Efficiently Training Neural Networks for Verified Robustness

Alessandro De Palma
Imperial College London

Adversarial Examples

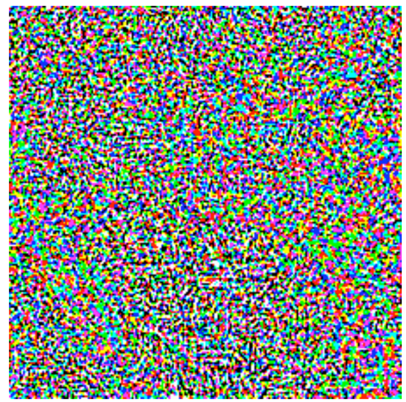
“panda”
57.7% confidence

[Goodfellow et al., 2015]

Adversarial Examples

“panda”
57.7% confidence

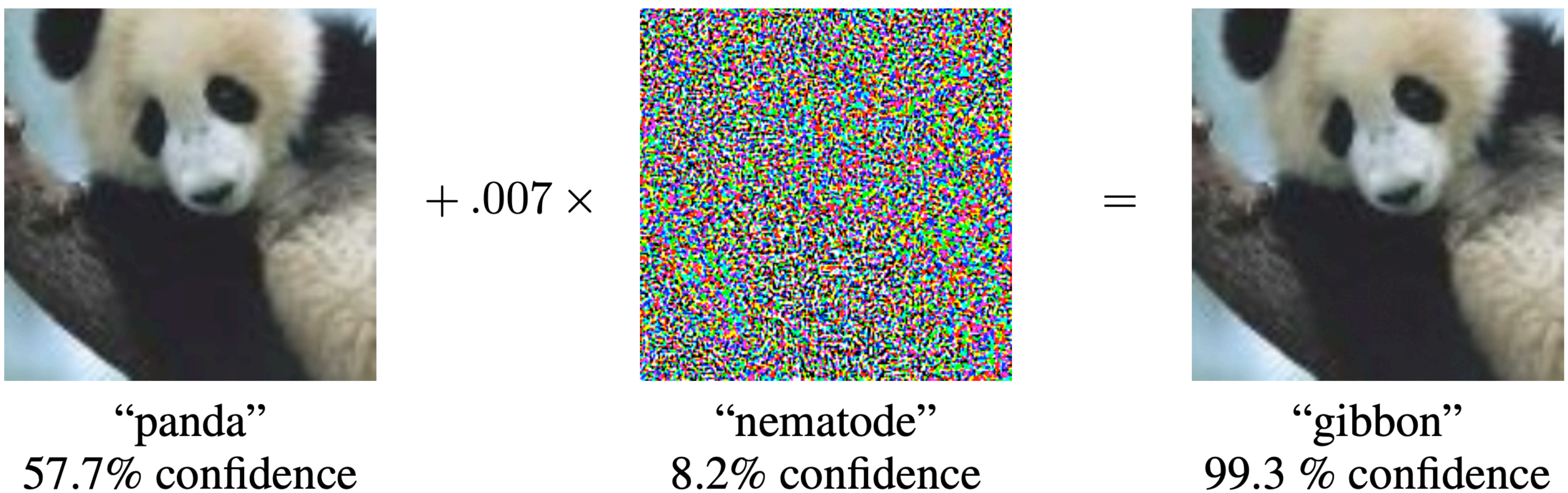
$+ .007 \times$



“nematode”
8.2% confidence

[Goodfellow et al., 2015]

Adversarial Examples

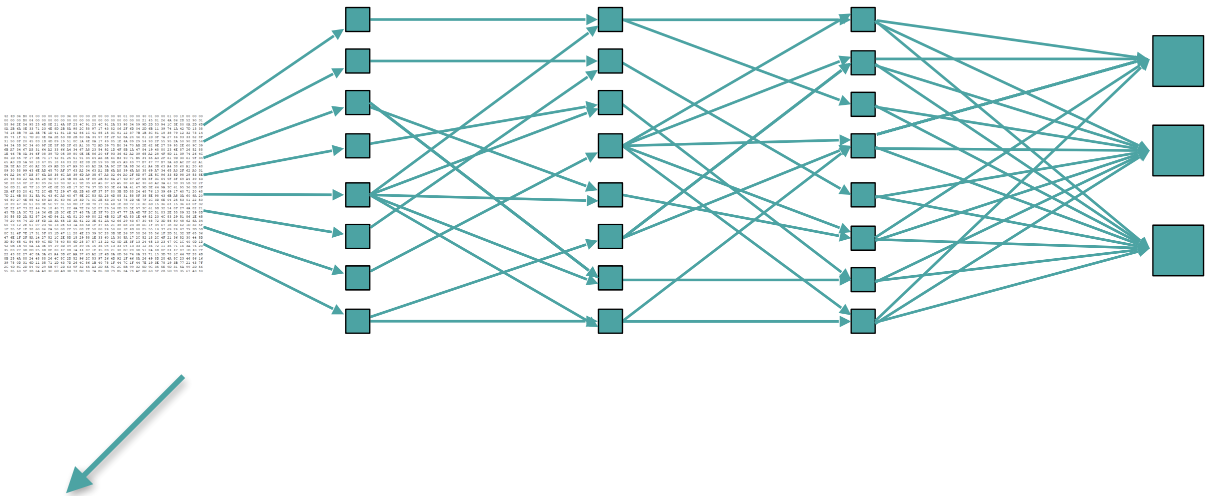


[Goodfellow et al., 2015]

Outline

- **Neural Network Verification**
- Training for Verified Robustness
- NLP?
- Discussion

