

Reliable and Interpretable Artificial Intelligence

Lecture 9: Certified Robustness to Geometric Transformations

Martin Vechev

ETH Zurich

Fall 2020

Beyond L_p robustness

Limitations of threat model based on L_p perturbations:

- Less likely to occur naturally in **real world** scenarios
- There are many transformations which **preserve semantic meaning** of the original image while **not being covered** by a small L_p ball

Geometric adversarial examples



revolver



mousetrap

Natural transformations such as rotation and translation are enough to cause **misclassification**

Geometric Transformations

We represent geometric transformation as a bijective function $T_{\kappa}: \mathbb{R}^2 \rightarrow \mathbb{R}^2$

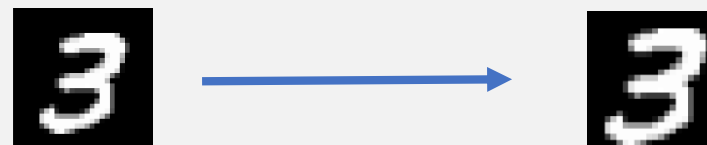
Rotation: $T_{\phi}(x, y) = (x\cos(\phi) - y\sin(\phi), x\sin(\phi) + y\cos(\phi))$



Translation: $T_{\delta_x, \delta_y}(x, y) = (x + \delta_x, y + \delta_y)$



Scaling: $T_{\lambda}(x, y) = (\lambda x, \lambda y)$



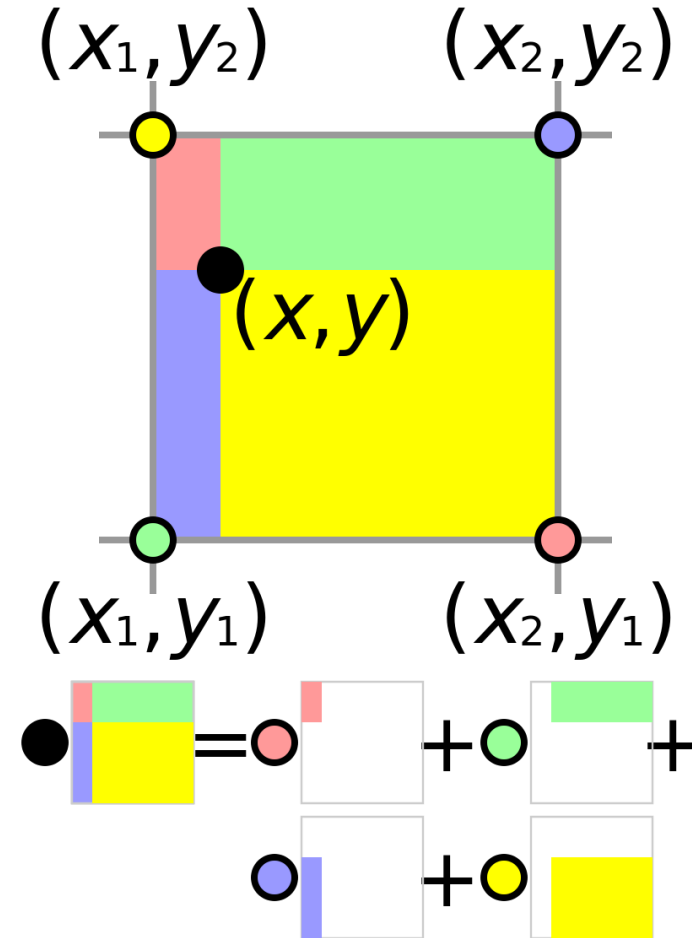
Interpolation

Pixel values in the original image are defined only at **integer** coordinates

To compute pixel value at **non-integer** coordinate (x, y) we perform *bilinear interpolation* $I: \mathbb{R}^2 \rightarrow \mathbb{R}$,

$$\begin{aligned} I(x, y) = & p_{x_1, y_1} (x_2 - x)(y_2 - y) \\ & + p_{x_1, y_2} (x_2 - x)(y - y_1) \\ & + p_{x_2, y_1} (x - x_1)(y_2 - y) \\ & + p_{x_2, y_2} (x - x_1)(y - y_1) \end{aligned}$$

where $x_1 \leq x \leq x_2, y_1 \leq y \leq y_2$
and $x_2 = x_1 + 1, y_2 = y_1 + 1$.



Credits: Wikipedia

Computing pixel values

To compute pixel value at the coordinate (x, y) after transformation T_{κ} we need to:

- 1) compute the preimage of the point (x, y) under the transformation T_{κ}
- 2) interpolate the resulting coordinate to obtain the pixel value

Computing pixel values

To compute pixel value at the coordinate (x, y) after transformation T_{κ} we need to:

- 1) compute the preimage of the point (x, y) under the transformation T_{κ}
- 2) interpolate the resulting coordinate to obtain the pixel value

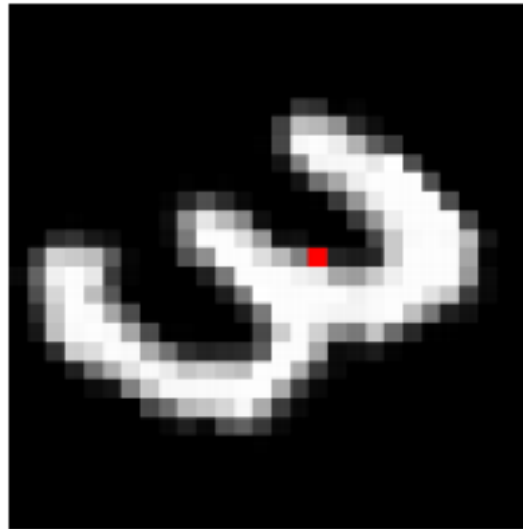
We define function $I_{\kappa} : \mathbb{R}^2 \rightarrow \mathbb{R}$
as a composition of interpolation I and inverse of
geometric transformation T parametrized by κ

$$I_{\kappa}(x, y) = I \circ T_{\kappa}^{-1}(x, y)$$

An example of I_{κ}



Original image



Rotated image by $\pi/4$

Computing pixel value at the coordinate (5, 1) shown in red:

1. Apply inverse rotation R^{-1} to the point (5, 1):

$$A = R_{\pi/4}^{-1} (5, 1) = (2\sqrt{2}, 3\sqrt{2})$$

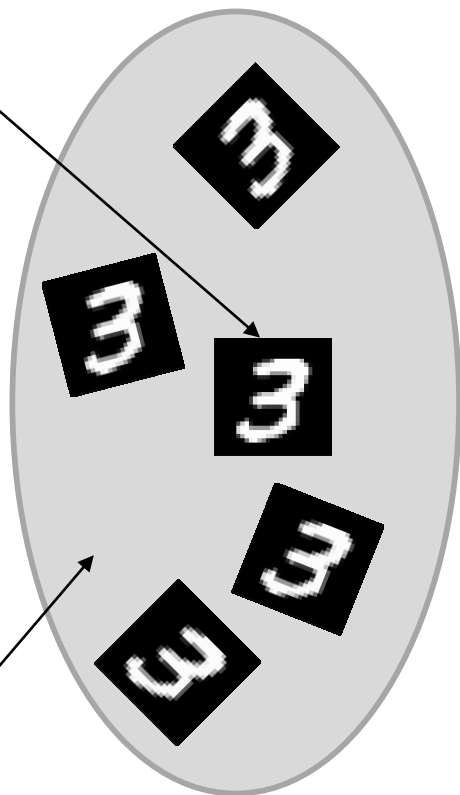
2. As $(2\sqrt{2}, 3\sqrt{2})$ is at non-integer coordinate we need to perform interpolation $I(2\sqrt{2}, 3\sqrt{2})$ which results in the pixel value **0.3**

Our goal now will be to certify geometric robustness

Certifying Geometric Robustness of Neural Networks, NeurIPS'2019
Balunovic, Baader, Singh, V

Certifying geometric robustness

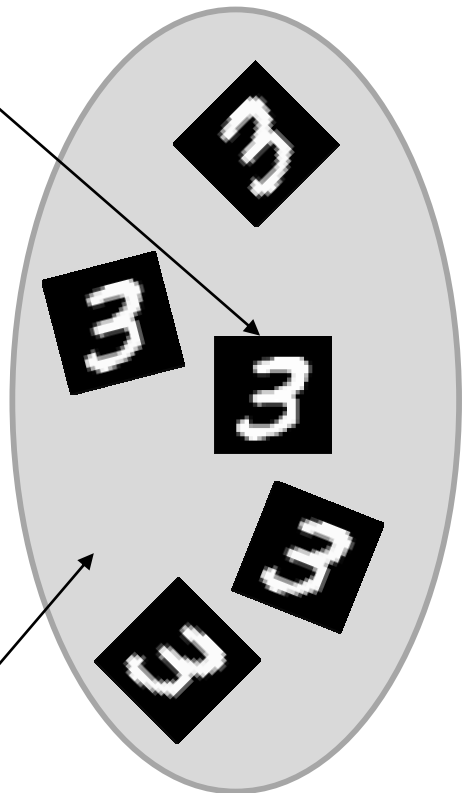
Original image O



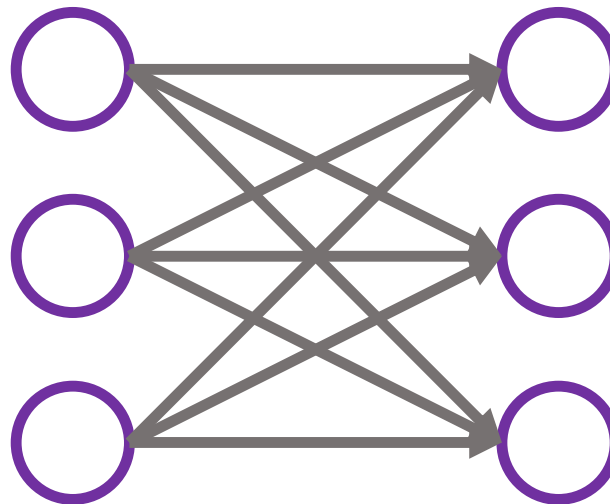
Exact region
 $R_\phi(O)$ for $\phi \in [-30, 30]$

