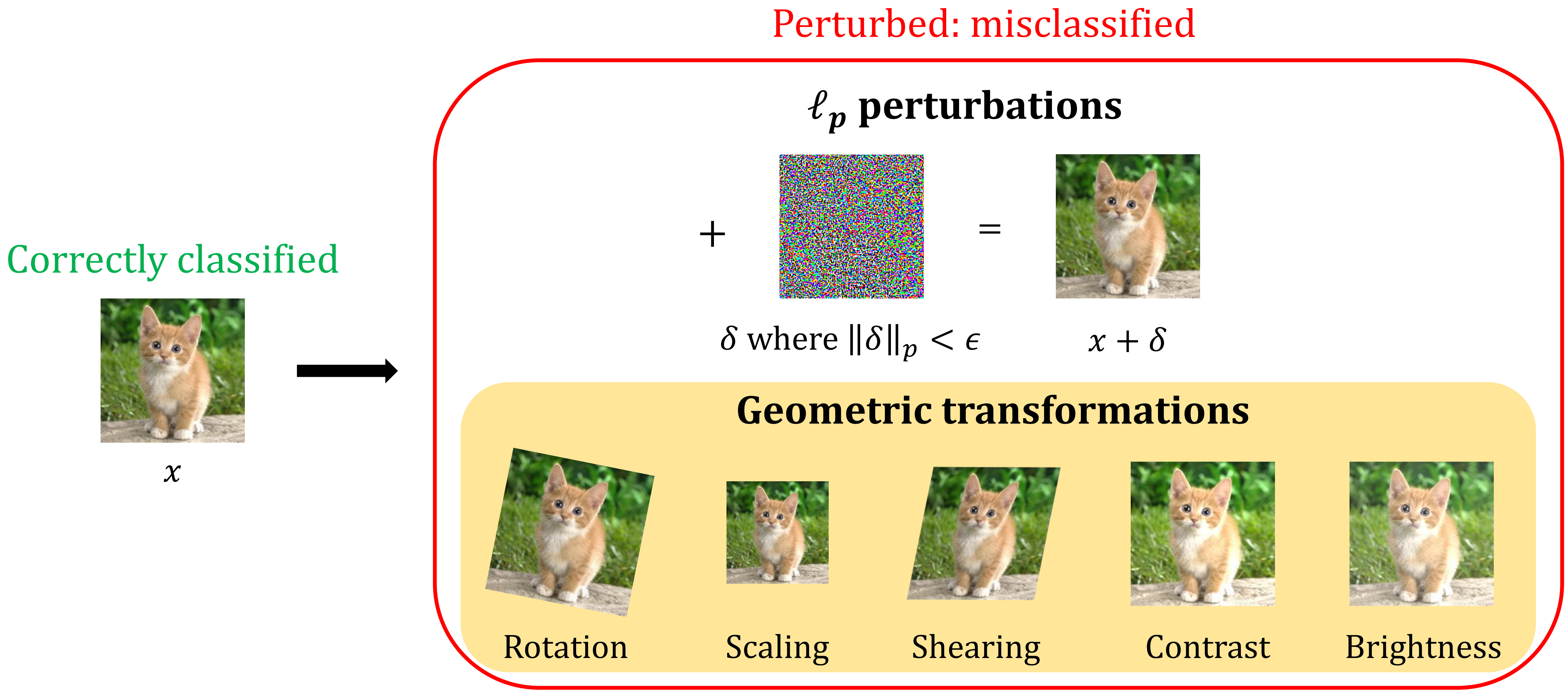
δ where $\|\delta\|_p < \epsilon$

$x + \delta$

Geometric transformations

Rotation Scaling Shearing Contrast Brightness

Vulnerability of Deep Neural Networks

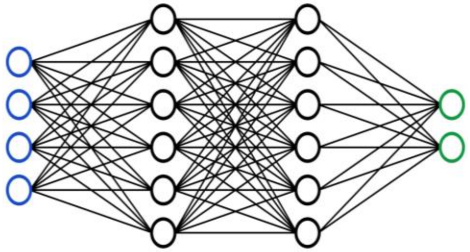


Certified Robustness

Certified Robustness

Set of perturbed images X'

Classifier f



Provably correct?

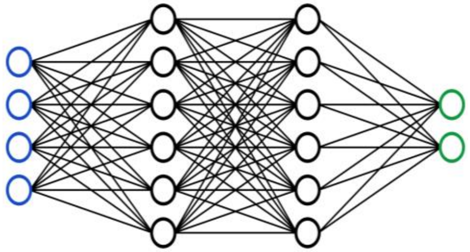
$$y = \underset{i}{\operatorname{argmax}} f_i(x') \;\; \forall x' \in X'$$

Certified Robustness

Set of perturbed images X'

Classifier f

Provably correct?

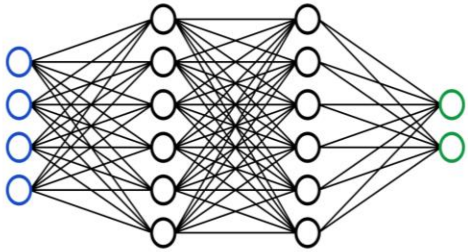


$$y = \operatorname*{argmax}_i f_i(x') \;\; \forall x' \in X'$$

Set of perturbed images X'

Regression net f

Certified output bounds



$$\underline{y} \leq \min_{x' \in X'} f(x') \leq \max_{x' \in X'} f(x') \leq \overline{y}$$

Geometric Robustness Verification

Probabilistic (Fischer et al., 2020; Hao et al., 2022; Li et al., 2021)	**Deterministic** (Balunovic et al., 2019; Mohapatra et al., 2020)
Scales to larger datasets	Only scaled up to CIFAR-10
Large inference overhead	No inference overhead

Geometric Robustness Verification

Probabilistic (Fischer et al., 2020; Hao et al., 2022; Li et al., 2021)	**Deterministic** (Balunovic et al., 2019; Mohapatra et al., 2020)
Scales to larger datasets	Only scaled up to CIFAR-10
Large inference overhead	No inference overhead

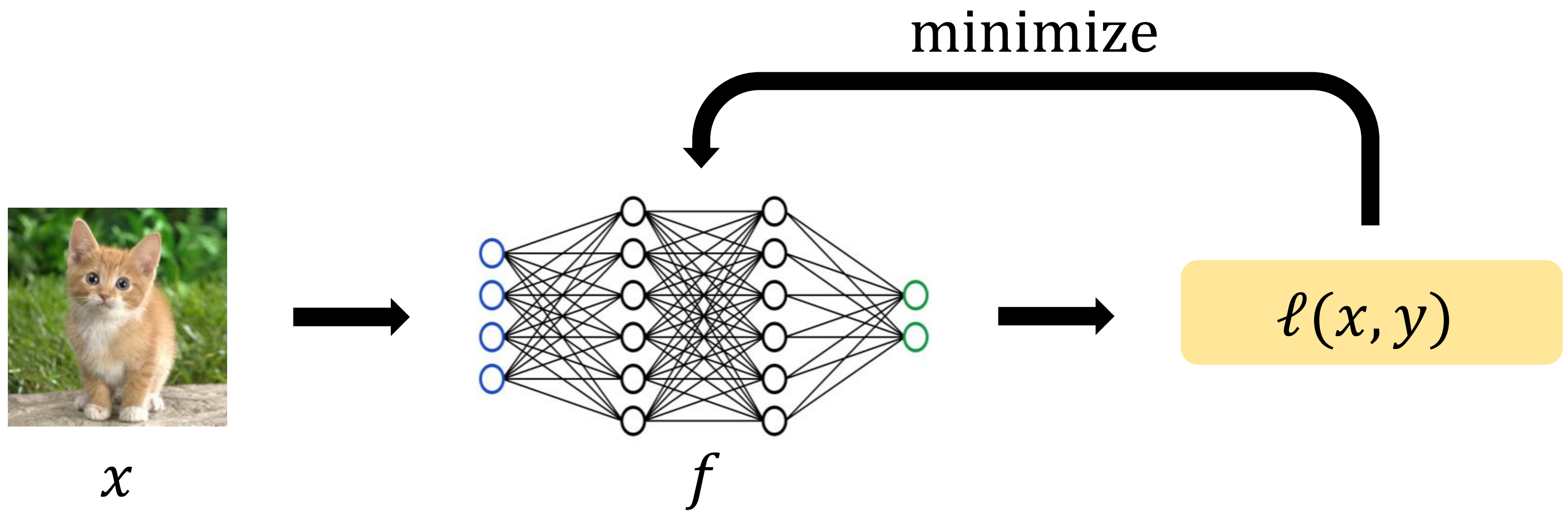
Geometric Robustness Verification

Probabilistic (Fischer et al., 2020; Hao et al., 2022; Li et al., 2021)	**Deterministic** (Balunovic et al., 2019; Mohapatra et al., 2020)
Scales to larger datasets	Only scaled up to CIFAR-10
Large inference overhead	No inference overhead

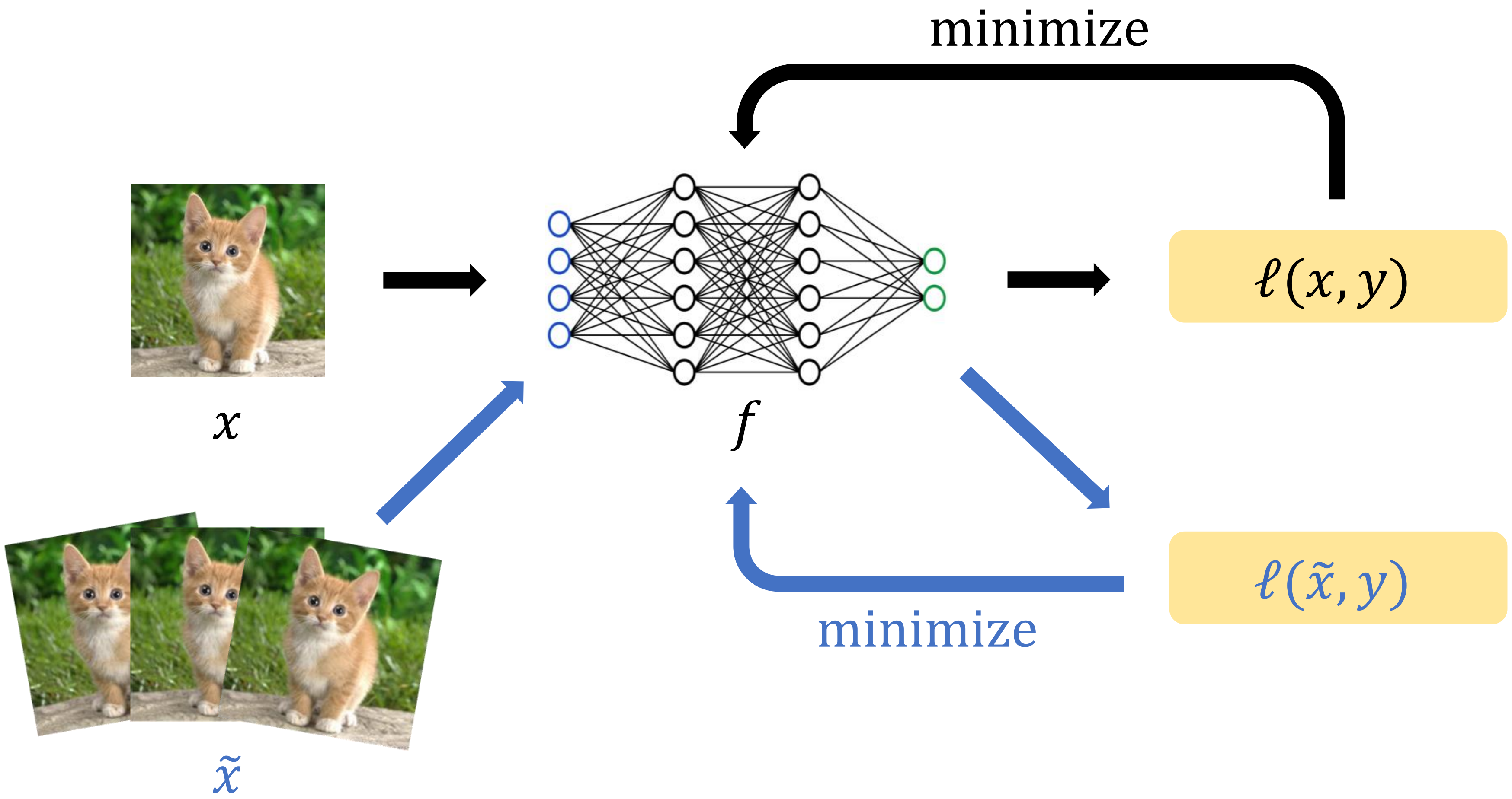
These works only verify networks not explicitly trained to be provably robust