Neural Network Verification



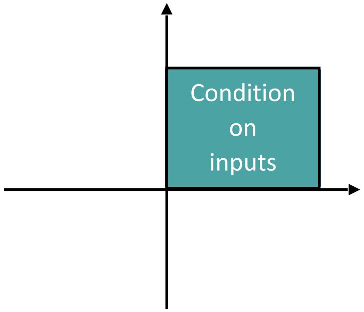


Image Deformations

Neural Network Verification

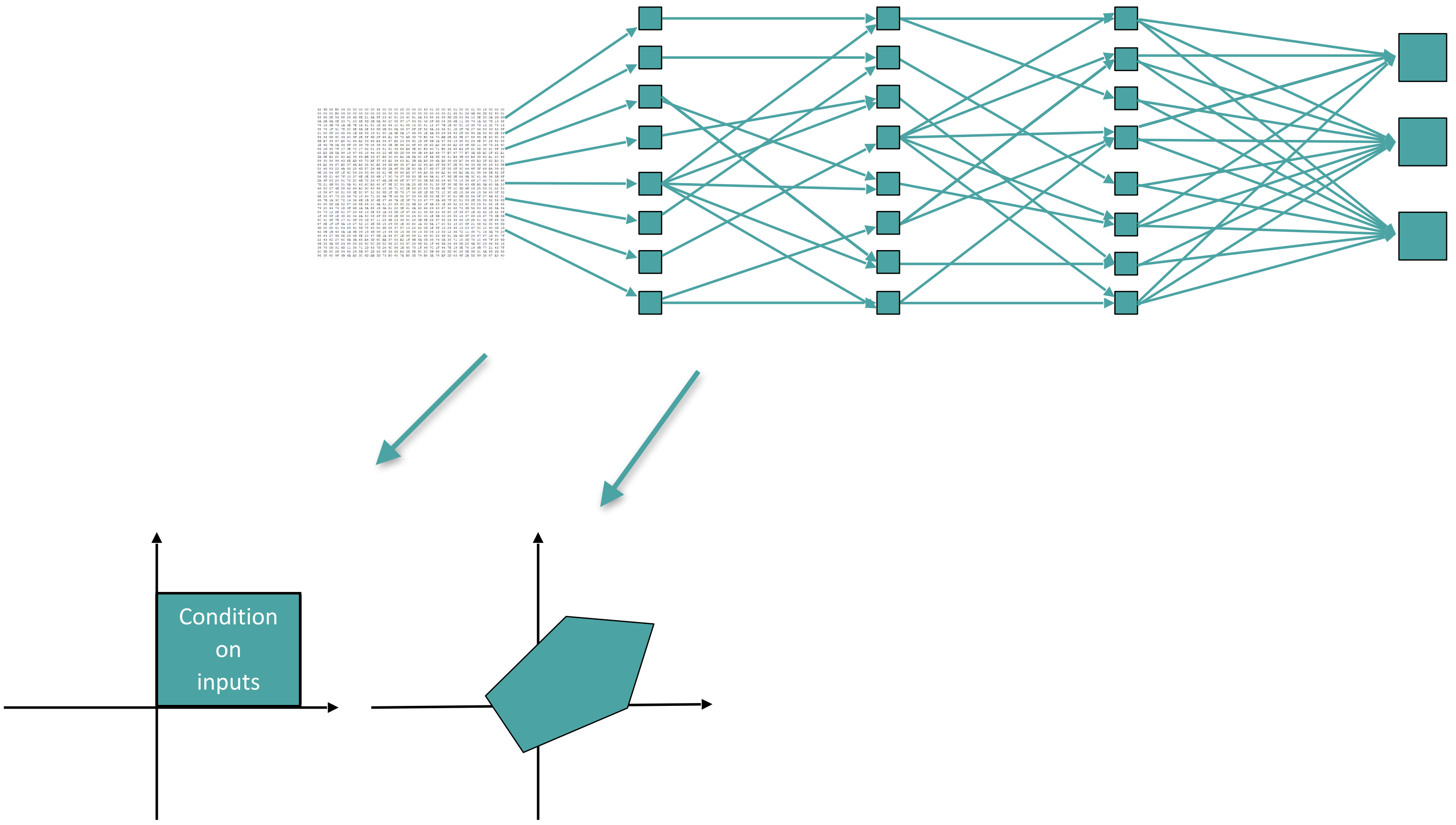


Image Deformations

Neural Network Verification

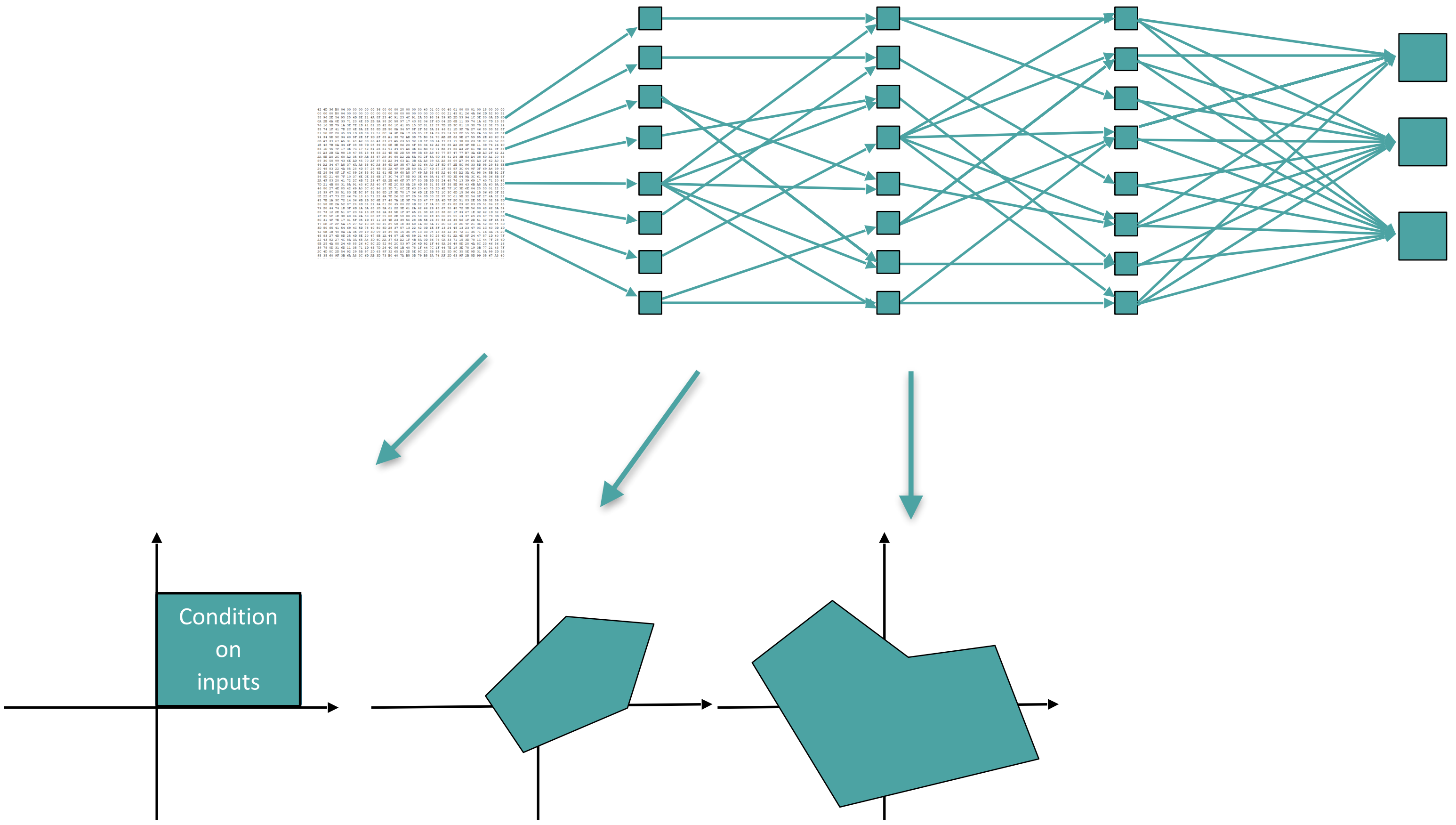