Certifying geometric robustness

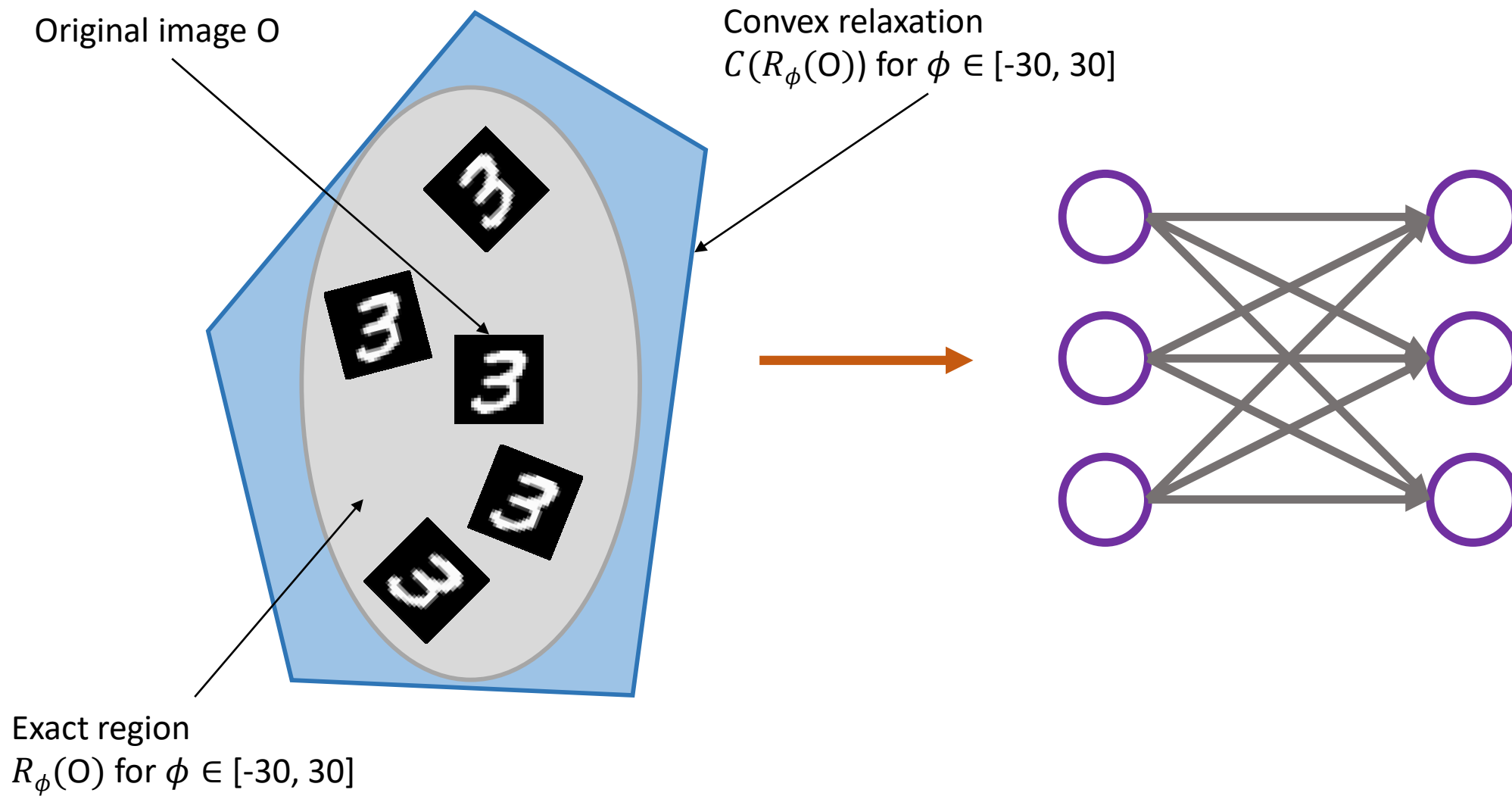
Original image O



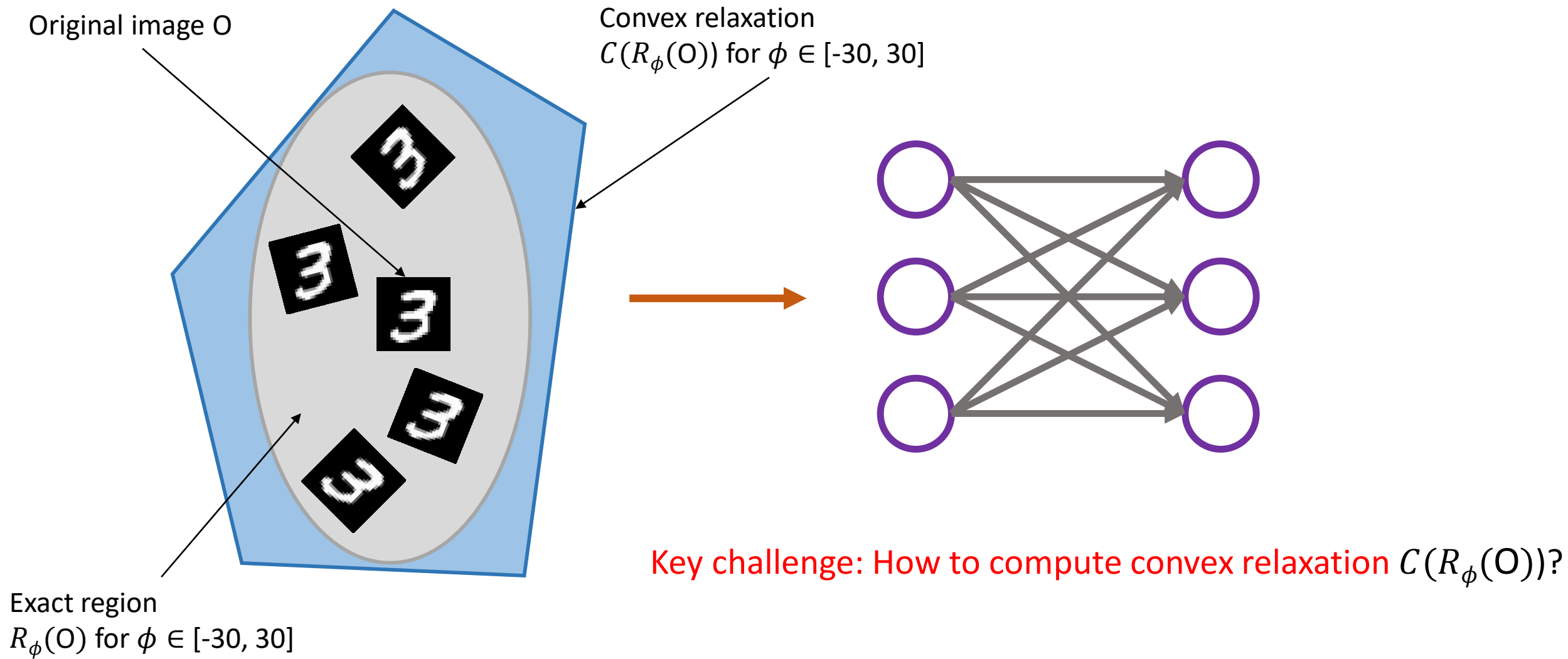
Exact region
 $R_\phi(O)$ for $\phi \in [-30, 30]$



Certifying geometric robustness



Certifying geometric robustness



DeepPoly constraints for capturing $\mathcal{C}(R_\phi(\mathcal{O}))$

- To certify robustness, we will use the DeepPoly convex relaxation shape (but not its transformers – we will see why later).
- **A possible baseline:** we can compute interval bounds on $I_\kappa(x, y)$ by pushing hyperrectangle κ via Box to compute lower and upper bounds for the pixel value.
- For geometric transformations, there is relationship between pixels through the transformation parameters κ . As interval bounds do **not capture** this relationship, we can benefit from a more precise relaxation than Box.