Provable Defense

Provable Defense



Provable Defense

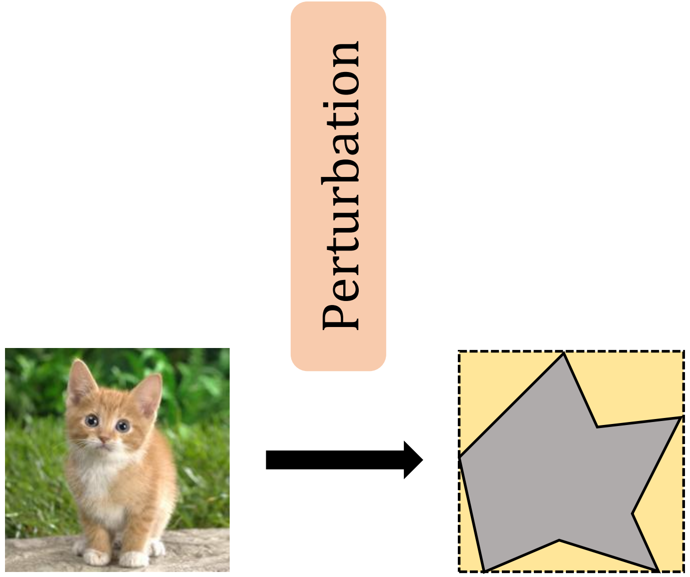


Interval Bound Propagation

Interval Bound Propagation

x

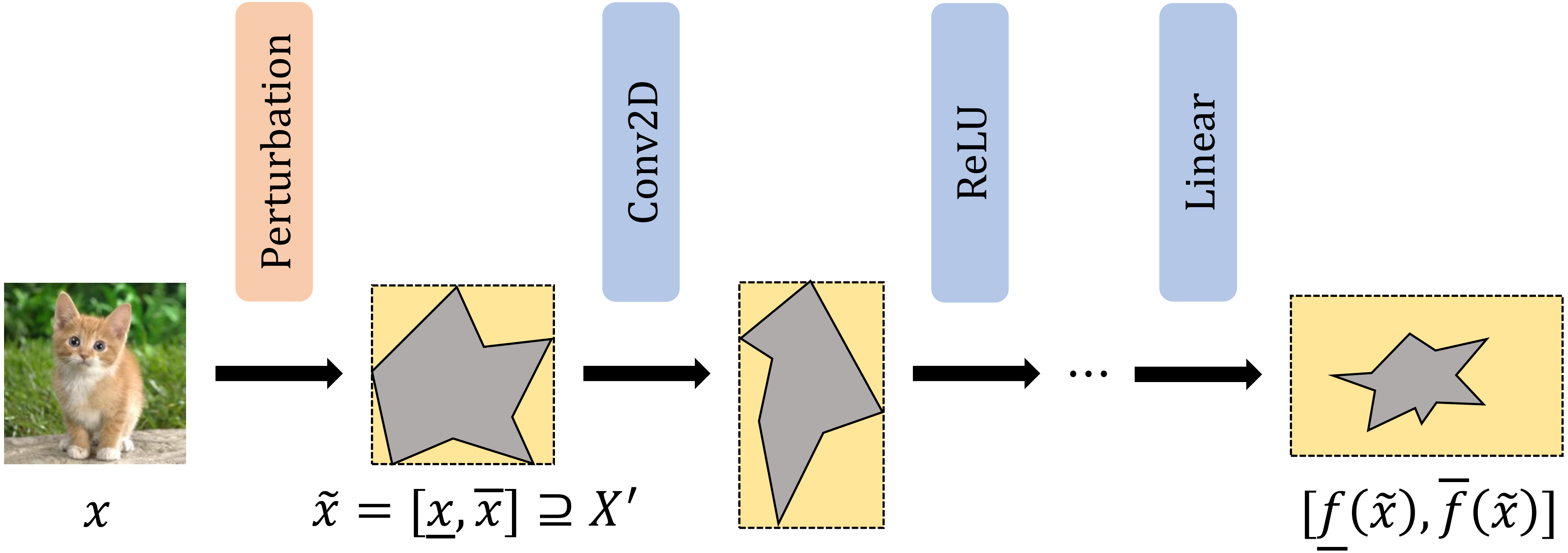
Interval Bound Propagation



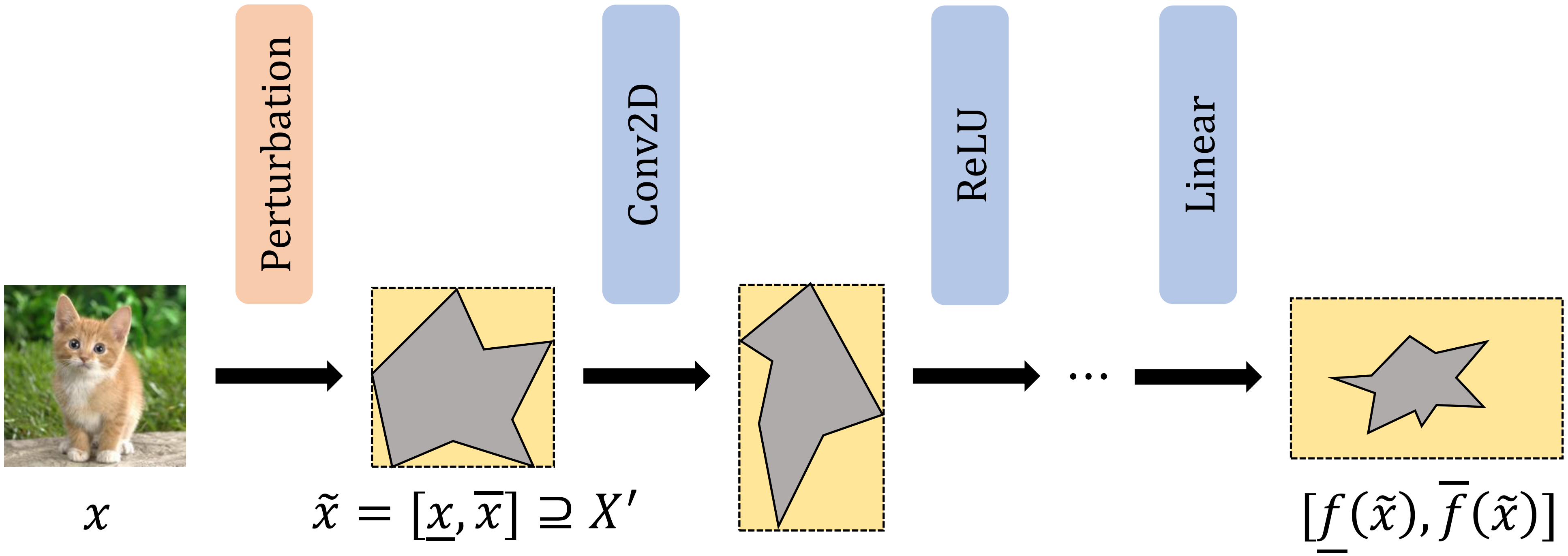
x

$\tilde{x} = [\underline{x}, \overline{x}] \supseteq X'$

Interval Bound Propagation

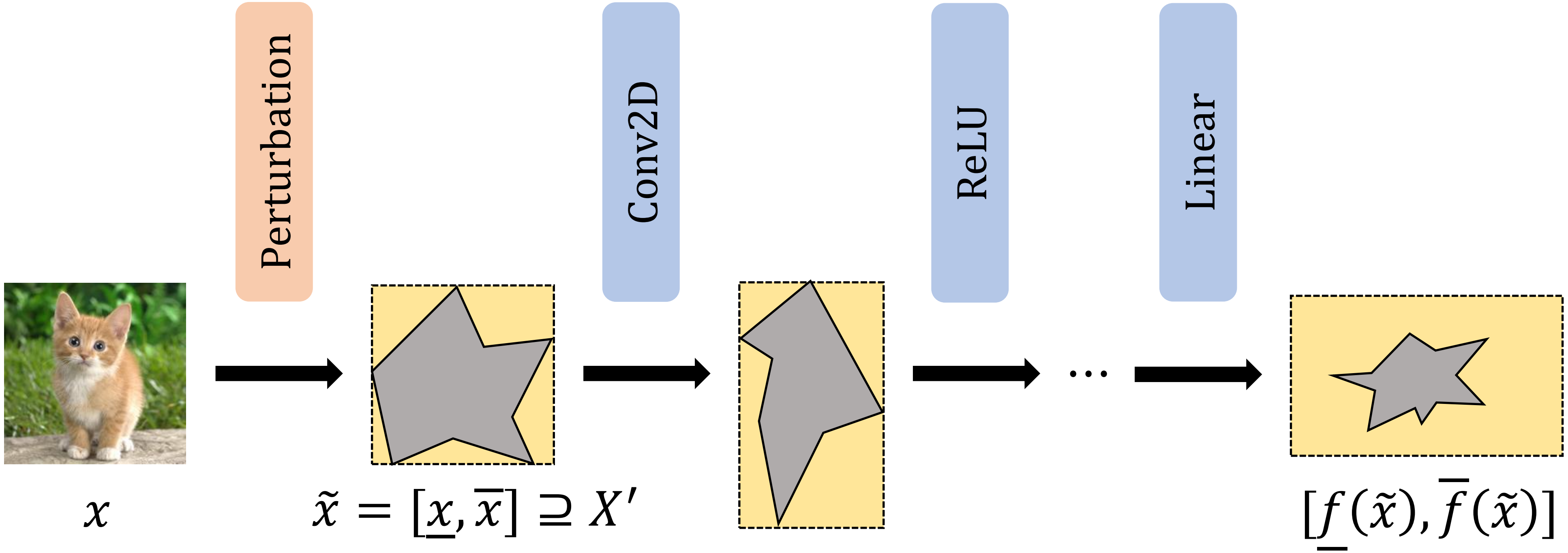


Interval Bound Propagation



Classification: $\underline{f_y}(\tilde{x}) > \overline{f_j}(\tilde{x}) \; \forall j \neq y$

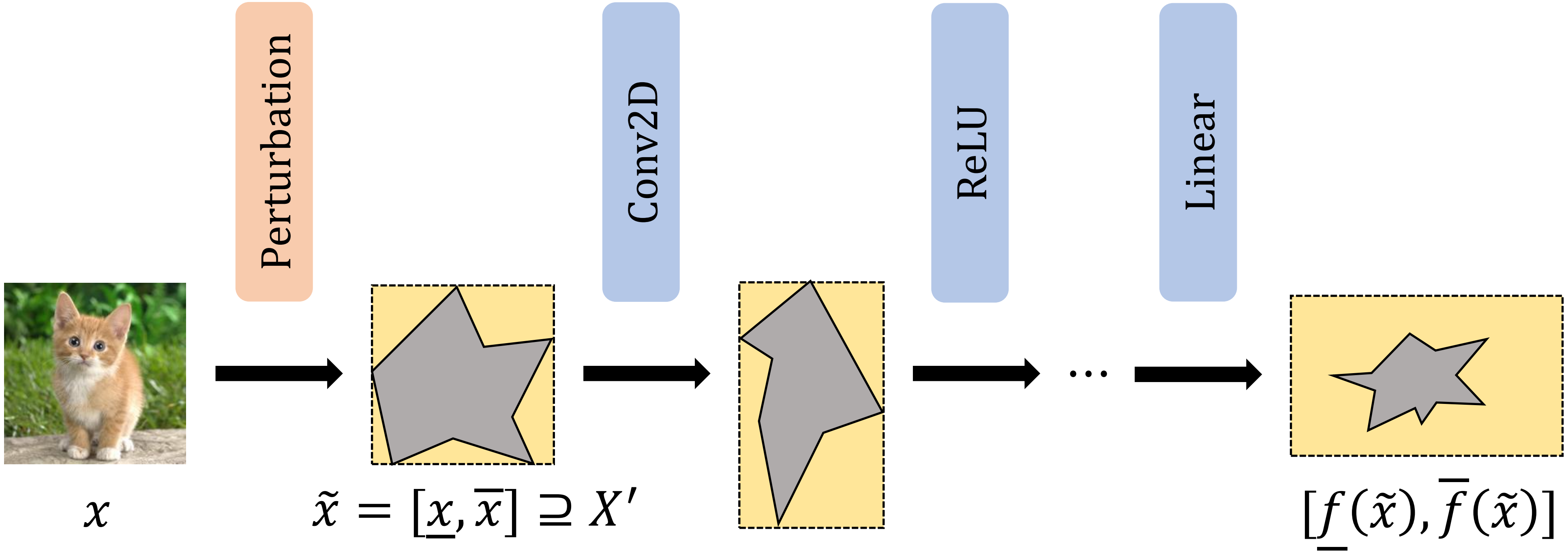
Interval Bound Propagation



Classification: $\underline{f_y}(\tilde{x}) > \overline{f_j}(\tilde{x})\ \forall j \neq y$

Regression: directly use obtained bounds

Interval Bound Propagation



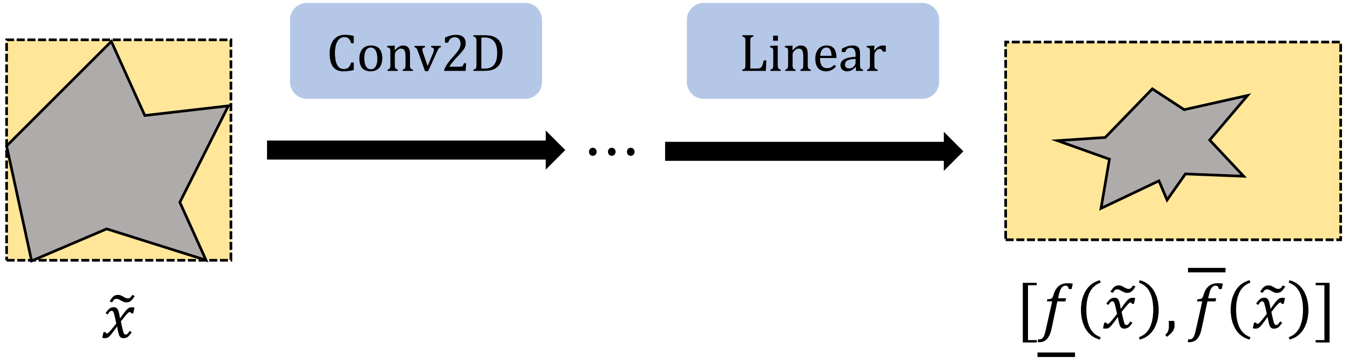
Classification: $\underline{f_y}(\tilde{x}) > \overline{f_j}(\tilde{x})\ \forall j \neq y$

Regression: directly use obtained bounds

Existing works* only handle perturbations with simple formulas, e.g., $\tilde{x} = [x - \epsilon, x + \epsilon]$