Image Deformations

Neural Network Verification

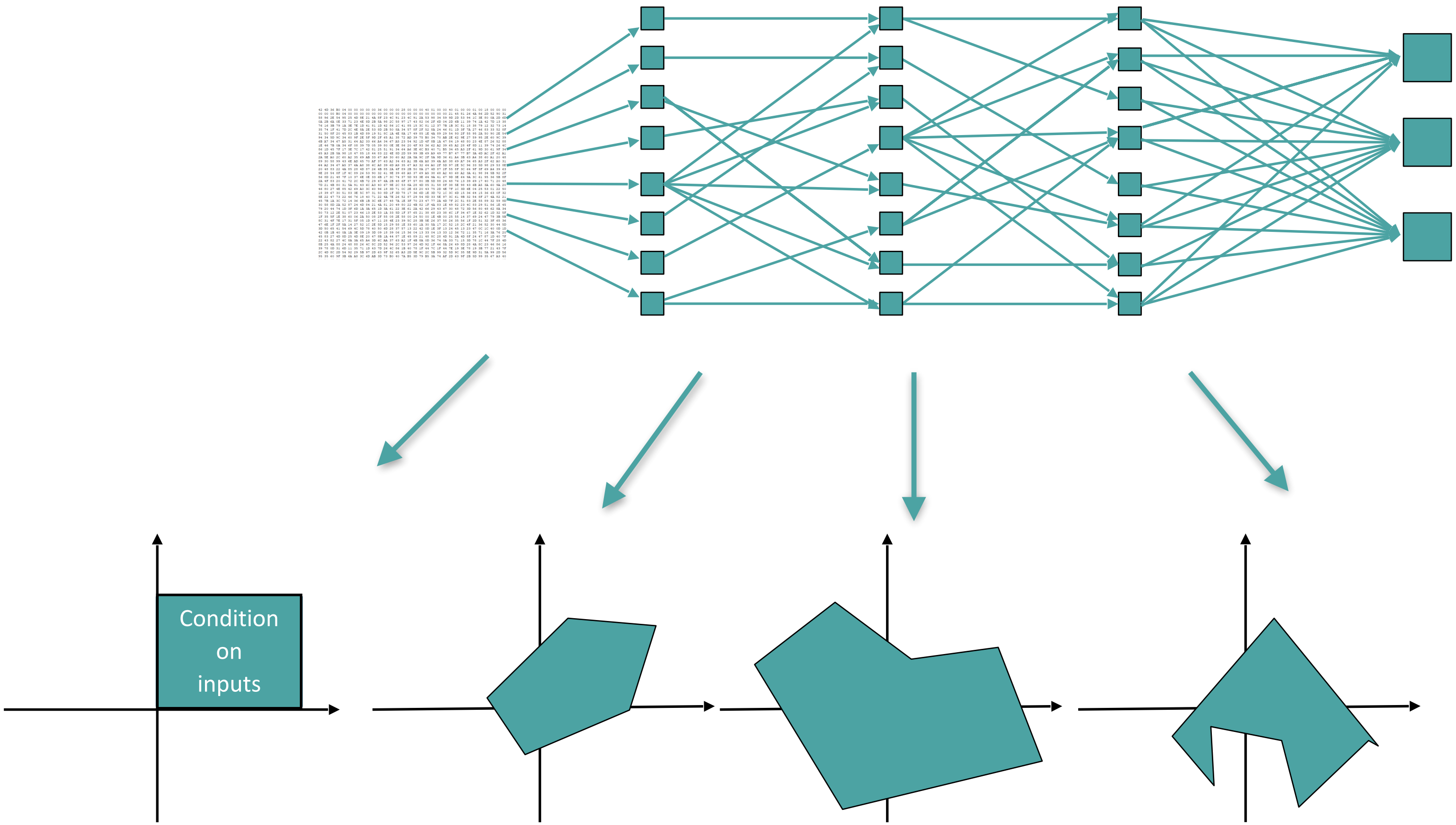


Image Deformations

Neural Network Verification

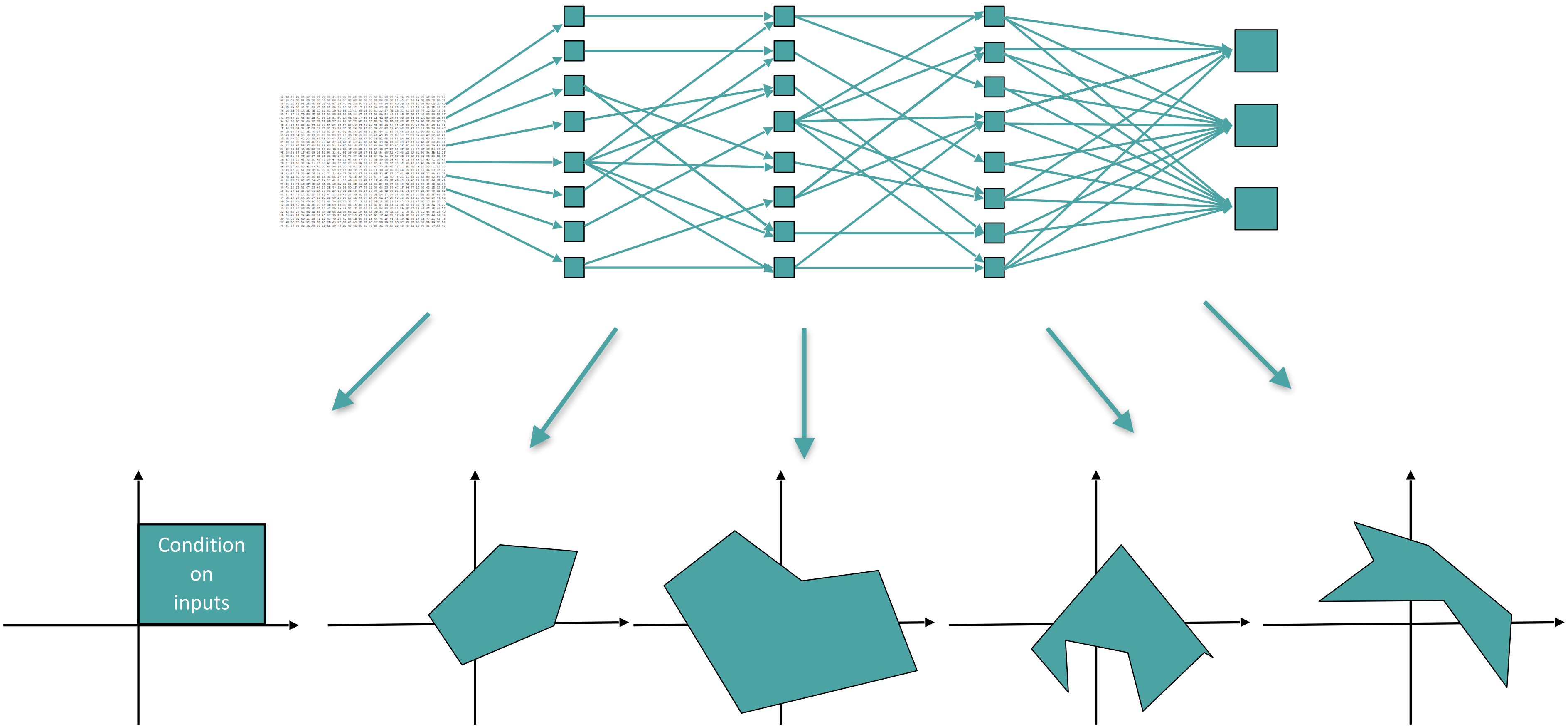
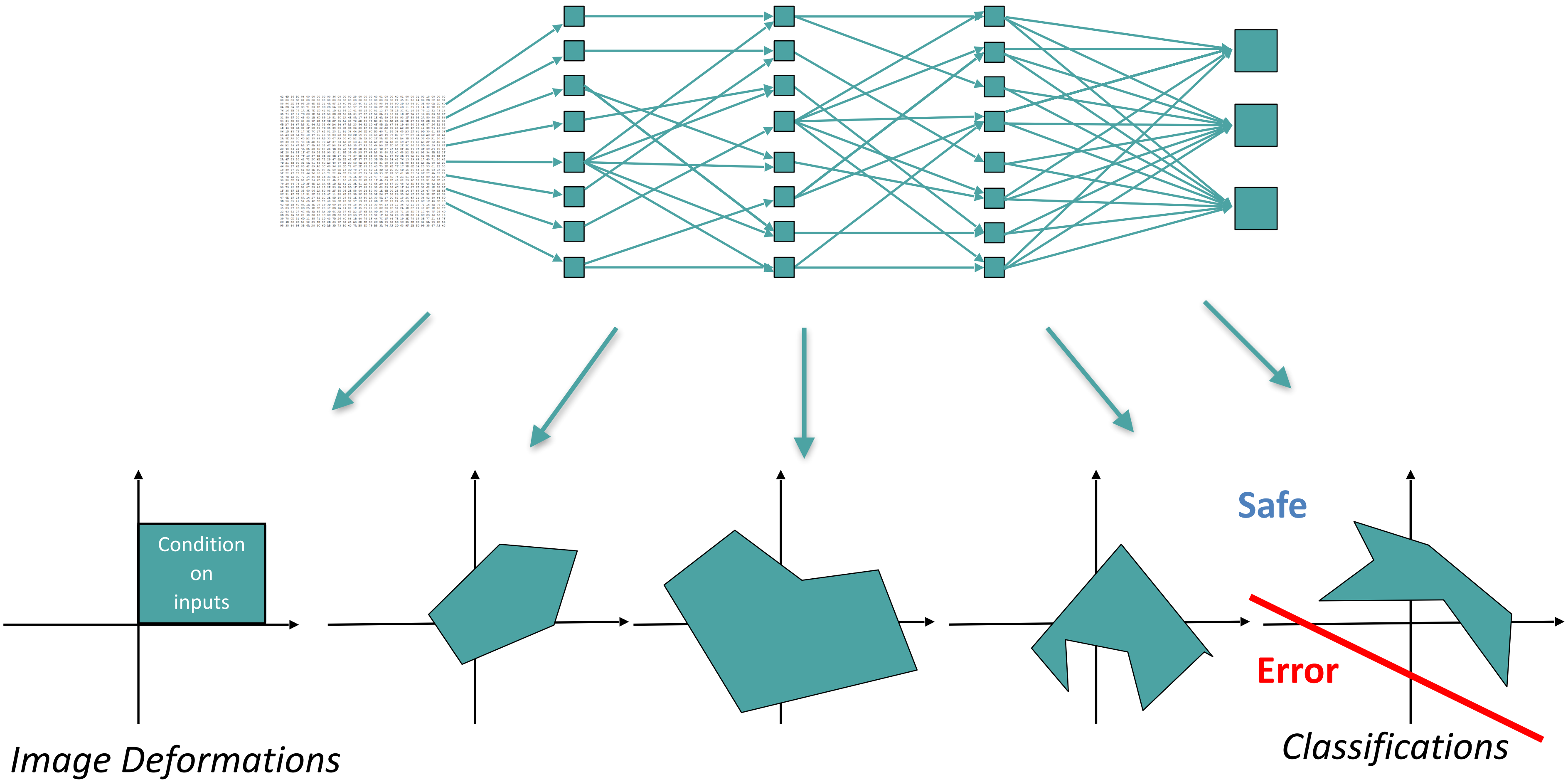