Sound lower and upper constraints

To compute **sound** lower and upper constraints we want to find a pair of hyperplanes (\mathbf{w}_l, b_l) , (\mathbf{w}_u, b_u) which satisfy

$$\mathbf{w}_l^T \boldsymbol{\kappa} + b_l \leq I_{\kappa}(x, y) \leq \mathbf{w}_u^T \boldsymbol{\kappa} + b_u$$

for all $\boldsymbol{\kappa}$ in the parameter space D .

Sound lower and upper constraints

To compute **sound** lower and upper constraints we want to find a pair of hyperplanes (\mathbf{w}_l, b_l) , (\mathbf{w}_u, b_u) which satisfy

$$\mathbf{w}_l^T \boldsymbol{\kappa} + b_l \leq I_{\boldsymbol{\kappa}}(x, y) \leq \mathbf{w}_u^T \boldsymbol{\kappa} + b_u$$

for all $\boldsymbol{\kappa}$ in the parameter space D .

The key challenge is to find **sound** constraints which are actually reasonably tight (we also need to define how to measure tightness)

Measure of tightness for a convex relaxation

2-dim case (reminder): DeepPoly relaxation for ReLU has minimum **area** in the 2-dimensional input-output plane

Measure of tightness for a convex relaxation

2-dim case (reminder): DeepPoly relaxation for ReLU has minimum **area** in the 2-dimensional input-output plane

K-dim case: Measure **volume** between relaxation and the exact function


$$L(\mathbf{w}_l, b_l) := \int_{\boldsymbol{\kappa} \in D} \left(I_{\boldsymbol{\kappa}}(x, y) - (\mathbf{w}_l^T \boldsymbol{\kappa} + b_l) \right) d\boldsymbol{\kappa}$$
$$U(\mathbf{w}_u, b_u) := \int_{\boldsymbol{\kappa} \in D} \left((\mathbf{w}_u^T \boldsymbol{\kappa} + b_u) - I_{\boldsymbol{\kappa}}(x, y) \right) d\boldsymbol{\kappa}$$

Two Optimization problems

Find \mathbf{w}_l, b_l and \mathbf{w}_u, b_u which minimize the volume

$$L(\mathbf{w}_l, b_l) := \int_{\boldsymbol{\kappa} \in D} \left(I_{\boldsymbol{\kappa}}(x, y) - (\mathbf{w}_l^T \boldsymbol{\kappa} + b_l) \right) d\boldsymbol{\kappa}$$

Can't even compute
the objective function




$$U(\mathbf{w}_u, b_u) := \int_{\boldsymbol{\kappa} \in D} \left((\mathbf{w}_u^T \boldsymbol{\kappa} + b_u) - I_{\boldsymbol{\kappa}}(x, y) \right) d\boldsymbol{\kappa}$$

subject to the soundness constraints:

$$\mathbf{w}_l^T \boldsymbol{\kappa} + b_l \leq I_{\boldsymbol{\kappa}}(x, y) \leq \mathbf{w}_u^T \boldsymbol{\kappa} + b_u, \forall \boldsymbol{\kappa} \in D$$

Need to ensure constraint
holds for uncountable
set of $\boldsymbol{\kappa}$ values



Approximation of optimization problem

Step 1: Replace the **intractable** objective with a Monte Carlo **approximation**:

$$L(\mathbf{w}_l, b_l) \approx \frac{1}{N} \sum_{i=1}^N \left(I_{\kappa^i} - (\mathbf{w}_l^T \kappa^i + b_l) \right)$$

Step 2: Replace the **uncountable** set of constraints with **finite** set of constraints:

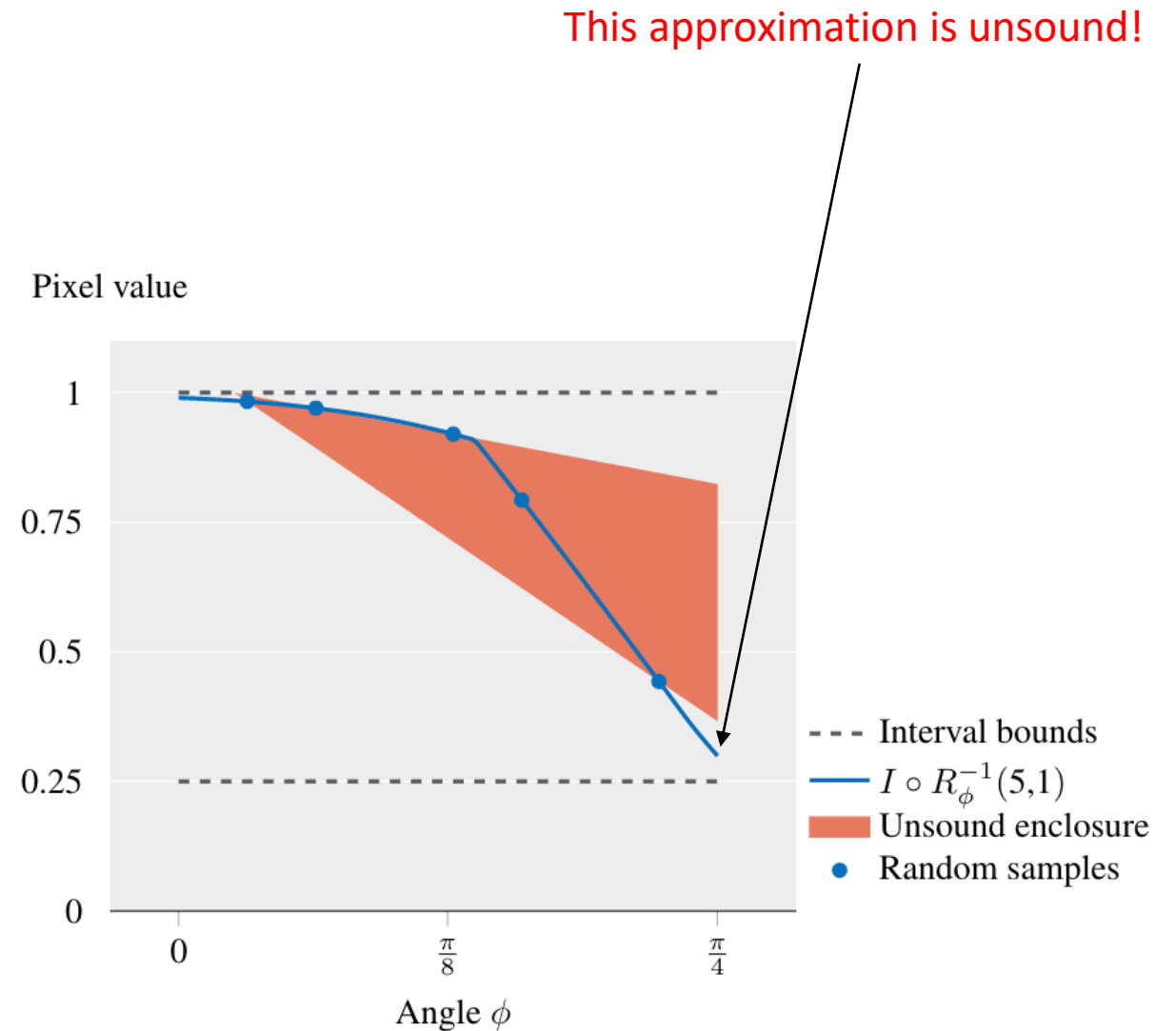
$$\mathbf{w}_l^T \kappa^i + b_l \leq I_{\kappa^i}(x, y), \forall i \in \{1, 2, \dots, N\}$$

We can solve this exactly in polynomial time using linear programming (LP) and obtain approximate solutions to the original problem $\hat{\mathbf{w}}_l, \hat{b}_l$

Running example

We sample random angle points for our parameter $\phi \in [0, \pi/4]$ and evaluate $I \circ R_\phi^{-1}(5, 1)$ to obtain pixel values, shown as blue points.

Solving the LP in yields $\hat{b}_l = 1.07$ and $\hat{w}_l = -0.9$. Together with the upper constraint, it forms the orange enclosure in the figure.



Computing sound constraints

So far, our constraints are sound at a finite set of points:

$$\hat{\mathbf{w}}_l^T \boldsymbol{\kappa}^i + \hat{b}_l \leq I_{\boldsymbol{\kappa}^i}(x, y) \leq \hat{\mathbf{w}}_u^T \boldsymbol{\kappa}^i + \hat{b}_u, \forall i \in \{1, 2, \dots, N\}$$

Computing sound constraints

So far, our constraints are sound at a finite set of points:

$$\hat{\mathbf{w}}_l^T \boldsymbol{\kappa}^i + \hat{b}_l \leq I_{\boldsymbol{\kappa}^i}(x, y) \leq \hat{\mathbf{w}}_u^T \boldsymbol{\kappa}^i + \hat{b}_u, \forall i \in \{1, 2, \dots, N\}$$

We would like to modify the constraints so that they are sound on the entire parameter space D :

$$\mathbf{w}_l^T \boldsymbol{\kappa} + b_l \leq I_{\boldsymbol{\kappa}}(x, y) \leq \mathbf{w}_u^T \boldsymbol{\kappa} + b_u, \forall \boldsymbol{\kappa} \in D$$

Computing sound constraints

So far, our constraints are sound at a finite set of points:

$$\hat{\mathbf{w}}_l^T \boldsymbol{\kappa}^i + \hat{b}_l \leq I_{\boldsymbol{\kappa}^i}(x, y) \leq \hat{\mathbf{w}}_u^T \boldsymbol{\kappa}^i + \hat{b}_u, \forall i \in \{1, 2, \dots, N\}$$

We would like to modify the constraints so that they are sound on the entire parameter space D :

$$\mathbf{w}_l^T \boldsymbol{\kappa} + b_l \leq I_{\boldsymbol{\kappa}}(x, y) \leq \mathbf{w}_u^T \boldsymbol{\kappa} + b_u, \forall \boldsymbol{\kappa} \in D$$

Key idea:

Suppose we have an upper bound δ on the **maximum soundness violation** on the entire parameter space D

$$\begin{aligned} (\hat{\mathbf{w}}_l^T \boldsymbol{\kappa} + \hat{b}_l) - I_{\boldsymbol{\kappa}}(x, y) &\leq \delta_l, \forall \boldsymbol{\kappa} \in D \\ I_{\boldsymbol{\kappa}}(x, y) - (\hat{\mathbf{w}}_u^T \boldsymbol{\kappa} + \hat{b}_u) &\leq \delta_u, \forall \boldsymbol{\kappa} \in D \end{aligned}$$

Then, the constraints $\mathbf{w}_l = \hat{\mathbf{w}}_l$, $b_l = \hat{b}_l - \delta_l$ and $\mathbf{w}_u = \hat{\mathbf{w}}_u$, $b_u = \hat{b}_u + \delta_u$ are sound.