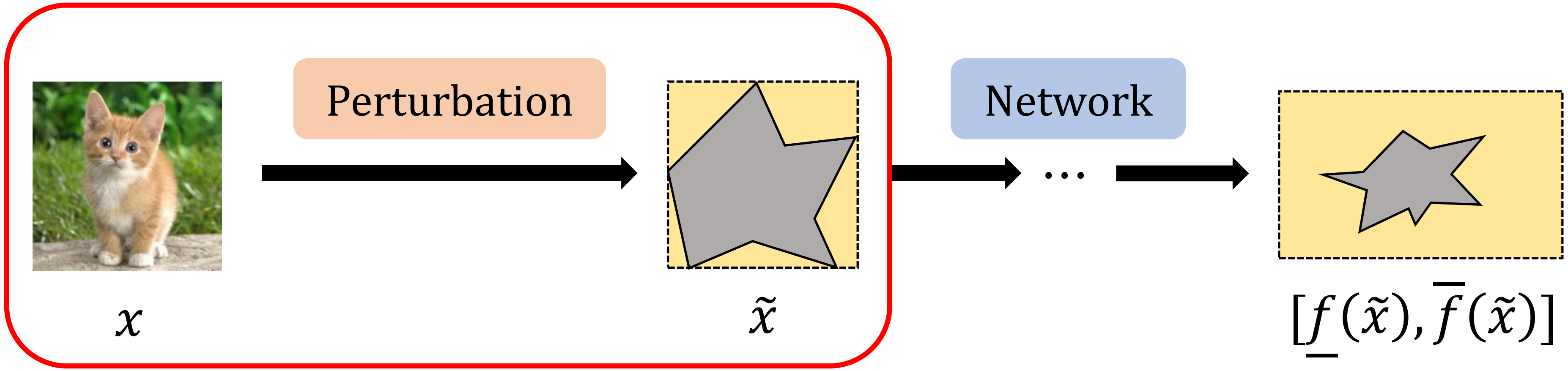
* (Gowal et al., 2019; Mirman et al., 2018; Xu et al., 2020; Zhang et al., 2020)

Interval Bound Propagation



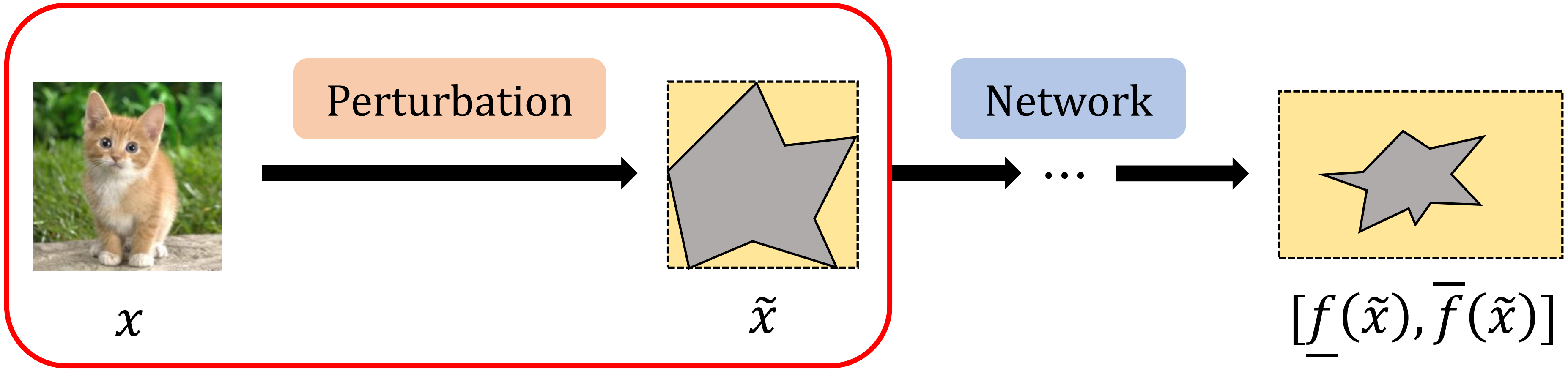
Dataset	Time to Propagate Bounds (s)
CIFAR-10	0.004
Tiny ImageNet	0.018

Interval Bound Propagation



Dataset	Time to Compute Bounds (s)	Time to Propagate Bounds (s)
CIFAR-10	22.81	0.004
Tiny ImageNet	62.83	0.018

Interval Bound Propagation



Dataset	Time to Compute Bounds (s)	Time to Propagate Bounds (s)
CIFAR-10	22.81	0.004
Tiny ImageNet	62.83	0.018

Need faster way to compute geometric perturbation bounds on GPU

Geometric Certification with Splitting

Geometric Certification with Splitting

Define geometric transformation P: $\mathbb{R}^{C\times H\times W} \times \mathbb{R}^{|\theta|} \rightarrow \mathbb{R}^{C\times H\times W}$ and interval range of parameters $\tilde{\theta} = [\underline{\theta}, \overline{\theta}]$

Geometric Certification with Splitting

Define geometric transformation P: $\mathbb{R}^{C\times H\times W} \times \mathbb{R}^{|\theta|} \to \mathbb{R}^{C\times H\times W}$ and interval range of parameters $\tilde{\theta} = [\underline{\theta}, \overline{\theta}]$

$\underline{x}$ $\quad$ $\overline{x}$

$$\tilde{x} = P(x, \tilde{\theta})$$

Geometric Certification with Splitting

Define geometric transformation $P\colon\ \mathbb{R}^{C\times H\times W} \times \mathbb{R}^{|\theta|} \to \mathbb{R}^{C\times H\times W}$ and interval range of parameters $\tilde{\theta} = [\underline{\theta}, \overline{\theta}]$

$\underline{x}$ $\overline{x}$

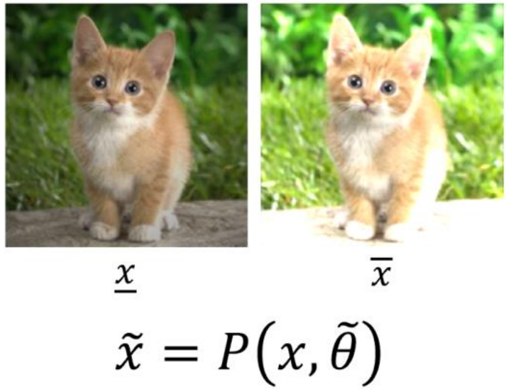
$\tilde{x} = P(x, \tilde{\theta})$

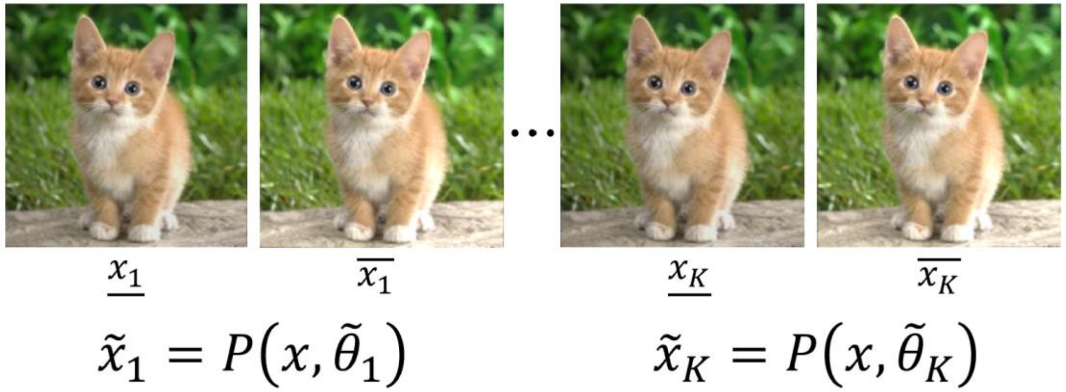
$\underline{x_1}$ $\overline{x_1}$ … $\underline{x_K}$ $\overline{x_K}$

$\tilde{x}_1 = P(x, \tilde{\theta}_1)$ $\tilde{x}_K = P(x, \tilde{\theta}_K)$

Geometric Certification with Splitting

Define geometric transformation P: $\mathbb{R}^{C\times H\times W} \times \mathbb{R}^{|\theta|} \to \mathbb{R}^{C\times H\times W}$ and interval range of parameters $\tilde{\theta} = [\underline{\theta}, \overline{\theta}]$





Classification: $\underline{f_y}(\tilde{x}_k) > \overline{f_j}(\tilde{x}_k)\ \forall j \neq y \quad \forall k \in \{1, 2, \ldots, K\}$

Geometric Certification with Splitting

Define geometric transformation P: $\mathbb{R}^{C\times H\times W} \times \mathbb{R}^{|\theta|} \to \mathbb{R}^{C\times H\times W}$ and interval range of parameters $\tilde{\theta} = [\underline{\theta}, \overline{\theta}]$

$\underline{x}$ $\overline{x}$

$\tilde{x} = P(x, \tilde{\theta})$

$\underline{x_1}$ $\overline{x_1}$ $\underline{x_K}$ $\overline{x_K}$

$\tilde{x}_1 = P(x, \tilde{\theta}_1)$ $\tilde{x}_K = P(x, \tilde{\theta}_K)$

Classification: $\underline{f_y}(\tilde{x}_k) > \overline{f_j}(\tilde{x}_k)\ \forall j \neq y\ \ \forall k \in \{1, 2, \ldots, K\}$

Regression: $\bigcup_{k=1}^{K} \{\, f(\tilde{x}_k)\, \}$

Geometric Certification with Splitting

Define geometric transformation P: $\mathbb{R}^{C\times H\times W} \times \mathbb{R}^{|\theta|} \rightarrow \mathbb{R}^{C\times H\times W}$ and interval range of parameters $\tilde{\theta} = [\underline{\theta}, \overline{\theta}]$

$\underline{x}$ $\overline{x}$

$\tilde{x} = P(x, \tilde{\theta})$

$\underline{x_1}$ $\overline{x_1}$ $\underline{x_K}$ $\overline{x_K}$

$\tilde{x}_1 = P(x, \tilde{\theta}_1)$ $\tilde{x}_K = P(x, \tilde{\theta}_K)$

Classification: $\underline{f_y}(\tilde{x}_k) > \overline{f_j}(\tilde{x}_k)\ \forall j \neq y \quad \forall k \in \{1, 2, \dots, K\}$

Regression: $\cup_{k=1}^{K}\{\, f(\tilde{x}_k)\,\}$

Need to account for splitting in training formulation

Main Contributions

Main Contributions

Certified Geometric Training (CGT)

Main Contributions

Certified Geometric Training (CGT)

- Fast Geometric Verifier (FGV)

Main Contributions

Certified Geometric Training (CGT)

- Fast Geometric Verifier (FGV)
- First provable training formulation for deterministic geometric robustness

Main Contributions

Certified Geometric Training (CGT)

- Fast Geometric Verifier (FGV)
- First provable training formulation for deterministic geometric robustness

Key Empirical Results

Main Contributions

Certified Geometric Training (CGT)

- Fast Geometric Verifier (FGV)
- First provable training formulation for deterministic geometric robustness

Key Empirical Results

- Deterministic geometric certification $60 - 42{,}000 \times$ faster than the SOTA

Main Contributions

Certified Geometric Training (CGT)

- Fast Geometric Verifier (FGV)
- First provable training formulation for deterministic geometric robustness

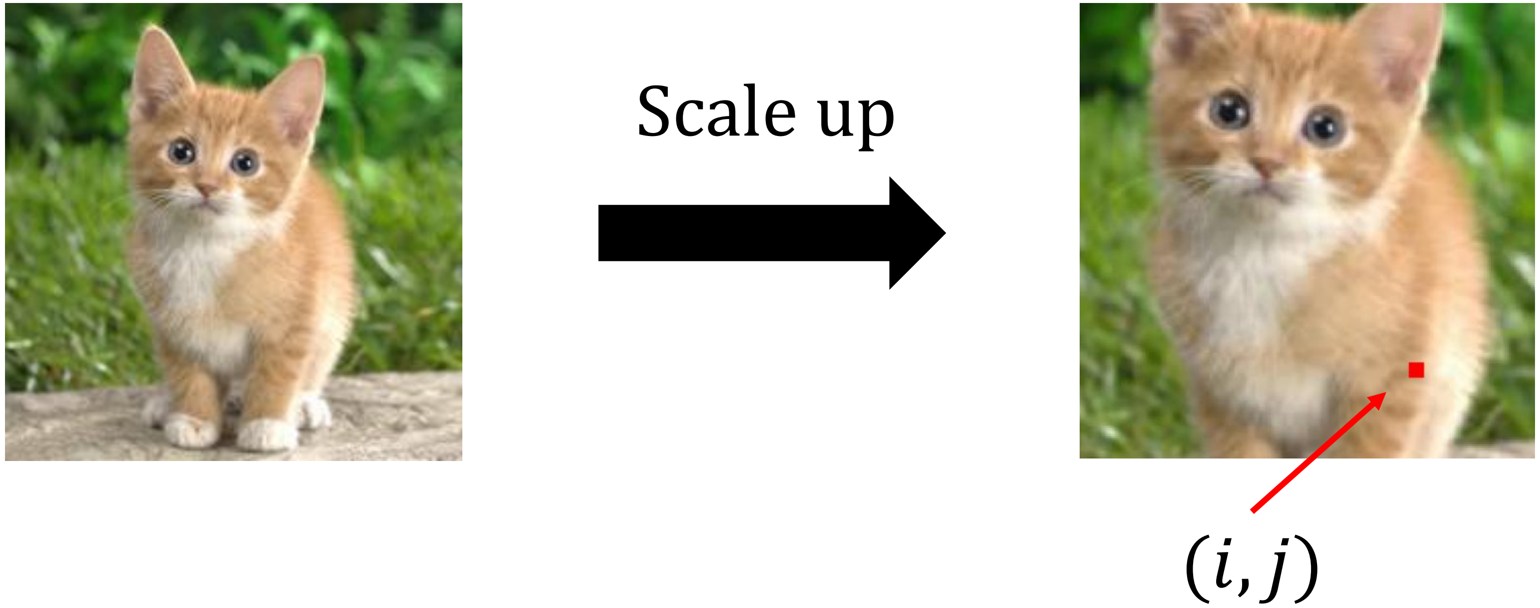
Key Empirical Results

- Deterministic geometric certification $60 - 42{,}000 \times$ faster than the SOTA
- Scales beyond CIFAR-10 to Tiny ImageNet and Udacity Self-Driving datasets