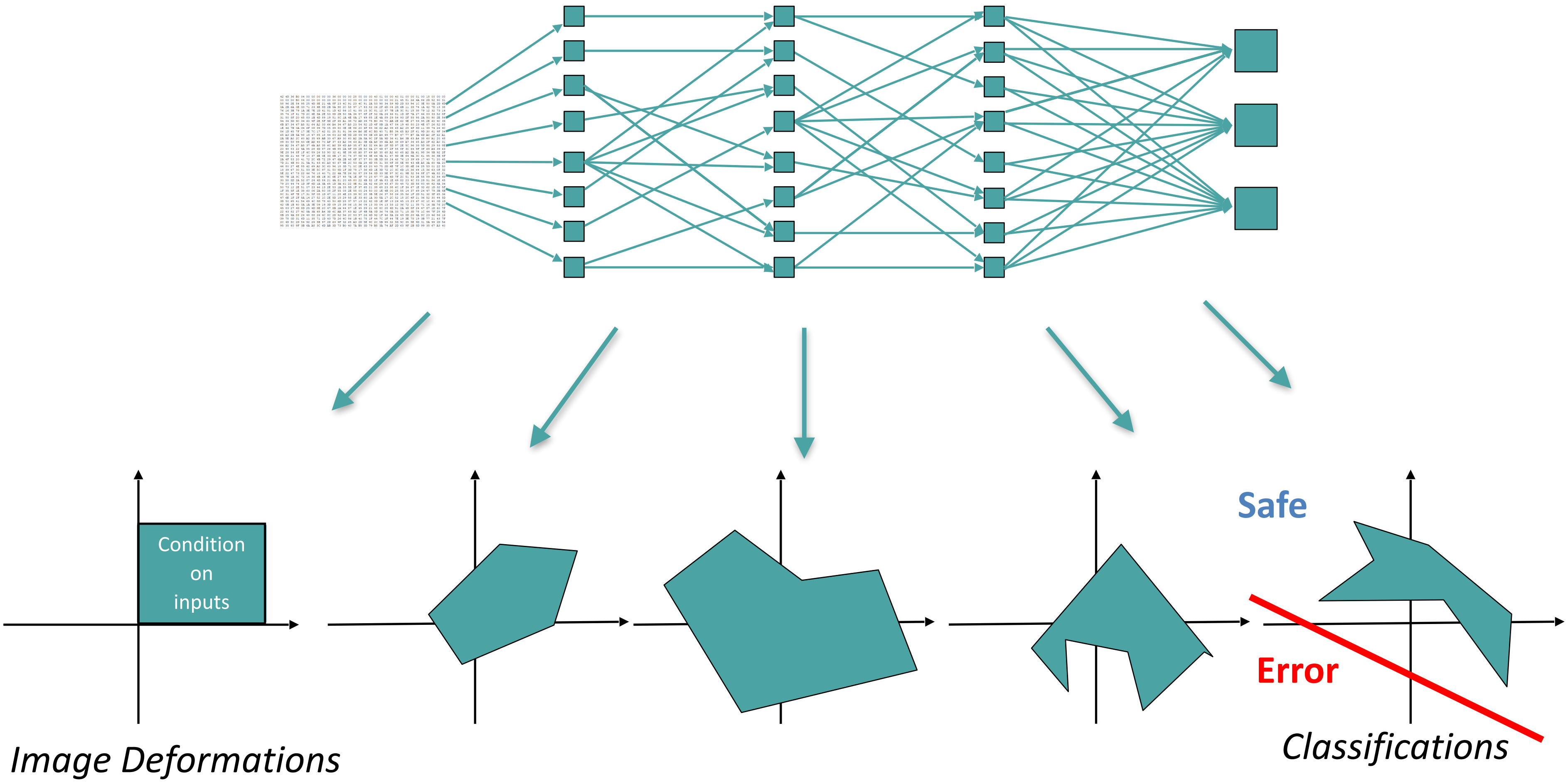
Image Deformations

Neural Network Verification



Property is satisfied

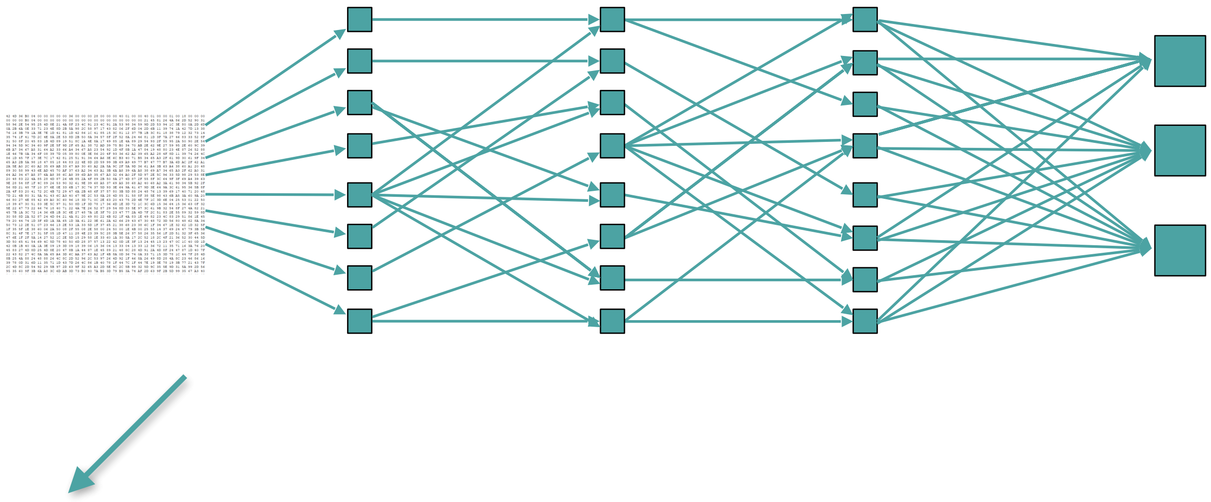
Neural Network Verification



Property is satisfied

NP-HARD

Branch and Bound: Bounding