Bounding the maximum violation

We need to compute an upper bound to the function f defined as (in domain D):

$$f(\boldsymbol{\kappa}) = (\hat{\mathbf{w}}_l^T \boldsymbol{\kappa} + \hat{b}_l) - I_{\boldsymbol{\kappa}}(x, y)$$

Bounding the maximum violation

We need to compute an upper bound to the function f defined as (in domain D):

$$f(\boldsymbol{\kappa}) = (\hat{\mathbf{w}}_l^T \boldsymbol{\kappa} + \hat{b}_l) - I_{\boldsymbol{\kappa}}(x, y)$$

Option I: bound function f by running box propagation to obtain l, u such that $f(\boldsymbol{\kappa}) \in [l, u], \forall \boldsymbol{\kappa} \in D$. This gives:

$$f(\boldsymbol{\kappa}) \leq u, \forall \boldsymbol{\kappa} \in D$$

Bounding the maximum violation

We need to compute an upper bound to the function f defined as (in domain D):

$$f(\boldsymbol{\kappa}) = (\hat{\mathbf{w}}_l^T \boldsymbol{\kappa} + \hat{b}_l) - I_{\boldsymbol{\kappa}}(x, y)$$

Option I: bound function f by running box propagation to obtain l, u such that $f(\boldsymbol{\kappa}) \in [l, u], \forall \boldsymbol{\kappa} \in D$. This gives:

$$f(\boldsymbol{\kappa}) \leq u, \forall \boldsymbol{\kappa} \in D$$

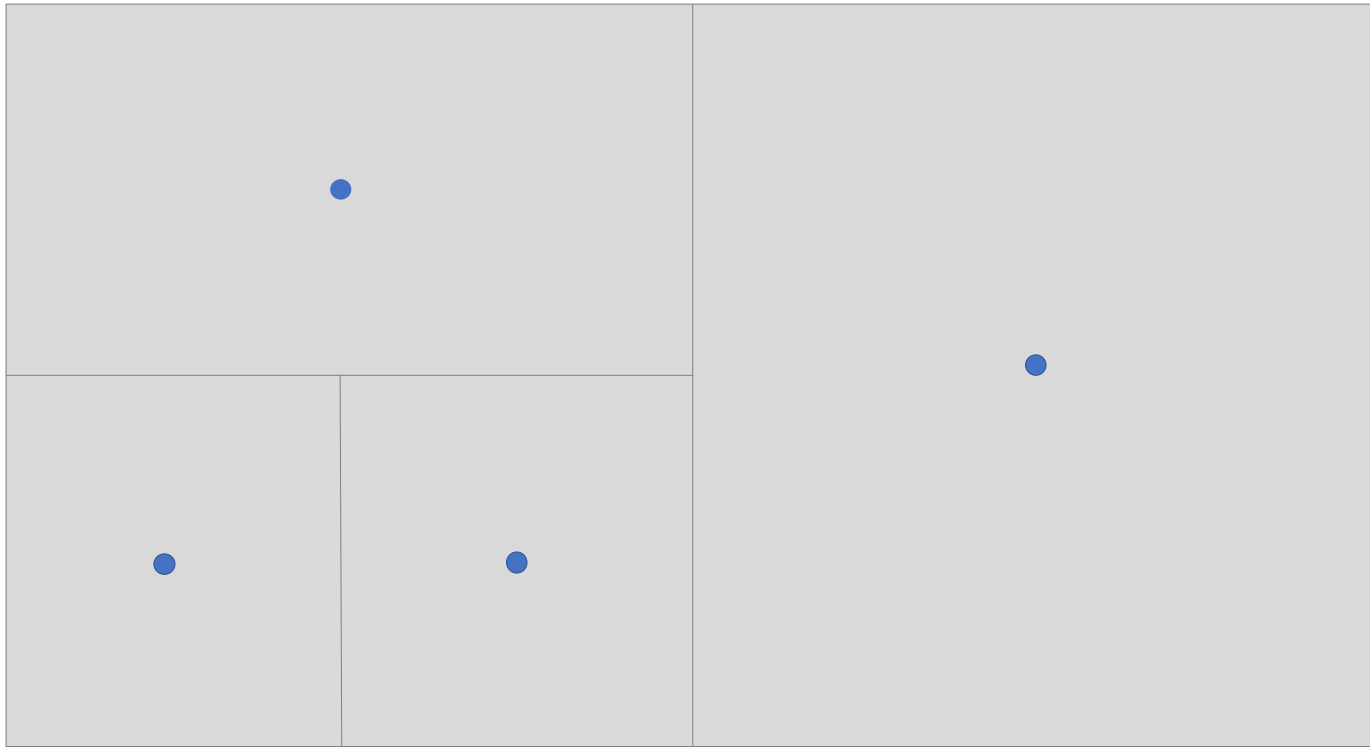
Option II: bound function f using mean-value theorem and Lipschitz continuity:

$$f(\boldsymbol{\kappa}) = f(\boldsymbol{\kappa}_c) + \nabla f(\boldsymbol{\kappa}')^T (\boldsymbol{\kappa} - \boldsymbol{\kappa}_c) \leq f(\boldsymbol{\kappa}_c) + |\mathbf{L}|^T (\boldsymbol{\kappa} - \boldsymbol{\kappa}_c) \leq \underbrace{f\left(\frac{1}{2}(\mathbf{h}_u + \mathbf{h}_l)\right) + \frac{1}{2} |\mathbf{L}|^T (\mathbf{h}_u - \mathbf{h}_l)}_{\text{Assuming } D = [\mathbf{h}_l, \mathbf{h}_u]}$$

where $|\partial_i f(\boldsymbol{\kappa}')| \leq |L_i|$ for any $\boldsymbol{\kappa}' \in D$.