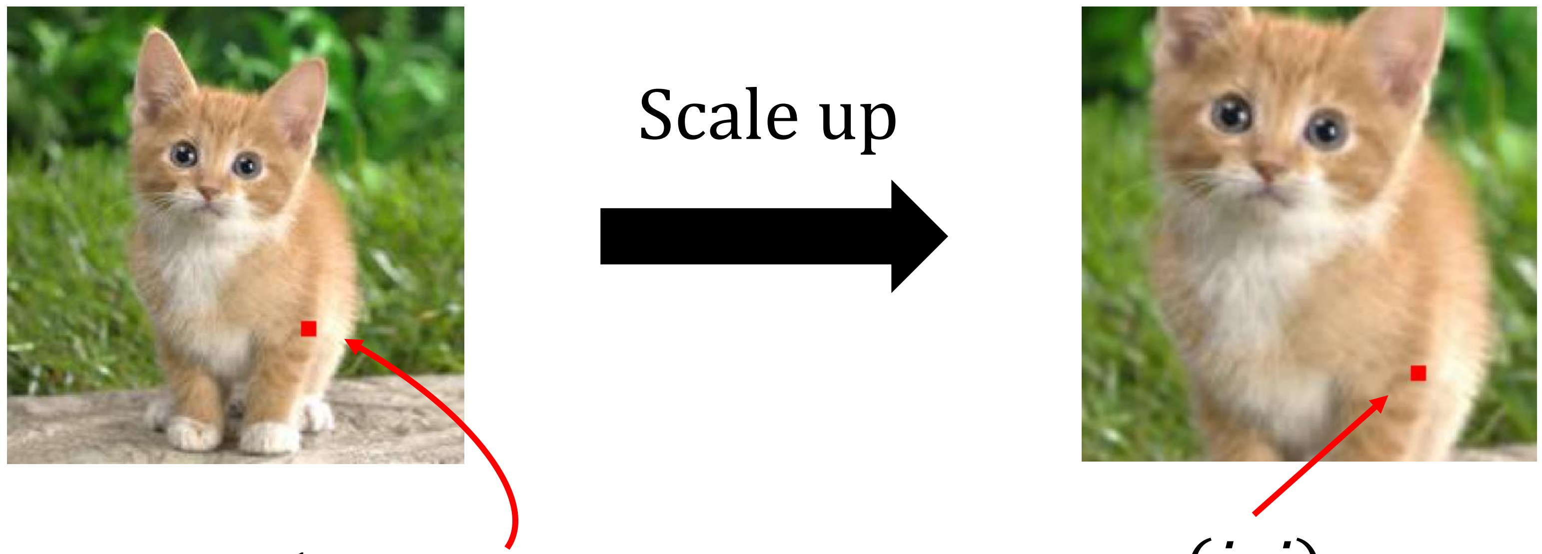
Computing Coordinate in Original Image

Scale up

Computing Coordinate in Original Image



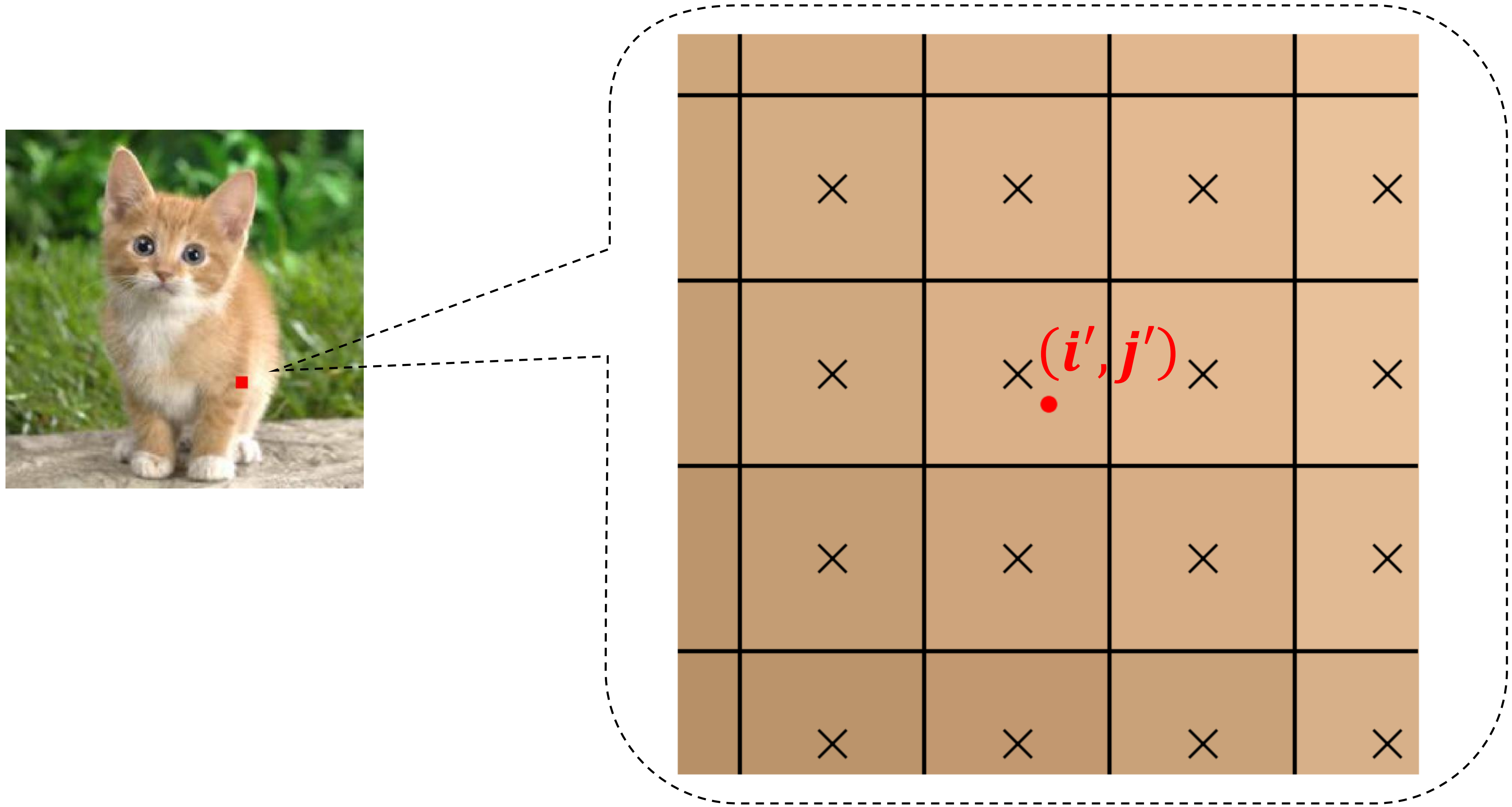
Computing Coordinate in Original Image



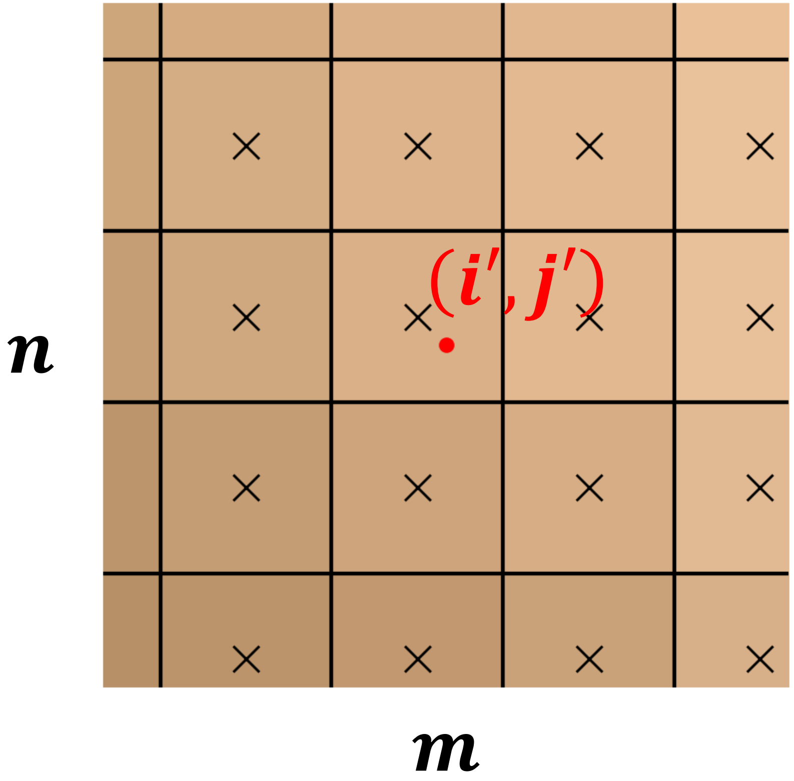
$(i', j') = {T^{-1}}_{\theta}(i, j)$

(i, j)

Coordinates Are Not Integers

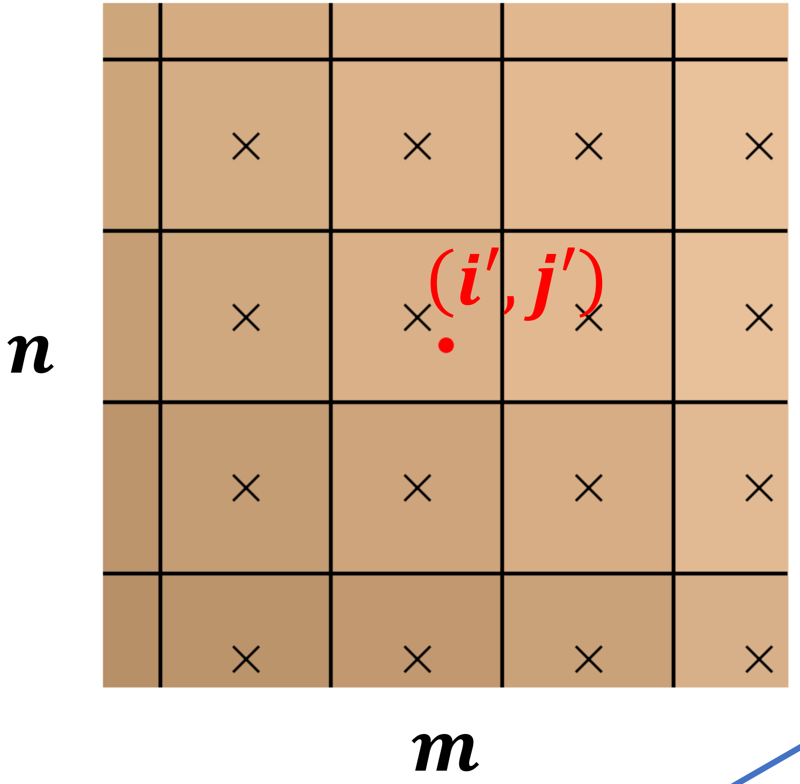


Interpolation



$$x'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \max(0, 1 - |i' - n|) \cdot \max(0, 1 - |j' - m|)$$

Interpolation



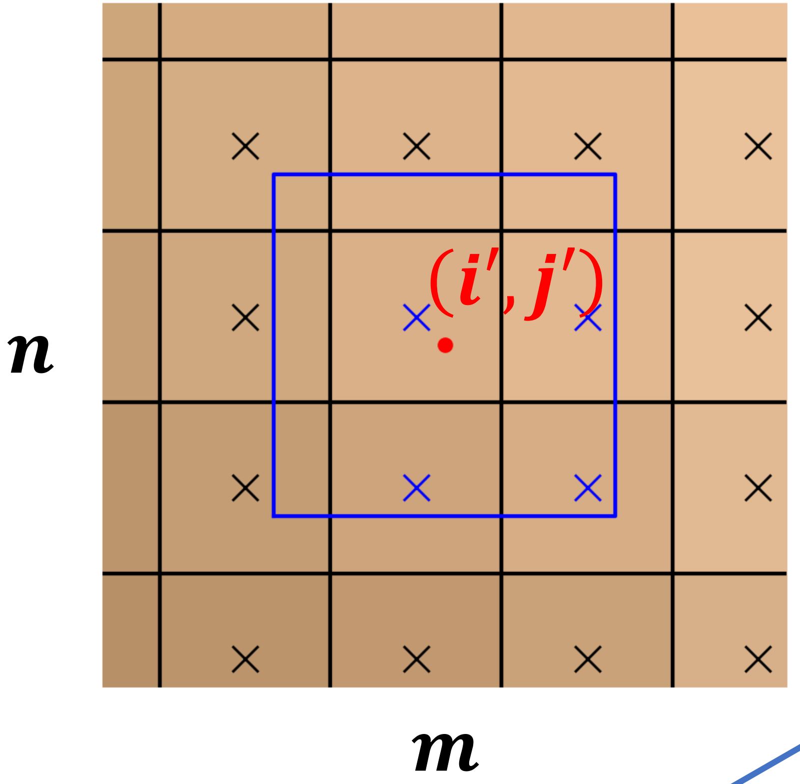
< 1

$$x'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \max(0, 1 - |i' - n|) \cdot \max(0, 1 - |j' - m|)$$