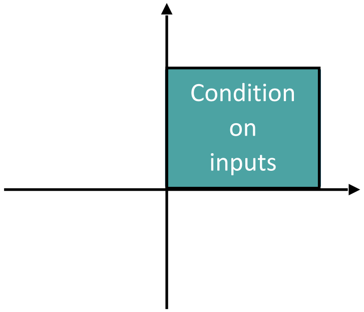


Image Deformations

Branch and Bound: Bounding

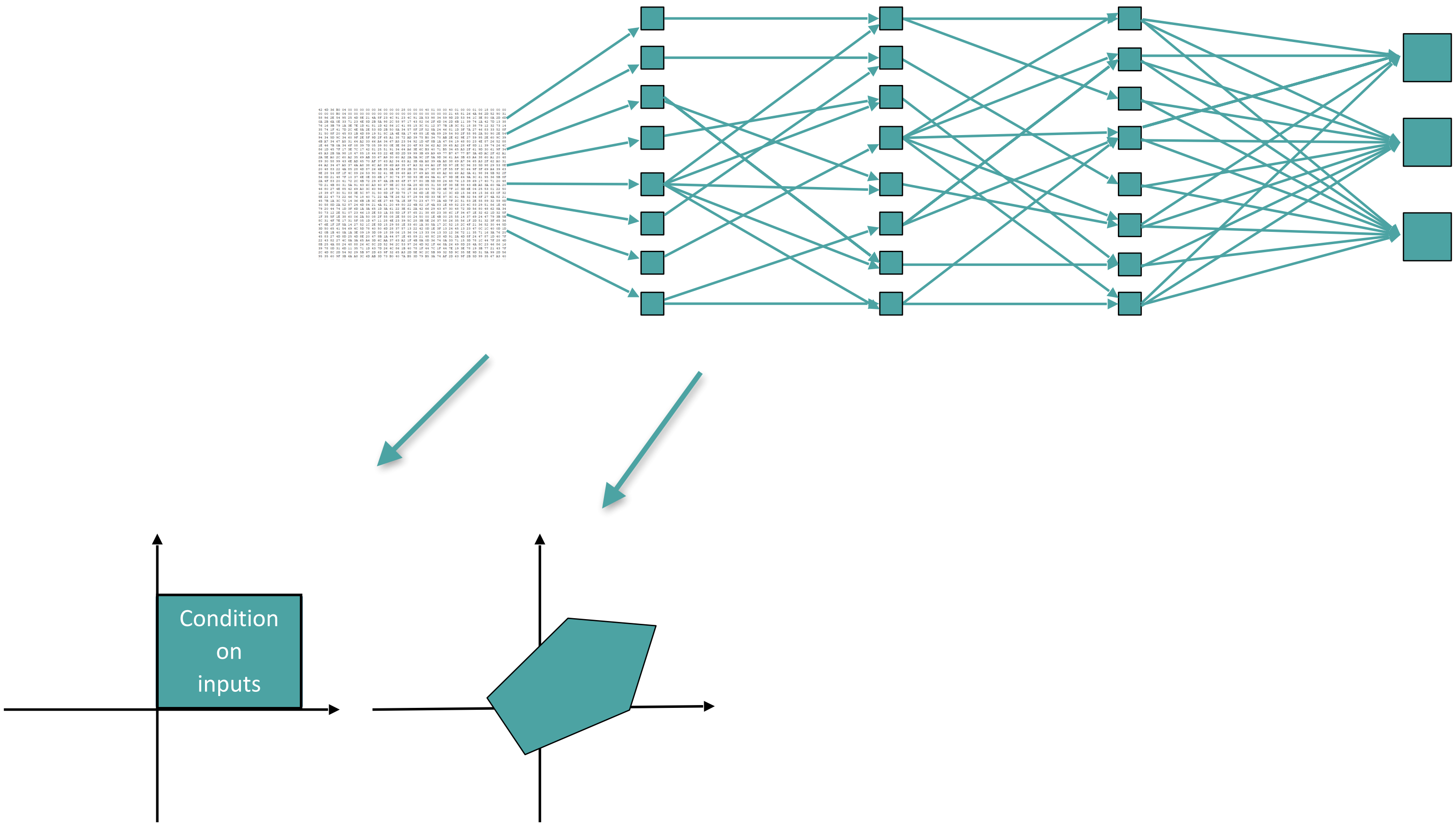


Image Deformations

Branch and Bound: Bounding

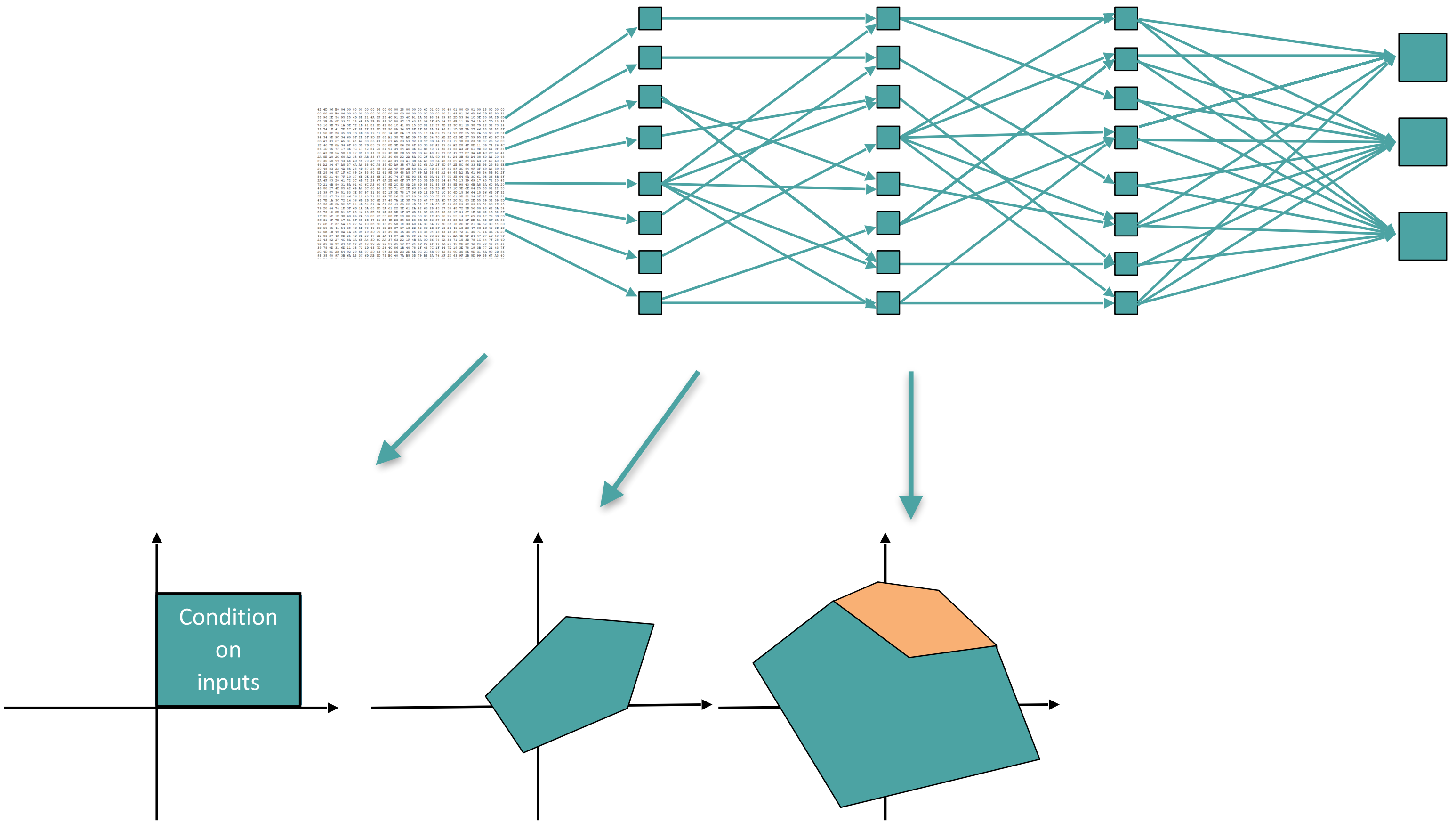
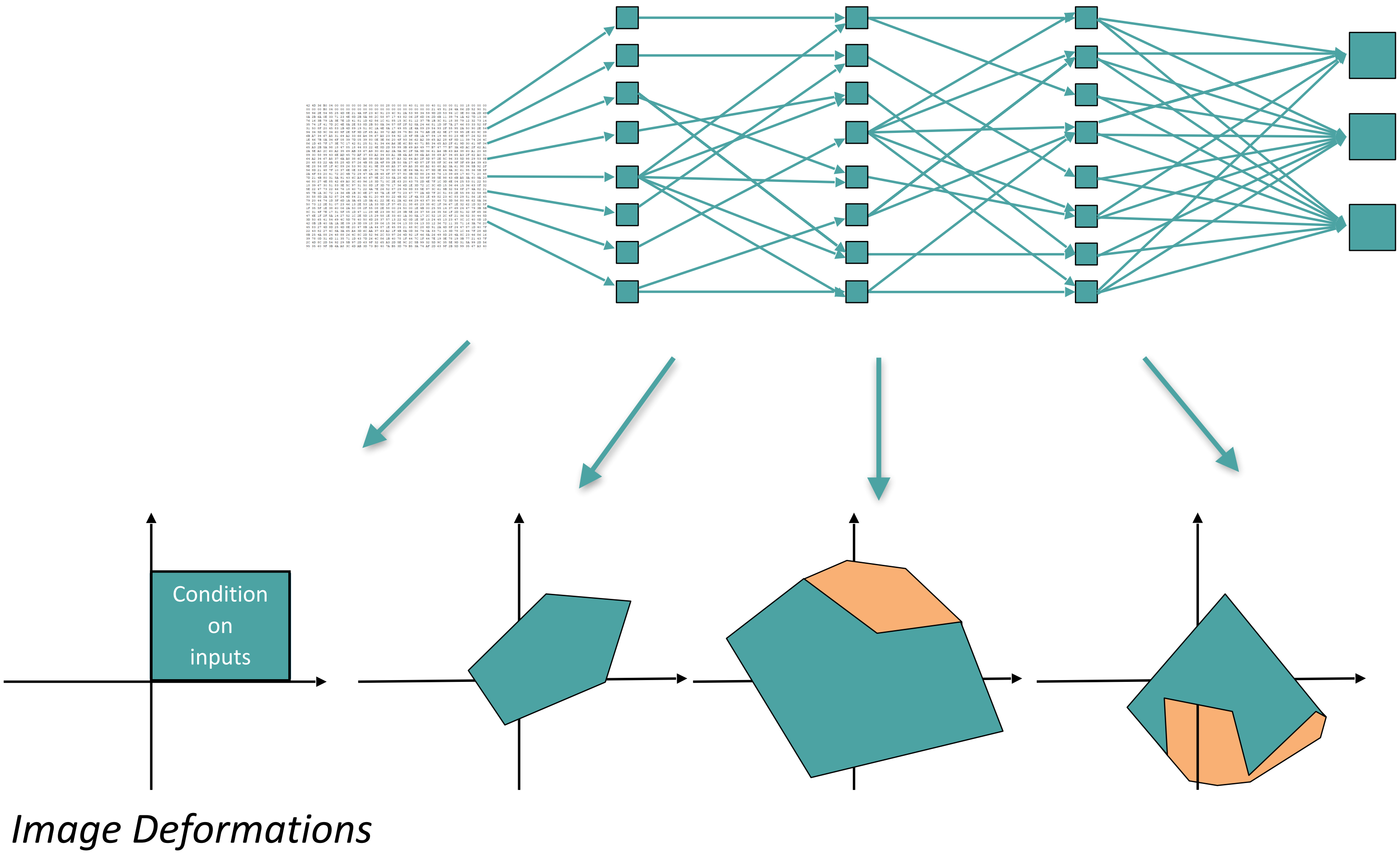