Assuming $D = [\mathbf{h}_l, \mathbf{h}_u]$

Refinement via branch and bound on D



We partition the domain D into a set of hyperrectangles and compute an upper bound for each hyperrectangle

If upper bound in some hyperrectangle is not tight enough, we refine it into smaller rectangles and then continue with the same algorithm

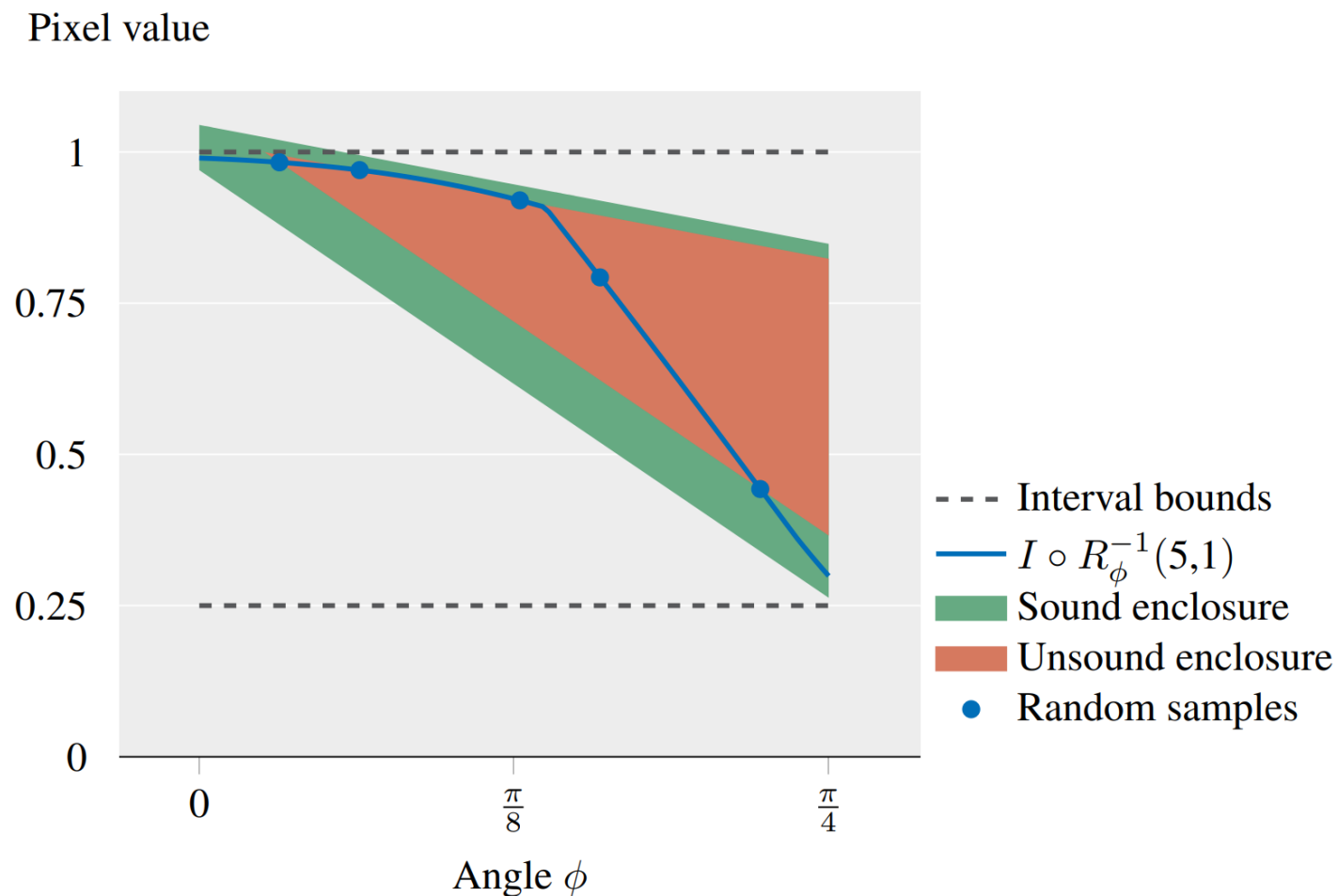
Running example

In the first phase, we computed unsound region (orange); we had $\hat{b}_l = 1.07$ and $\hat{\mathbf{w}}_l = -0.9$.

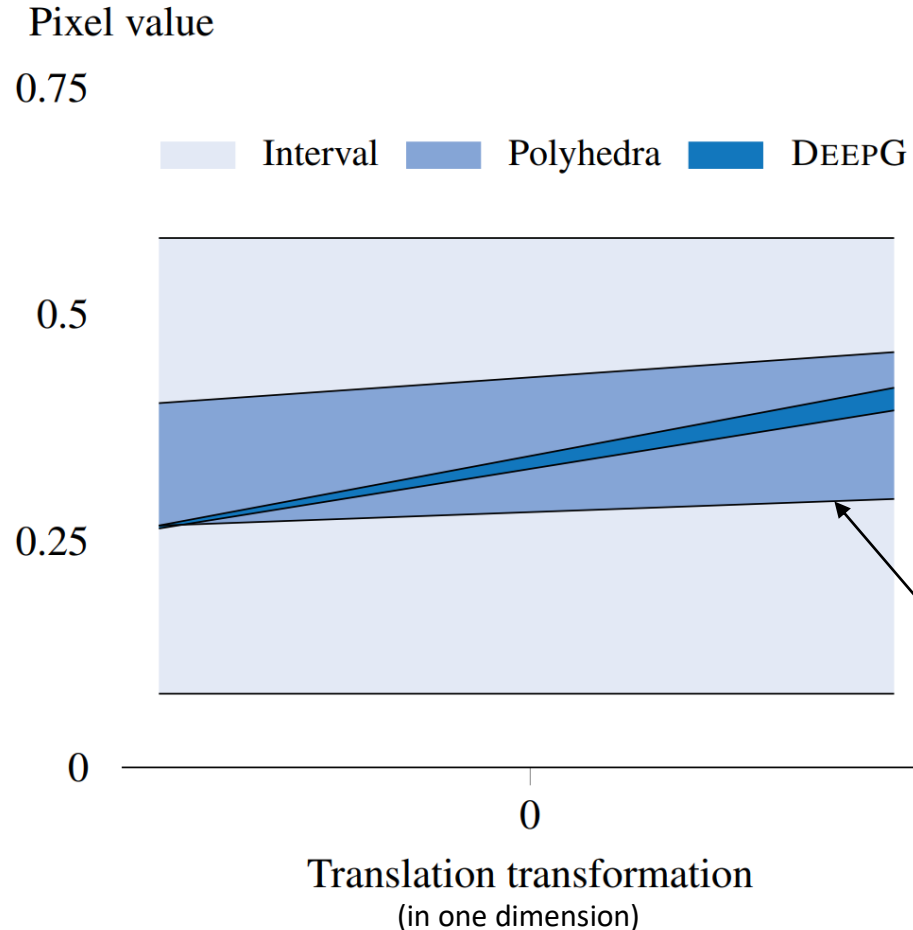
We then compute the maximum violation δ_l as follows:

$$1.07 - 0.9\phi - I\left(R_\phi^{-1}(5, 1)\right) \leq \mathbf{0.1}$$

Sound lower bound is then given by $b_l = \hat{b}_l - 0.1$ and $\mathbf{w}_l = \hat{\mathbf{w}}_l$, shown in green in the figure (together with upper bound).



How well does this work?



To evaluate how tight is our relaxation, we compare:

- **Interval relaxation:** does not capture relationship between pixel and parameters
- **Custom DeepPoly transformers:** the classic approach which computes relaxations one by one for each operation in the transformation.
- **DeepG** [this lecture]: captures relationship and computes the constraint for all operations at the same time

DeepG relaxation is significantly tighter than both DeepPoly and Interval

Certifying Geometric Robustness with DeepPoly

Consider rotation by angle ϕ where $\phi \in [0, 5]$: $I_\phi(x, y) = I \circ R_\phi^{-1}(x, y)$

Prove $x_{11} - x_{12} > 0$ for all $\phi \in [0, 5]$

Certifying Geometric Robustness with DeepPoly

Consider rotation by angle ϕ where $\phi \in [0, 5]$: $I_\phi(x, y) = I \circ R_\phi^{-1}(x, y)$

Prove $x_{11} - x_{12} > 0$ for all $\phi \in [0, 5]$

$$5 \leq x_1 \leq 6.5$$

$$\phi \leq x_1 \leq 1.1\phi + 1$$

$$x_1 = I_\phi(1, 1) \quad \textcircled{x_1}$$

$$5 \leq x_2 \leq 8$$

$$1.2\phi \leq x_2 \leq 1.2\phi + 2$$

$$x_2 = I_\phi(0, 1) \quad \textcircled{x_2}$$

Certifying Geometric Robustness with DeepPoly

Consider rotation by angle ϕ where $\phi \in [0, 5]$: $I_\phi(x, y) = I \circ R_\phi^{-1}(x, y)$

Prove $x_{11} - x_{12} > 0$ for all $\phi \in [0, 5]$

$$5 \leq x_1 \leq 6.5$$

$$\phi \leq x_1 \leq 1.1\phi + 1$$

$$x_1 = I_\phi(1, 1) \quad \textcircled{x_1}$$

$$5 \leq x_2 \leq 8$$

$$1.2\phi \leq x_2 \leq 1.2\phi + 2$$

$$x_2 = I_\phi(0, 1) \quad \textcircled{x_2}$$

Assume that backsubstitution with DeepPoly yields:

$$x_{11} - x_{12} \geq x_2 - x_1 + 1.2$$

Substituting **interval bounds** $x_1 \leq 6.5, x_2 \geq 5$ we get:

$$x_{11} - x_{12} \geq x_2 - x_1 + 1.2 \geq 5 - 6.5 + 1.2 = -0.3$$

Certifying Geometric Robustness with DeepPoly

Consider rotation by angle ϕ where $\phi \in [0, 5]$: $I_\phi(x, y) = I \circ R_\phi^{-1}(x, y)$

Prove $x_{11} - x_{12} > 0$ for all $\phi \in [0, 5]$

$$5 \leq x_1 \leq 6.5$$

$$\phi \leq x_1 \leq 1.1\phi + 1$$

$$x_1 = I_\phi(1, 1) \quad \textcircled{x_1}$$

$$5 \leq x_2 \leq 8$$

$$1.2\phi \leq x_2 \leq 1.2\phi + 2$$

$$x_2 = I_\phi(0, 1) \quad \textcircled{x_2}$$

Assume that backsubstitution with DeepPoly yields:

$$x_{11} - x_{12} \geq x_2 - x_1 + 1.2$$

Substituting **interval bounds** $x_1 \leq 6.5, x_2 \geq 5$ we get:

$$x_{11} - x_{12} \geq x_2 - x_1 + 1.2 \geq 5 - 6.5 + 1.2 = -0.3$$

Substituting **DeepPoly** bounds $x_1 \leq 1.1\phi + 1, x_2 \geq 1.2\phi$ we get:

$$\begin{aligned} x_{11} - x_{12} &\geq x_2 - x_1 + 1.2 \geq 1.2\phi - (1.1\phi + 1) + 1.2 \\ &\geq 0.1\phi - 1 + 1.2 \geq 0.1 \cdot 0 - 1 + 1.2 \\ &= 0.2 \end{aligned}$$

Summary

- We studied the problem of certifying neural networks to geometric transformations.
- Towards that, we presented methods to approximate the values a pixel can take after the transformation (which is in some range), captured via DeepPoly constraints.
- These DeepPoly constraints can then be fed to a standard neural network verifier in order to complete the certification.