vertical distance

horizontal distance

Interpolation



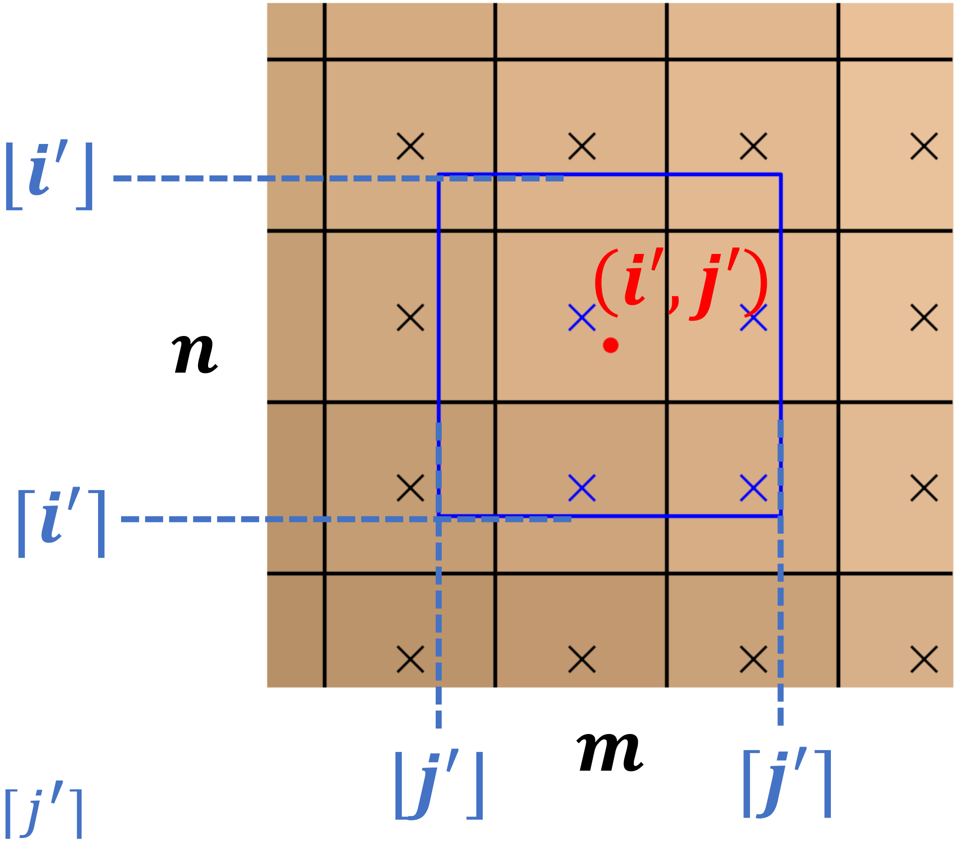
$$x'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \max(0, 1 - |i' - n|) \cdot \max(0, 1 - |j' - m|)$$

< 1

vertical distance

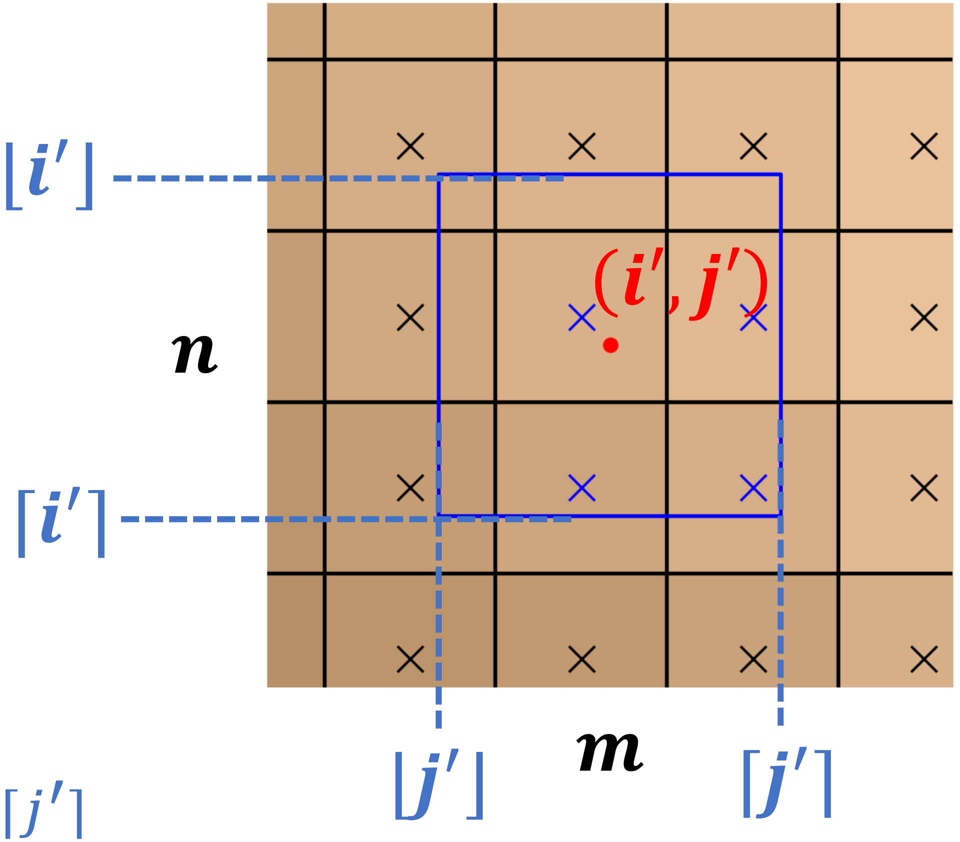
horizontal distance

Interpolation



$$x'_{i,j} = \sum_{n=\lfloor i' \rfloor}^{\lceil i' \rceil} \sum_{m=\lfloor j' \rfloor}^{\lceil j' \rceil} x_{n,m} \cdot \max(0, 1 - |i' - n|) \cdot \max(0, 1 - |j' - m|)$$

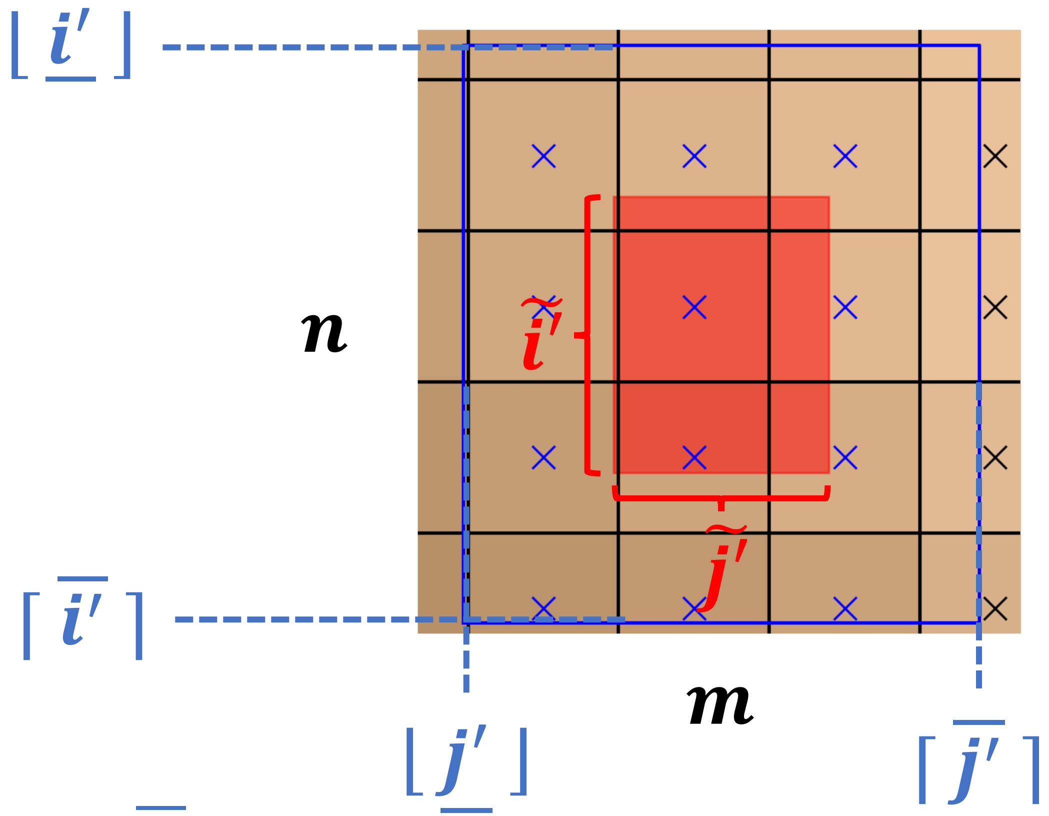
Interpolation



What if we consider an **interval** range of parameters $\tilde{\theta}$?

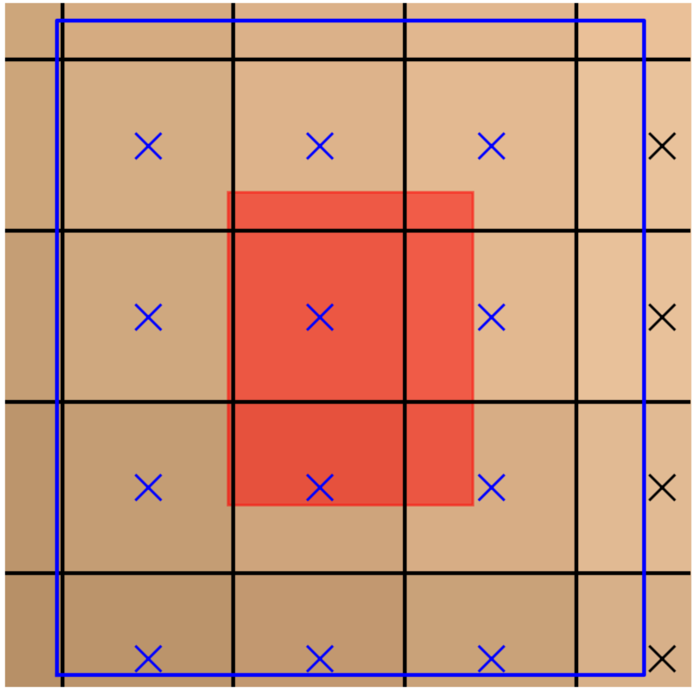
$$x'_{i,j} = \sum_{n=\lfloor i' \rfloor}^{\lceil i' \rceil} \sum_{m=\lfloor j' \rfloor}^{\lceil j' \rceil} x_{n,m} \cdot \max(0, 1 - |i' - n|) \cdot \max(0, 1 - |j' - m|)$$

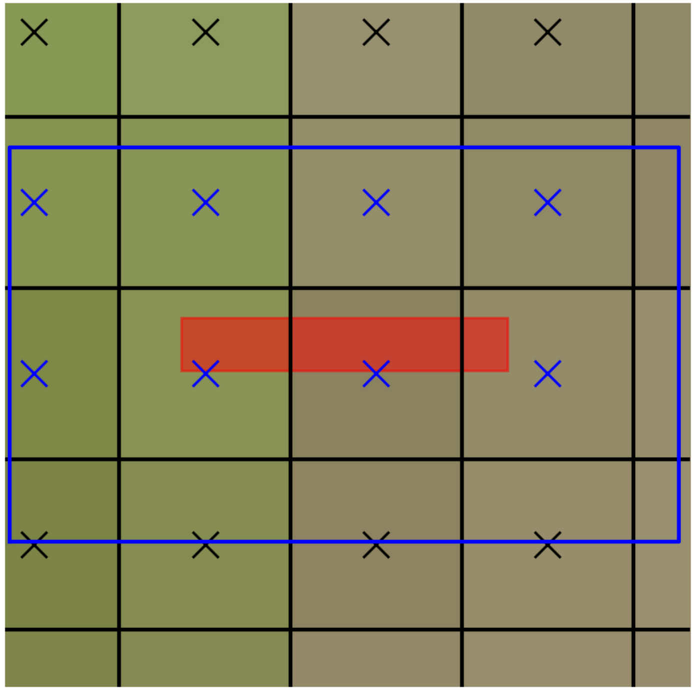
Interval Interpolation



$$\tilde{x}'_{i,j} = \sum_{n=\lfloor \underline{i}' \rfloor}^{\lceil \bar{i}' \rceil} \sum_{m=\lfloor \underline{j}' \rfloor}^{\lceil \bar{j}' \rceil} x_{n,m} \cdot \max\left(0, 1 - \left|\tilde{i}' - n\right|\right) \cdot \max\left(0, 1 - \left|\tilde{j}' - m\right|\right)$$