Image Deformations

Branch and Bound: Bounding



Branch and Bound: Bounding

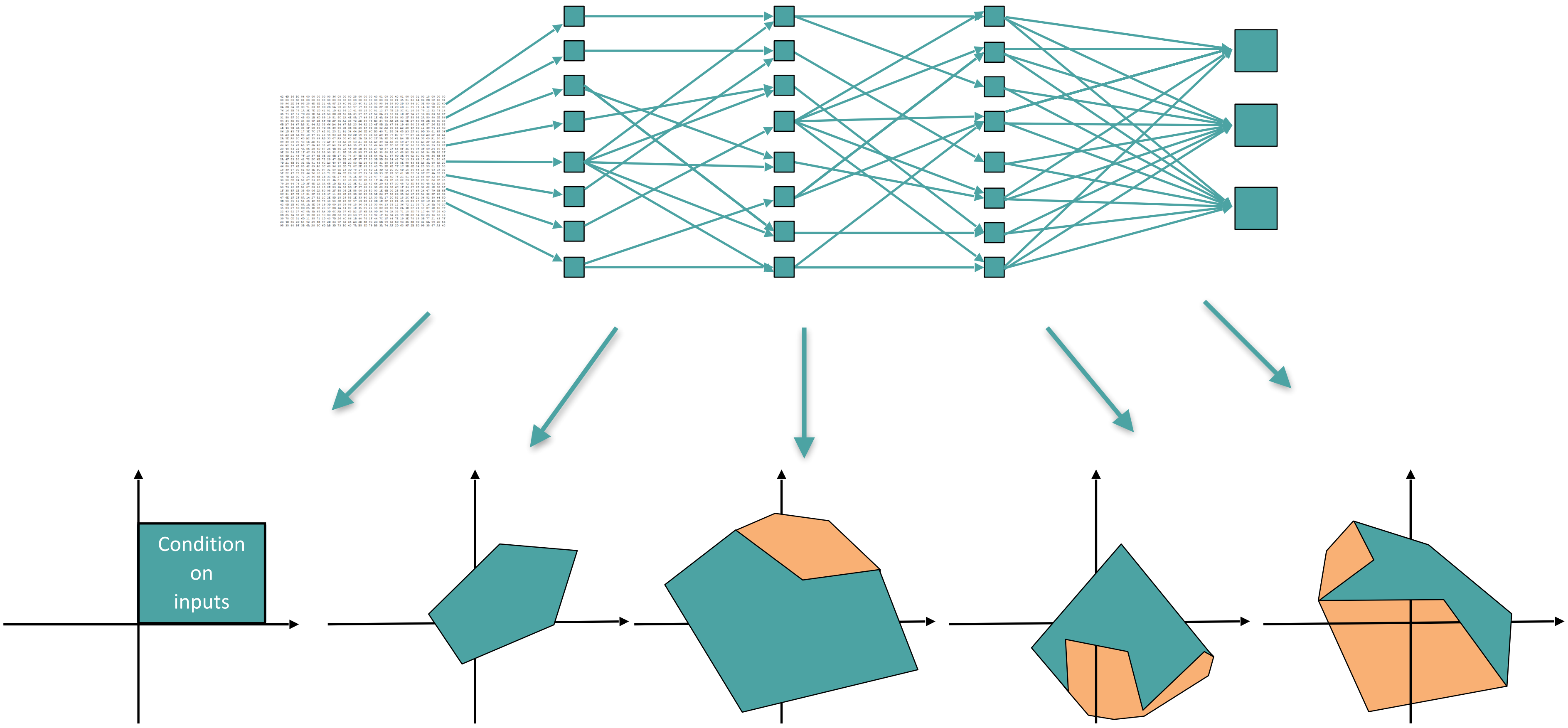


Image Deformations

Branch and Bound: Bounding

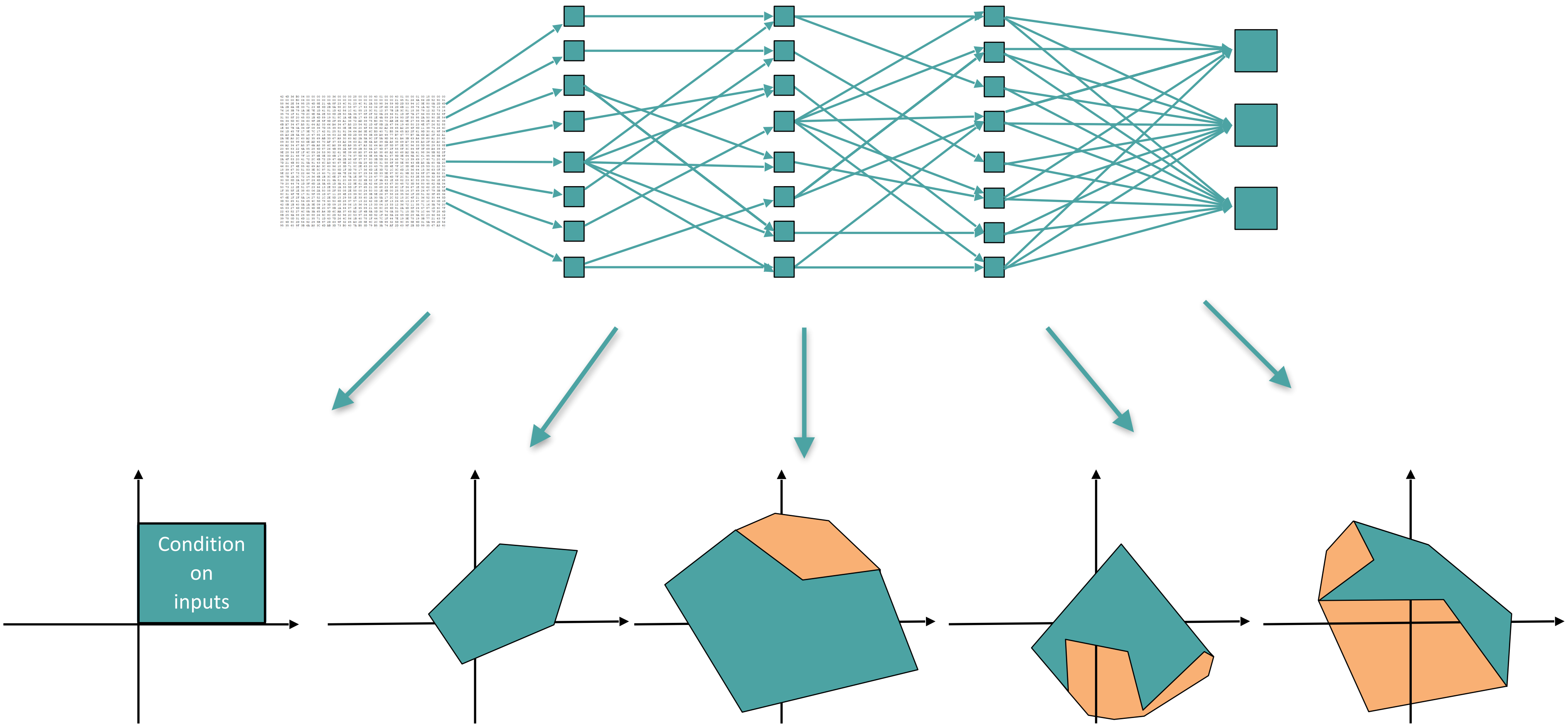
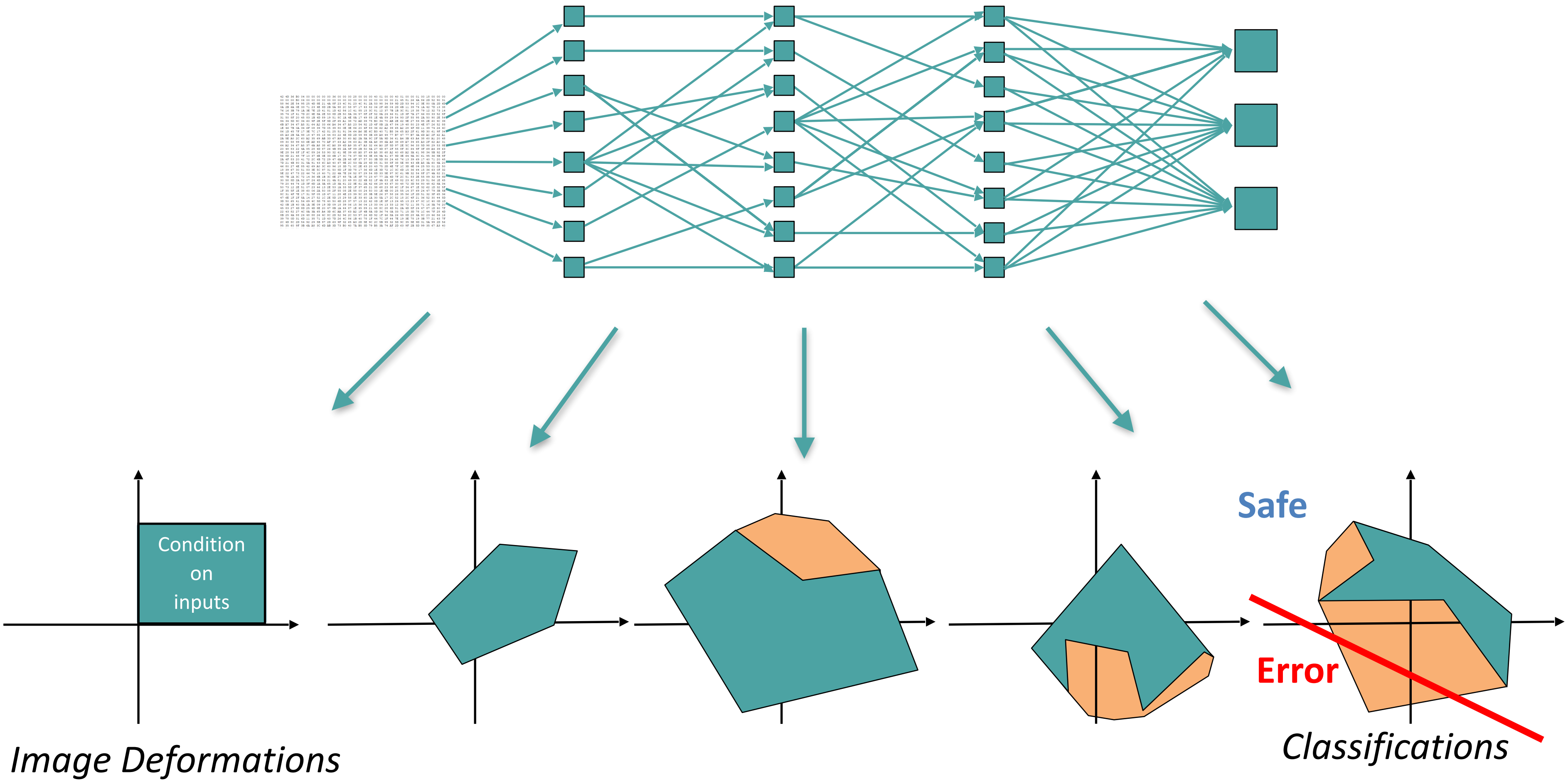


Image Deformations

Verify <u>subset</u> of properties

Branch and Bound: Bounding



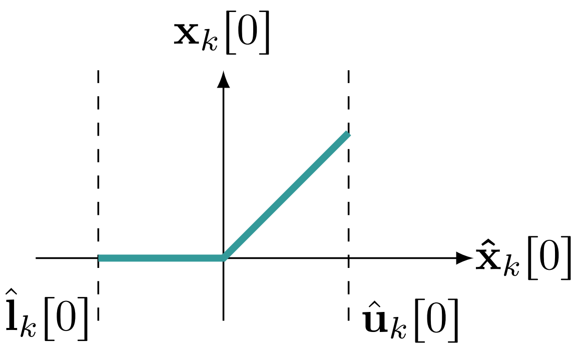
Verify subset of properties

Property is not satisfied

Convex Relaxation: Planet

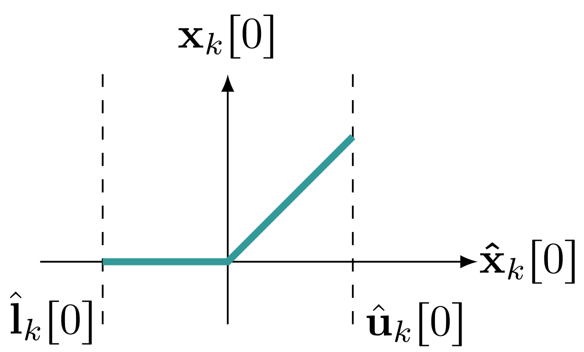
Convex Relaxation: Planet

$$\sigma\left(\hat{\mathbf{x}}_k\right)$$



Convex Relaxation: Planet

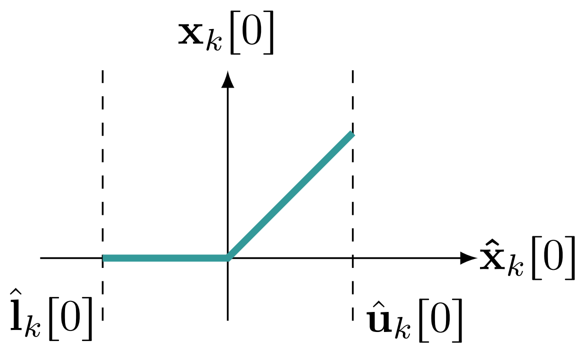
$$\sigma(\hat{\mathbf{x}}_k)$$



$$\longrightarrow$$

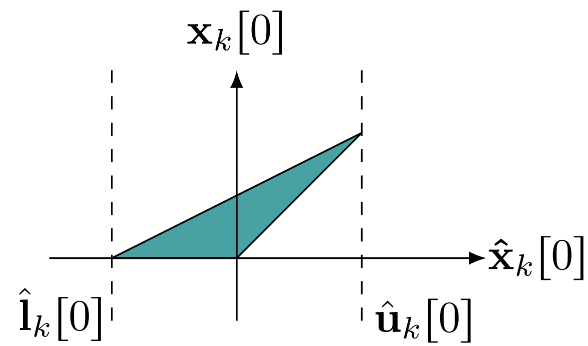
Convex Relaxation: Planet

$$\sigma(\hat{\mathbf{x}}_k)$$



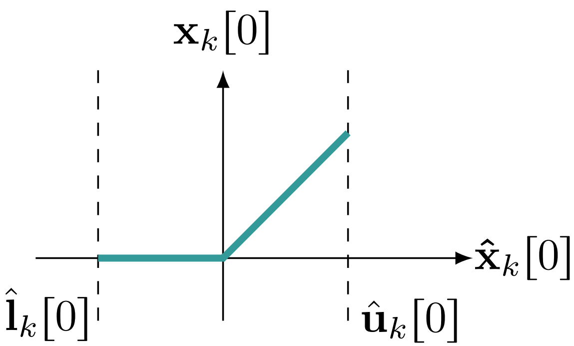
$$\longrightarrow$$

$$\mathrm{Conv}(\sigma(\hat{\mathbf{x}}_k), \hat{\mathbf{l}}_k, \hat{\mathbf{u}}_k)$$



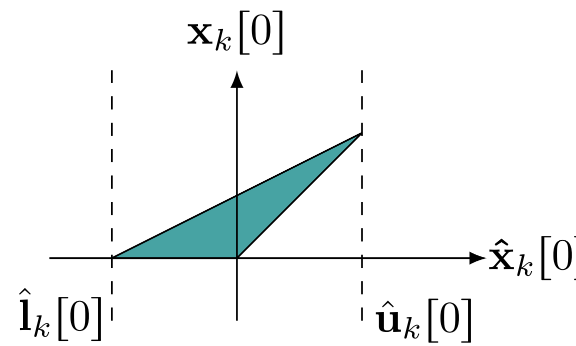
Convex Relaxation: Planet

$$\sigma(\hat{\mathbf{x}}_k)$$



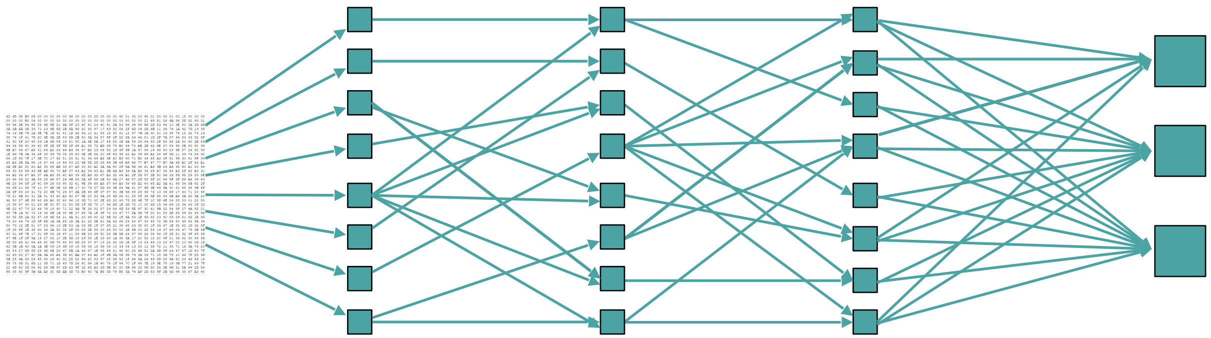
$$\longrightarrow$$

$$\mathrm{Conv}(\sigma(\hat{\mathbf{x}}_k), \hat{\mathbf{l}}_k, \hat{\mathbf{u}}_k)$$



[Ehlers 2017, Wong and Kolter, 2018; Zhang et al., 2018; Dvijotham et al. 2018; Singh et al. 2018; Bunel et al., 2020, Xu et al. 2021, Wang et al. 2021]