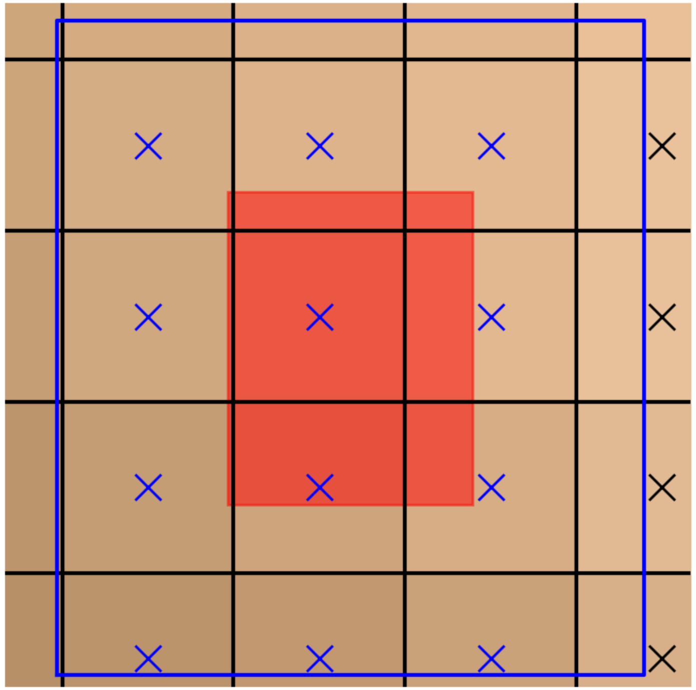
Interval Interpolation

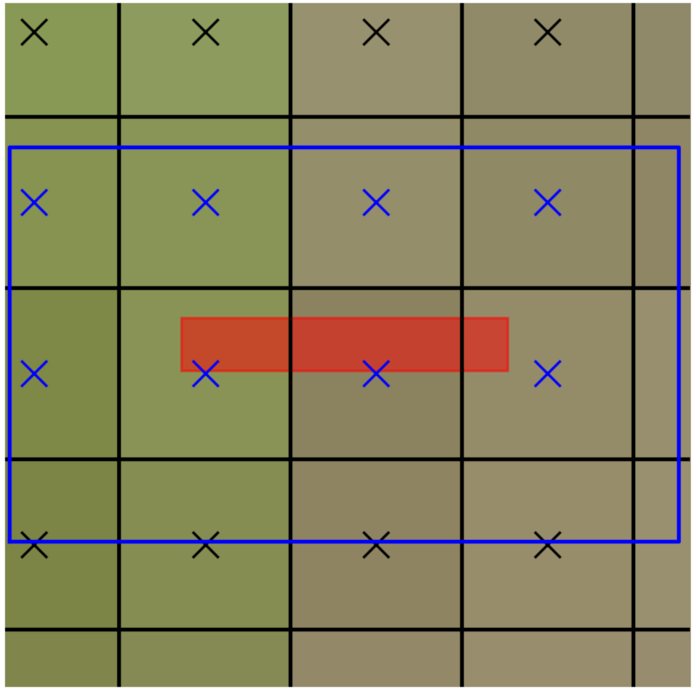




Interpolation region for a different $(\widetilde{i'}, \widetilde{j'})$

Interval Interpolation





Interpolation region for a different $(\widetilde{i'}, \widetilde{j'})$

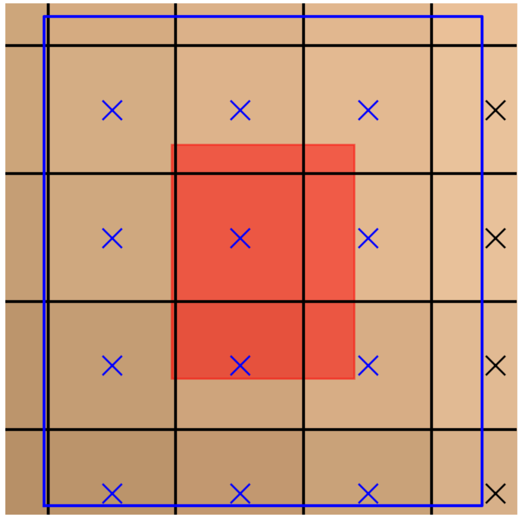
Each pixel's interpolation bounds differ; not GPU-parallelizable!

How do we GPU-parallelize interval interpolation?

How do we GPU-parallelize interval interpolation?

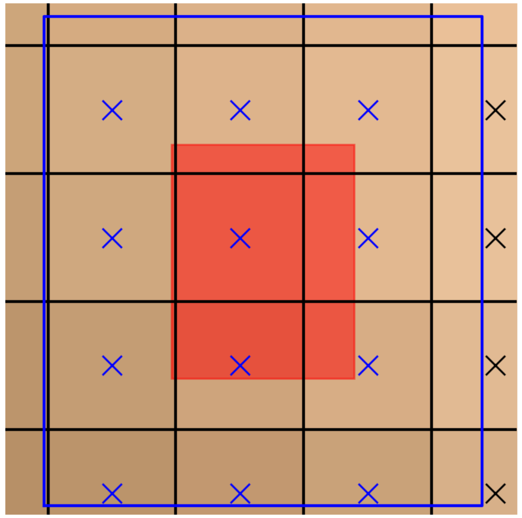
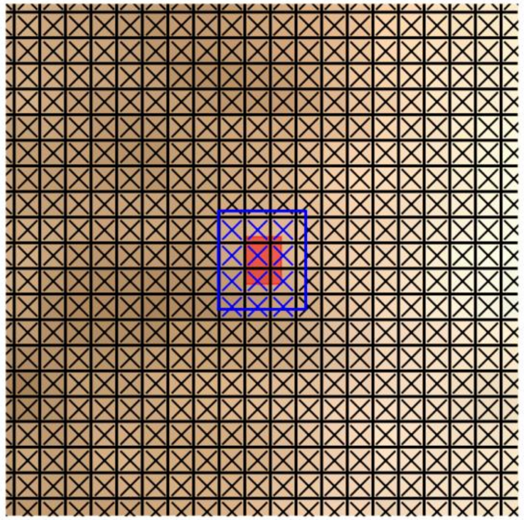
Fast Geometric Verifier

Fast Geometric Verifier



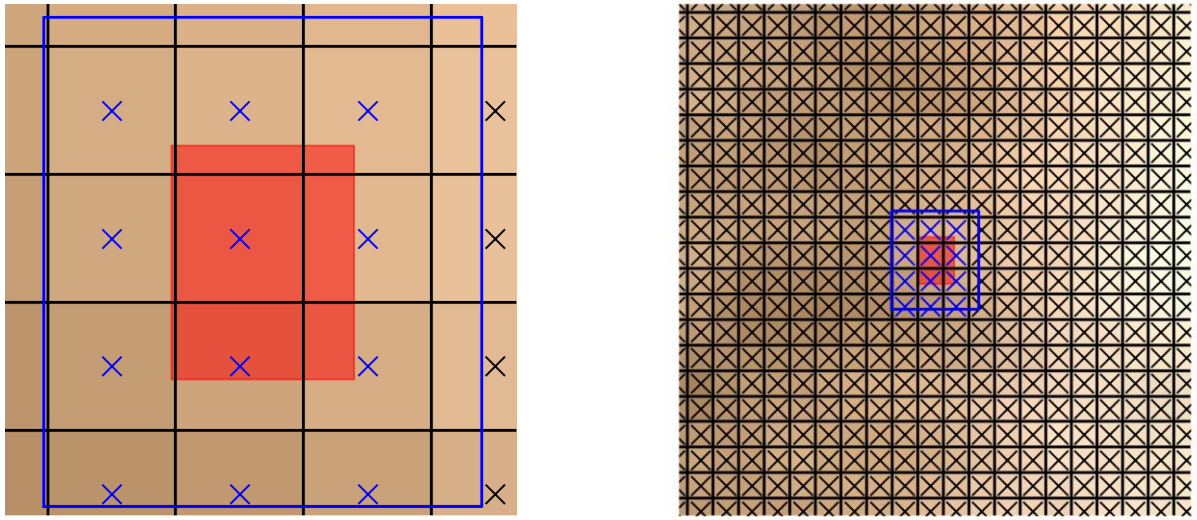
$$\tilde{x}'_{i,j} = \sum_{n=\lfloor \underline{i'} \rfloor}^{\lceil \overline{i'} \rceil} \sum_{m=\lfloor \underline{j'} \rfloor}^{\lceil \overline{j'} \rceil} x_{n,m} \cdot \max\left(0, 1 - \left|\widetilde{i'} - n\right|\right) \cdot \max\left(0, 1 - \left|\widetilde{j'} - m\right|\right)$$

Fast Geometric Verifier

$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \max\left(0, 1 - \left|\tilde{i}' - n\right|\right) \cdot \max\left(0, 1 - \left|\tilde{j}' - m\right|\right)$$

Fast Geometric Verifier



$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \max\left(0, 1 - \left|\tilde{i}' - n\right|\right) \cdot \max\left(0, 1 - \left|\tilde{j}' - m\right|\right)$$

Lots of multiplication of pixel values with zero distances!

Fast Geometric Verifier

Decompose expression and precompute interpolation distances

Fast Geometric Verifier

Decompose expression and precompute interpolation distances

$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \underbrace{\max\left(0, 1 - \left|\tilde{i}' - n\right|\right) \cdot \max\left(0, 1 - \left|\tilde{j}' - m\right|\right)}_{\tilde{d}^{i,j}_{n,m}}$$

Fast Geometric Verifier

Decompose expression and precompute interpolation distances

$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \underbrace{\max\left(0, 1 - \left|\tilde{i}' - n\right|\right) \cdot \max\left(0, 1 - \left|\tilde{j}' - m\right|\right)}_{\tilde{d}^{i,j}_{n,m}}$$

For a given (i, j):

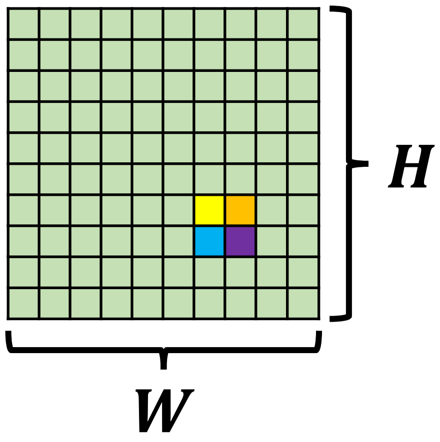
Fast Geometric Verifier

Decompose expression and precompute interpolation distances

$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \underbrace{\max(0, 1 - |\tilde{i}' - n|) \cdot \max(0, 1 - |\tilde{j}' - m|)}_{\tilde{d}^{i,j}_{n,m}}$$

For a given (i, j):

$\tilde{d}^{i,j} =$

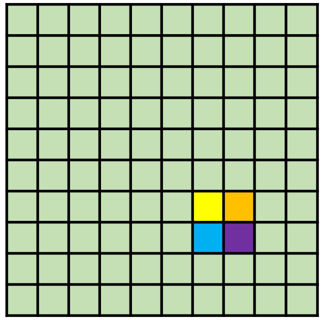


Fast Geometric Verifier

Decompose expression and precompute interpolation distances

$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \tilde{d}_{n,m}^{i,j}$$

For a given (i, j):



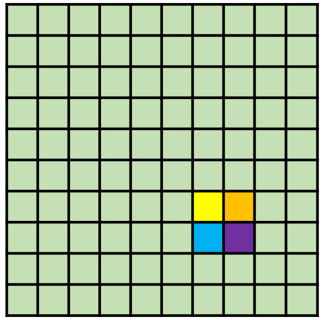
Fast Geometric Verifier

Decompose expression and precompute interpolation distances

$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \tilde{d}_{n,m}^{i,j}$$

For a given (i, j):

$\odot$



Fast Geometric Verifier

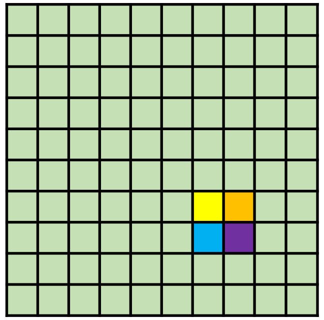
Decompose expression and precompute interpolation distances

$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \tilde{d}^{\,i,j}_{n,m}$$

For a given (i, j):

$\tilde{x}'_{i,j} = \sum$

$\odot$



Fast Geometric Verifier

Decompose expression and precompute interpolation distances

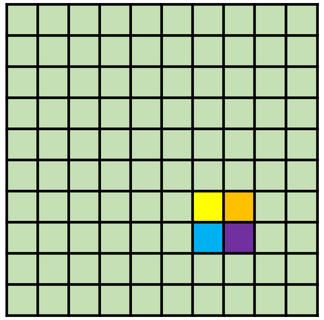
$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \tilde{d}_{n,m}^{i,j}$$

For a given (i, j):

$\tilde{x}'_{i,j} = \sum$

HW

$\odot$



HW

Fast Geometric Verifier

Decompose expression and precompute interpolation distances

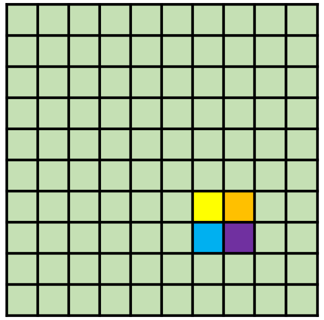
$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \tilde{d}^{i,j}_{n,m}$$

For a given (i, j):

$\tilde{x}'_{i,j} = \sum$

HW

$\odot$



HW

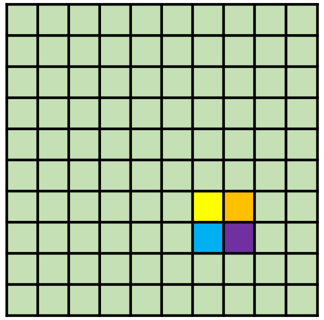
$\cdot$ CHW pixels

Fast Geometric Verifier

Decompose expression and precompute interpolation distances

$$\tilde{x}'_{i,j} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} x_{n,m} \cdot \tilde{d}_{n,m}^{i,j}$$

For a given (i, j):

$\tilde{x}'_{i,j} = \sum$  $\odot$ 

HW HW

- CHW pixels
- B images

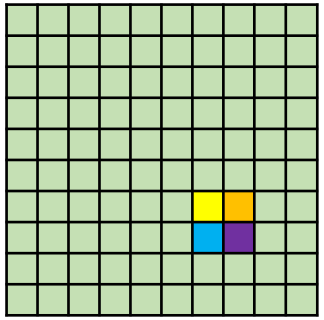
Prohibitive memory overhead!

Fast Geometric Verifier

Leverage custom sparse tensor representation

$\tilde{x}'_{i,j} = \sum$

$\odot$

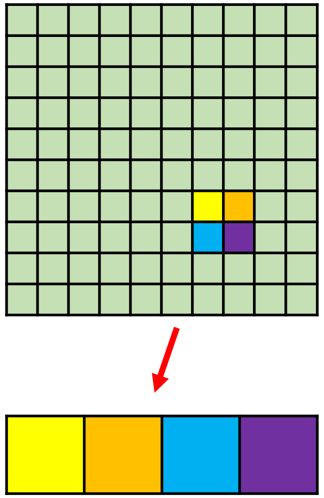


Fast Geometric Verifier

Leverage custom sparse tensor representation

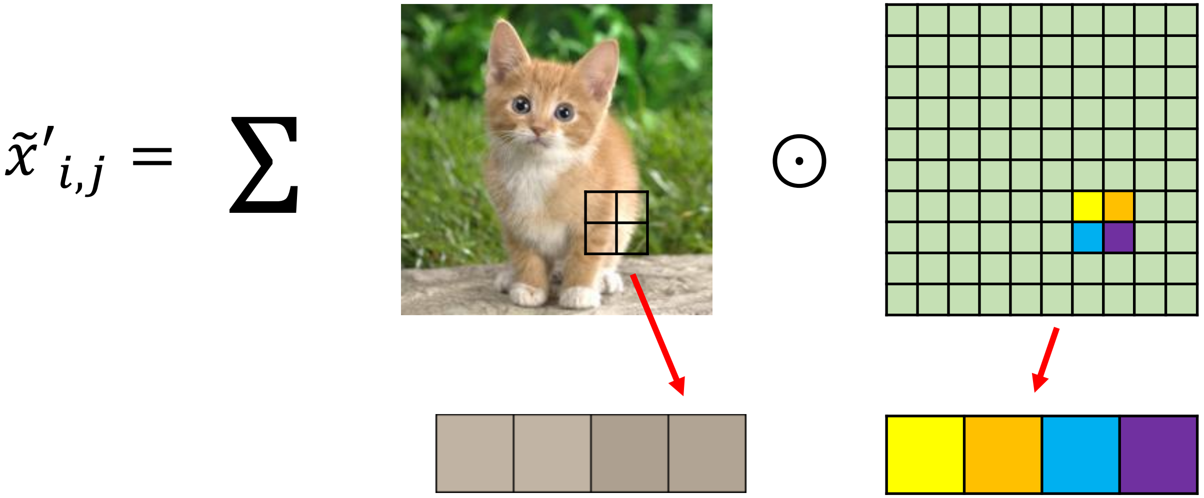
$\tilde{x}'_{i,j} = \sum$

$\odot$



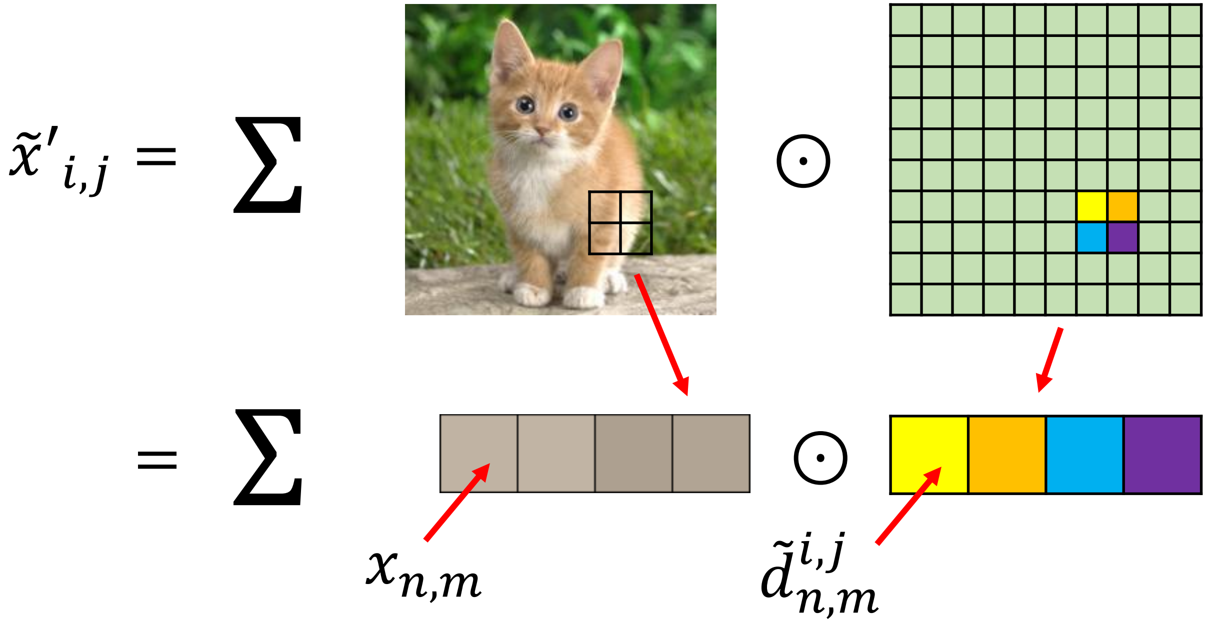
Fast Geometric Verifier

Leverage custom sparse tensor representation



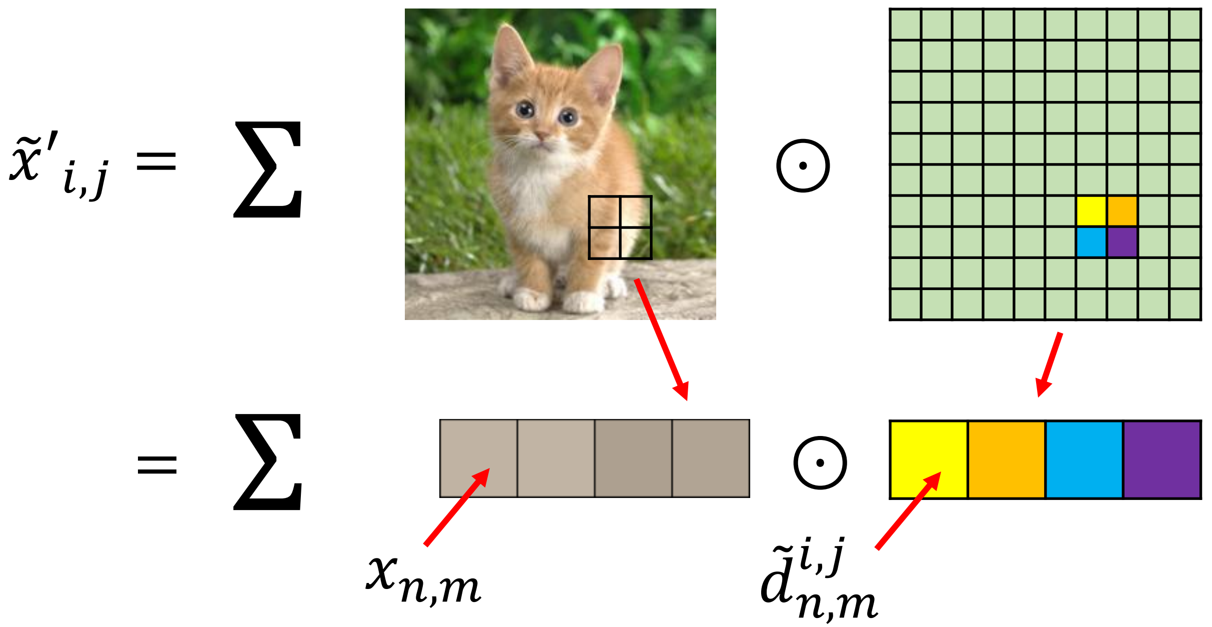
Fast Geometric Verifier

Leverage custom sparse tensor representation



Fast Geometric Verifier

Leverage custom sparse tensor representation



No multiplication of pixel values with zero distances

Robust Geometric Loss

Robust Geometric Loss

Define worst-case output vector $\hat{f}$ where $\hat{f}_y = \underline{f_y}$ and $\hat{f}_j = \overline{f_j} \;\; \forall j \neq y$

Robust Geometric Loss

Define worst-case output vector $\hat{f}$ where $\hat{f}_y = \underline{f_y}$ and $\hat{f}_j = \overline{f_j}\ \forall j \neq y$

Existing formulations [worst-case loss over *entire* perturbation region]

$$\ell\big(\hat{f}(\tilde{x}), y\big)$$

Robust Geometric Loss

Define worst-case output vector $\hat{f}$ where $\hat{f}_y = \underline{f_y}$ and $\hat{f}_j = \overline{f_j}\ \ \forall j \neq y$

Existing formulations [worst-case loss over *entire* perturbation region]

$$\ell\left(\hat{f}(\tilde{x}), y\right)$$

Our formulation [worst-case loss over *small, sampled* regions]

Robust Geometric Loss

Define worst-case output vector $\hat{f}$ where $\hat{f}_y = \underline{f_y}$ and $\hat{f}_j = \overline{f_j}$ $\forall j \neq y$

Existing formulations [worst-case loss over *entire* perturbation region]

$$\ell\left(\hat{f}(\tilde{x}), y\right)$$

Our formulation [worst-case loss over *small, sampled* regions]

To enforce robustness to P across entire parameter range $\tilde{\theta} = \left[\underline{\theta}, \overline{\theta}\right]$:

Robust Geometric Loss

Define worst-case output vector $\hat{f}$ where $\hat{f}_y = \underline{f_y}$ and $\hat{f}_j = \overline{f_j}\ \ \forall j \neq y$

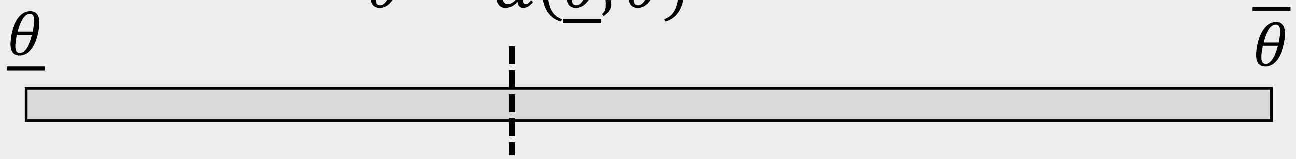
Existing formulations [worst-case loss over *entire* perturbation region]

$$\ell\left(\hat{f}(\tilde{x}), y\right)$$

Our formulation [worst-case loss over *small, sampled* regions]

To enforce robustness to P across entire parameter range $\tilde{\theta} = \left[\underline{\theta}, \overline{\theta}\right]$:

$$\theta \sim \mathcal{U}(\underline{\theta}, \overline{\theta})$$



Robust Geometric Loss

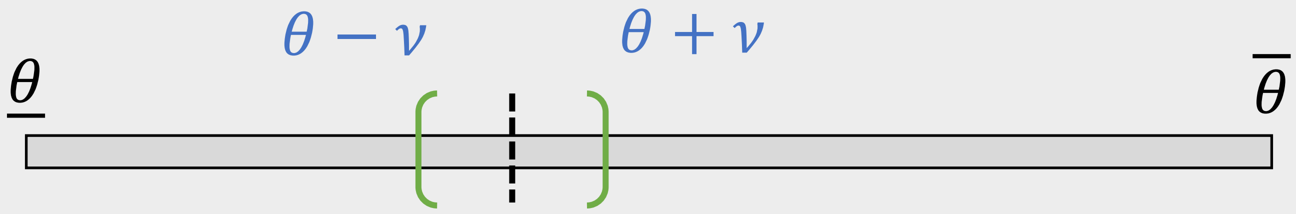
Define worst-case output vector $\hat{f}$ where $\hat{f}_y = \underline{f_y}$ and $\hat{f}_j = \overline{f_j}\ \ \forall j \neq y$

Existing formulations [worst-case loss over *entire* perturbation region]

$$\ell\left(\hat{f}(\tilde{x}), y\right)$$

Our formulation [worst-case loss over *small, sampled* regions]

To enforce robustness to P across entire parameter range $\tilde{\theta} = \left[\underline{\theta}, \overline{\theta}\right]$:



Robust Geometric Loss

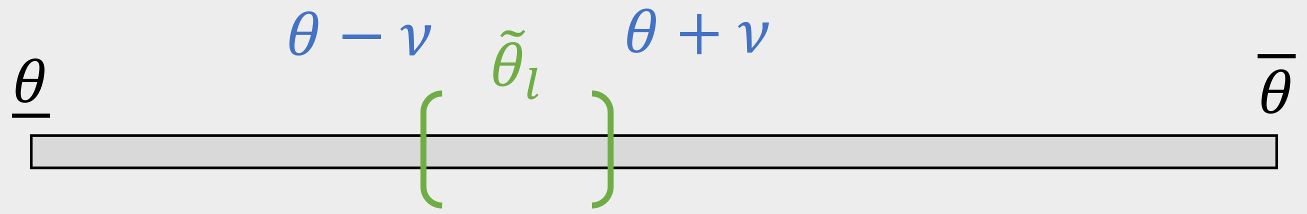
Define worst-case output vector $\hat{f}$ where $\hat{f}_y = \underline{f_y}$ and $\hat{f}_j = \overline{f_j}\ \ \forall j \neq y$

Existing formulations [worst-case loss over *entire* perturbation region]

$$\ell\left(\hat{f}(\tilde{x}), y\right)$$

Our formulation [worst-case loss over *small, sampled* regions]

To enforce robustness to P across entire parameter range $\tilde{\theta} = \left[\underline{\theta}, \overline{\theta}\right]$:



Robust Geometric Loss

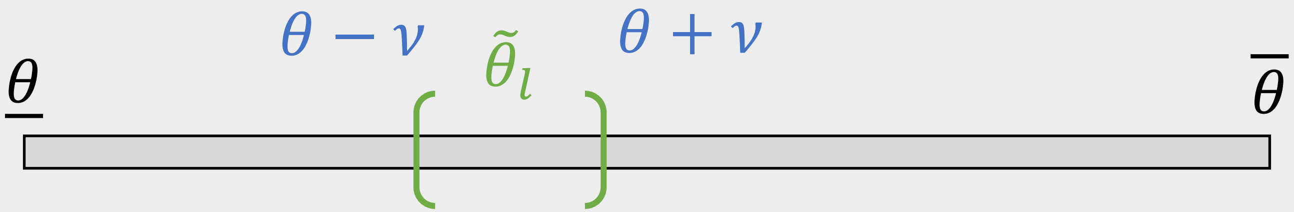
Define worst-case output vector $\hat{f}$ where $\hat{f}_y = \underline{f_y}$ and $\hat{f}_j = \overline{f_j}$ $\forall j \neq y$

Existing formulations [worst-case loss over *entire* perturbation region]

$$\ell\left(\hat{f}(\tilde{x}), y\right)$$

Our formulation [worst-case loss over *small, sampled* regions]

To enforce robustness to P across entire parameter range $\tilde{\theta} = \left[\underline{\theta}, \overline{\theta}\right]$:



$$\ell\left(\hat{f}\left(P\left(x, \tilde{\theta}_l\right)\right), y\right)$$

Robust Geometric Loss

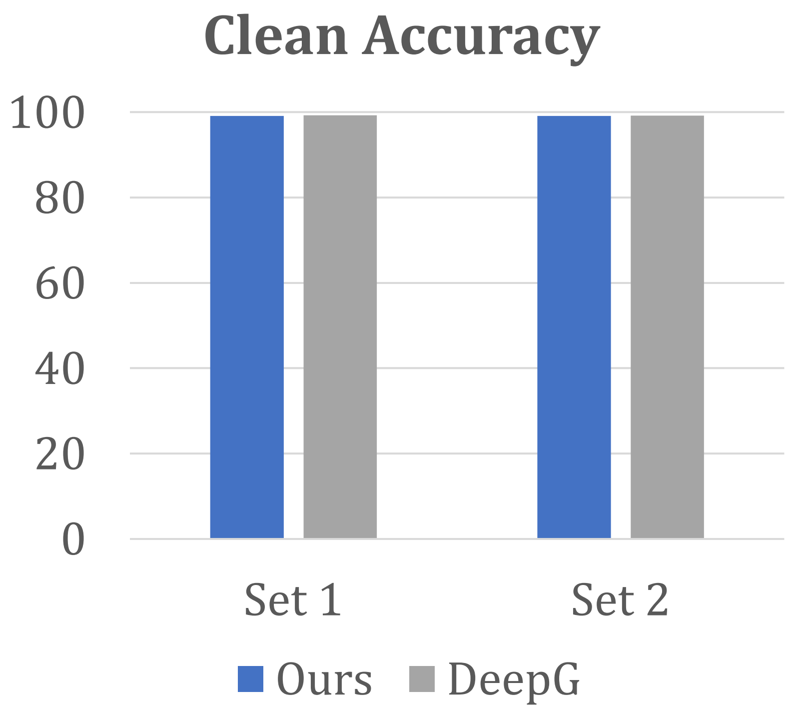
Regression

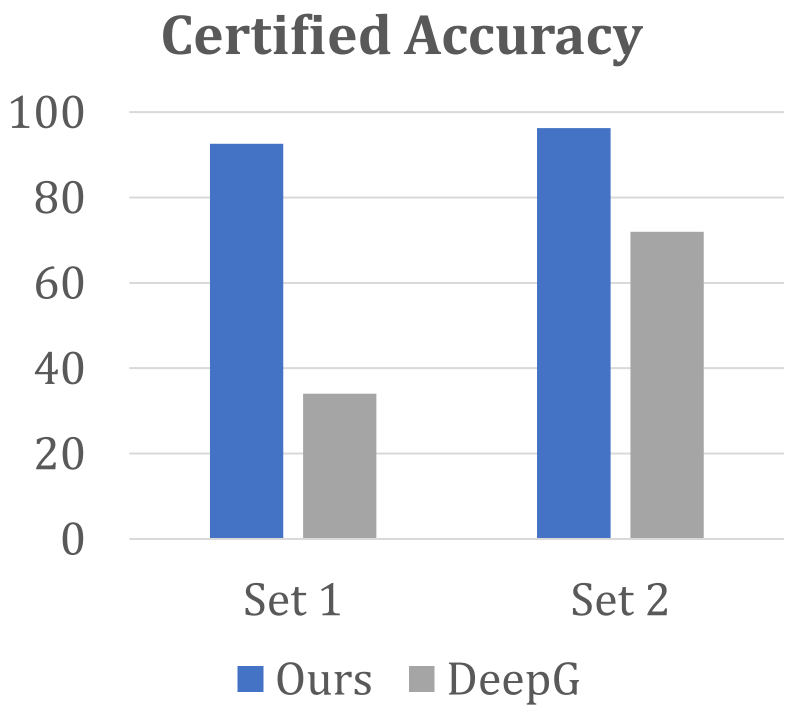
Minimize both the lower and upper bounds' distances to the ground truth

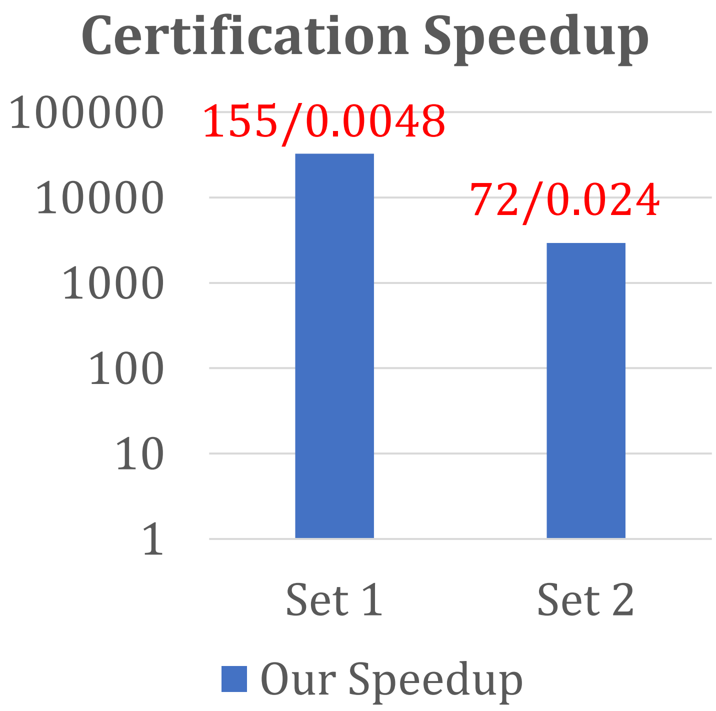
$$\frac{\ell\left(\underline{f}\left(P\left(x, \tilde{\theta}_l\right)\right), y\right) + \ell\left(\overline{f}\left(P\left(x, \tilde{\theta}_l\right)\right), y\right)}{2}$$

certified lower bound   certified upper bound

MNIST

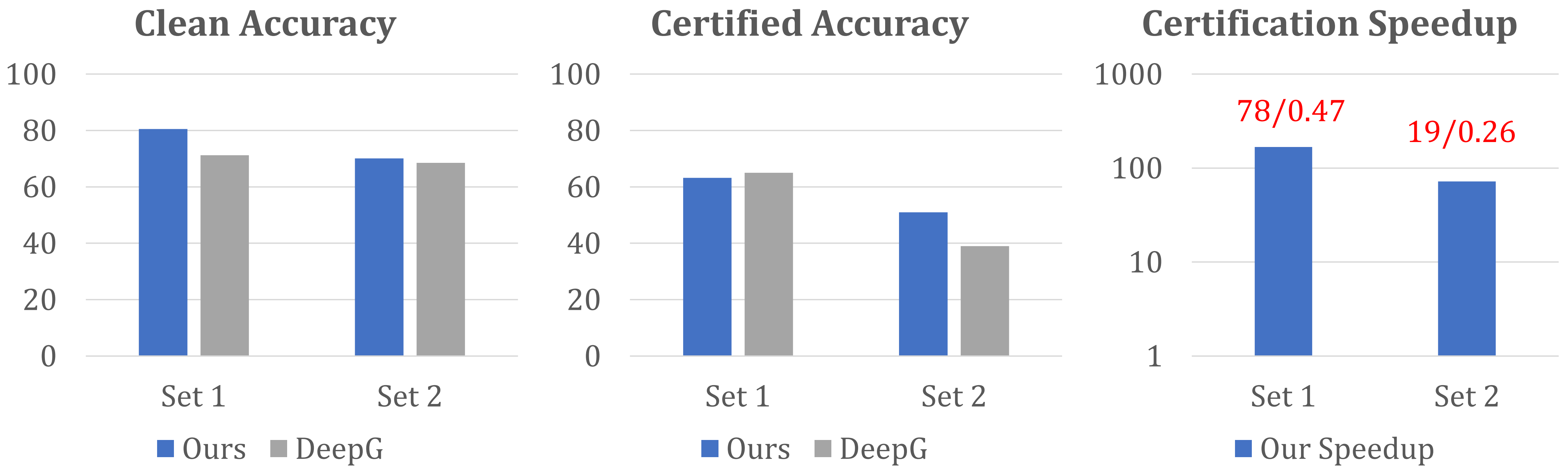






Set 1: Scale(5%), Rotate(5°), Contrast(5%), Brightness(0.01)
Set 2: Shear(2%), Rotate(2°), Scale(2%), Contrast(2%), Brightness(0.001)

CIFAR-10



Set 1: Rotate(10°)
Set 2: Rotate(2°), Shear(2%)

Tiny ImageNet

First results for deterministic certified geometric robustness on CIFAR-10+

Transforms	Accuracy (%)	Certified (%)	Certification Time per Image (s)
Shear(2%)	35.5	25.7	0.214
Scale(2%)	33.1	21.3	0.205
Rotate(5°)	32.2	17.4	1.006

In ℓ_∞-norm setting with $\epsilon = \frac{1}{255}$, 27.8% accuracy and 15.9% certified robustness in Xu et al., 2020

Udacity Self-Driving

Task: predict steering angle from $3 \times 66 \times 200$ driving scene image

Network: 9-layer convolutional network from Bojarski et al., 2016

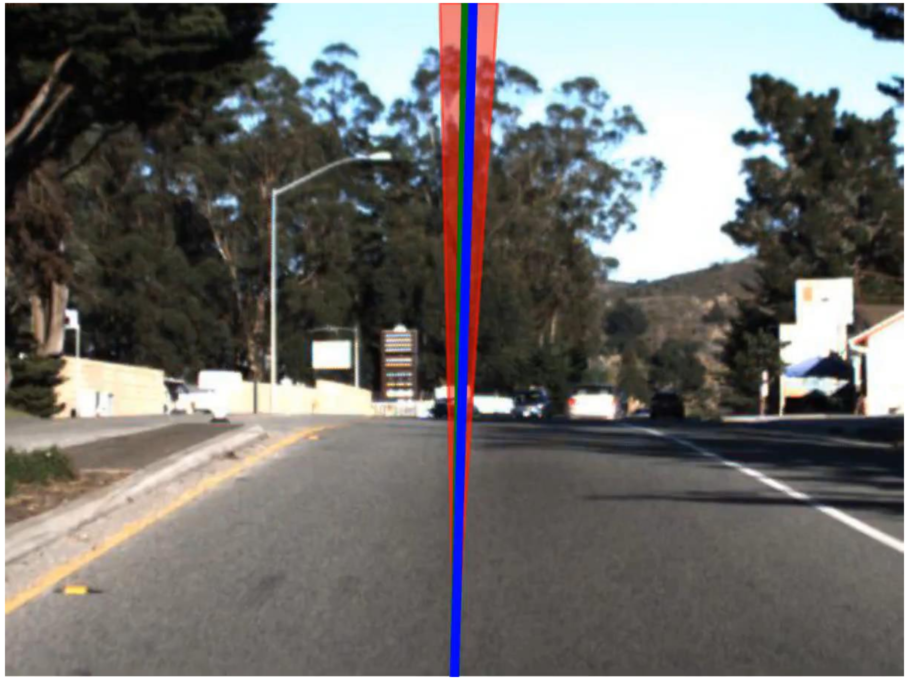
Transformation: $\pm 2°$ rotation

Udacity Self-Driving

Task: predict steering angle from $3 \times 66 \times 200$ driving scene image

Network: 9-layer convolutional network from Bojarski et al., 2016

Transformation: $\pm 2°$ rotation



Green: ground truth label
Blue: network prediction
Red: certified bound

Udacity Self-Driving

Certified training can **help** network performance

Training Method	MAE	Certified MAE	Certification Time per Image (s)
Regular	6.07°	97.56°	0.11
Dropout	4.85°	96.65°	0.12
Ours	**5.36°**	**8.05°**	**0.11**

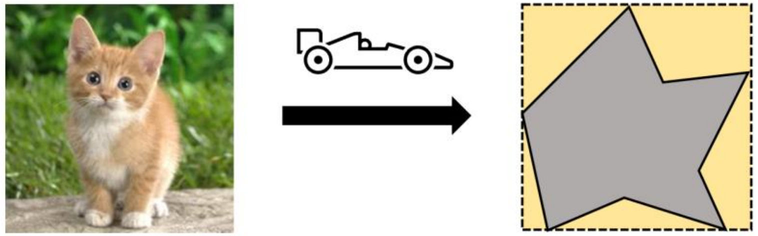
Conclusion

Robust geometric loss

$$\ell\left(\hat{f}\left(P(x, \tilde{\theta}_l)\right), y\right)$$

$\theta \sim \mathcal{U}(\underline{\theta}, \overline{\theta})$ and $\tilde{\theta}_l = [\theta - \nu, \theta + \nu]$

Fast geometric verifier



x $\quad$ $\tilde{x} = P(x, \tilde{\theta})$

Poster session 1

11:30 – 13:30
@ MH1-2-3-4 #155

Code