Branch and Bound: Branching



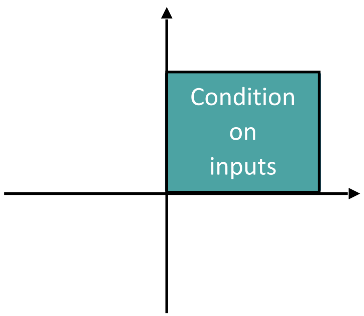


Image Deformations

Branch and Bound: Branching

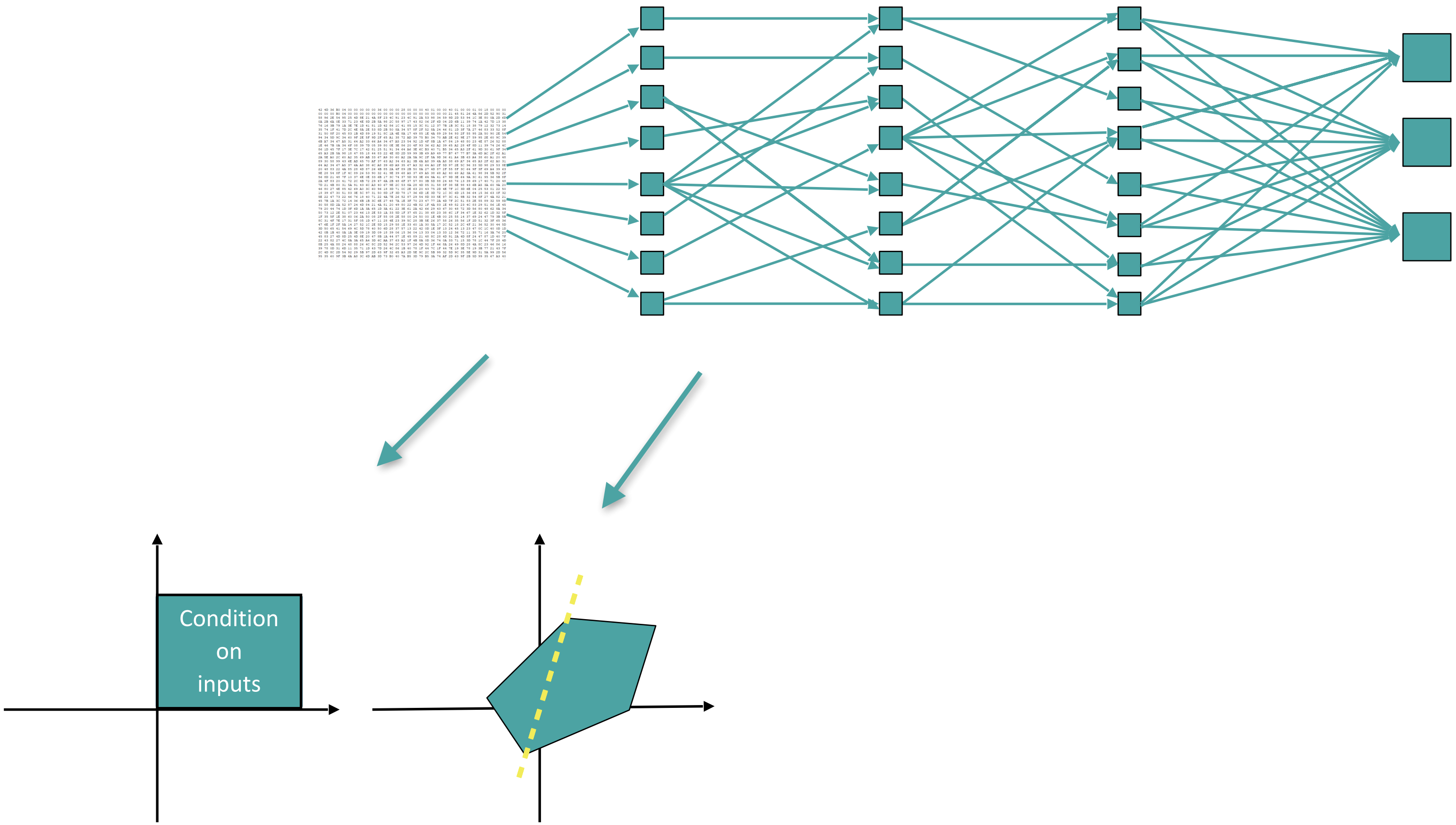


Image Deformations

Branch and Bound: Branching

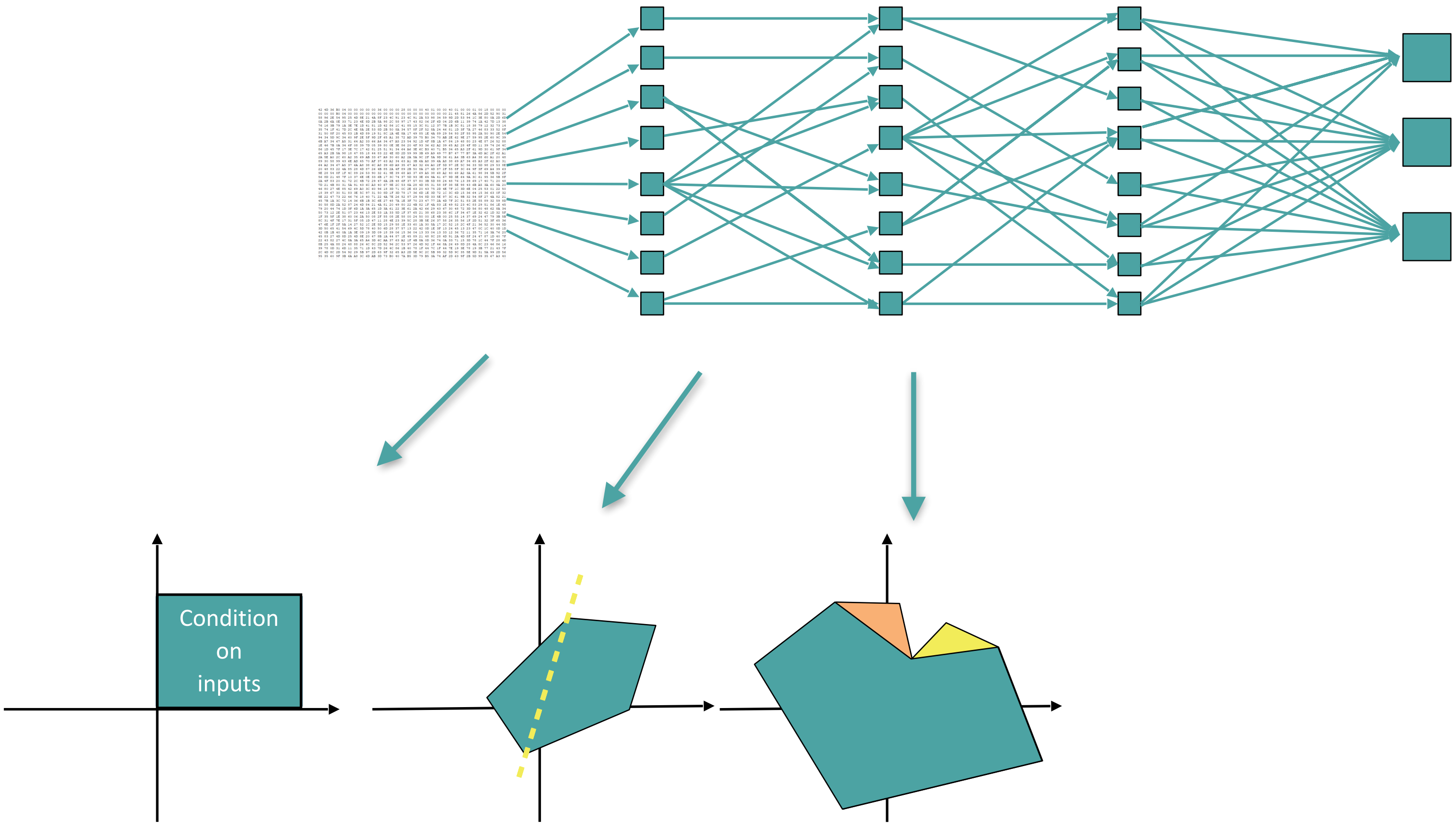


Image Deformations

Branch and Bound: Branching

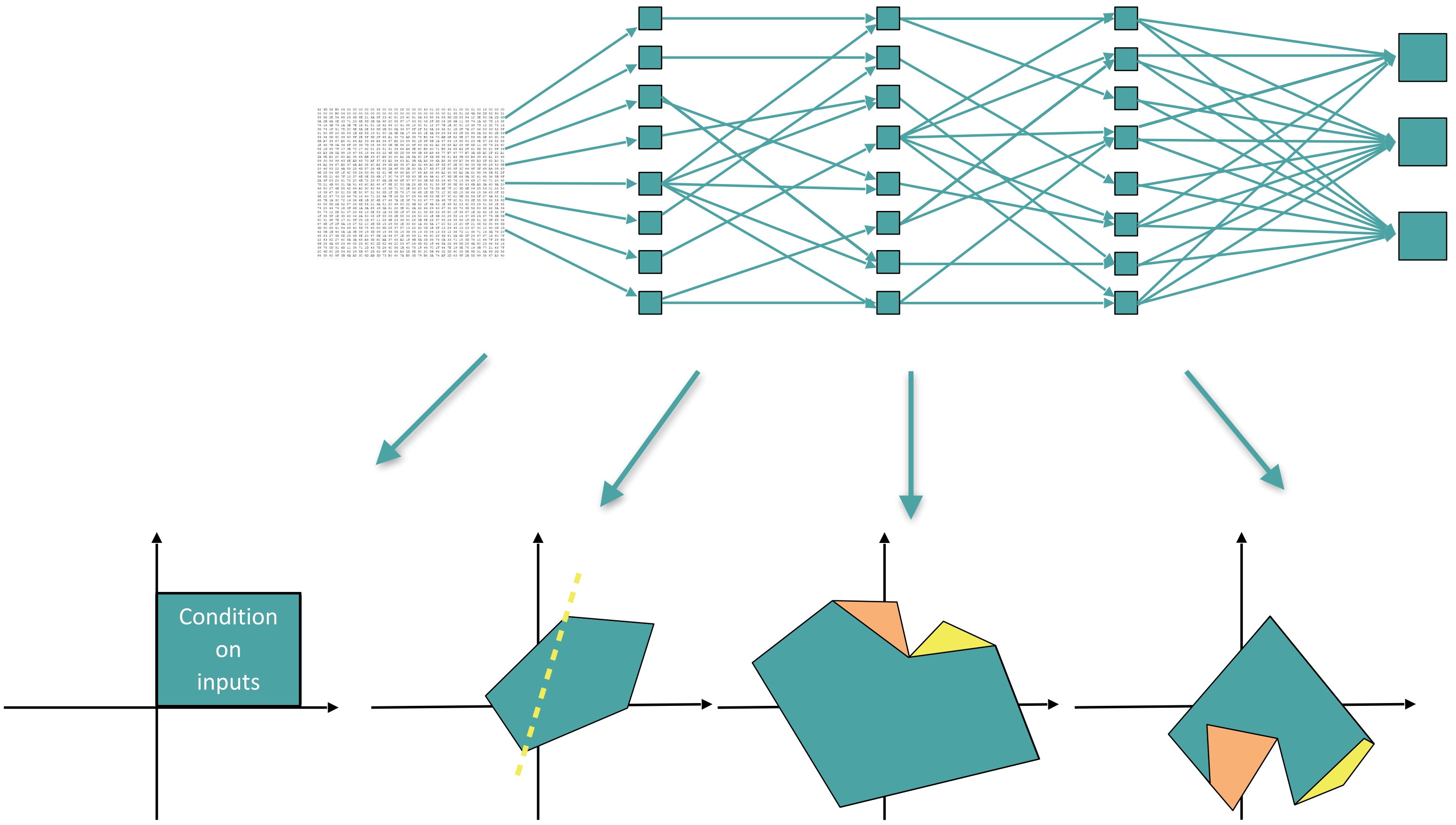


Image Deformations

Branch and Bound: Branching

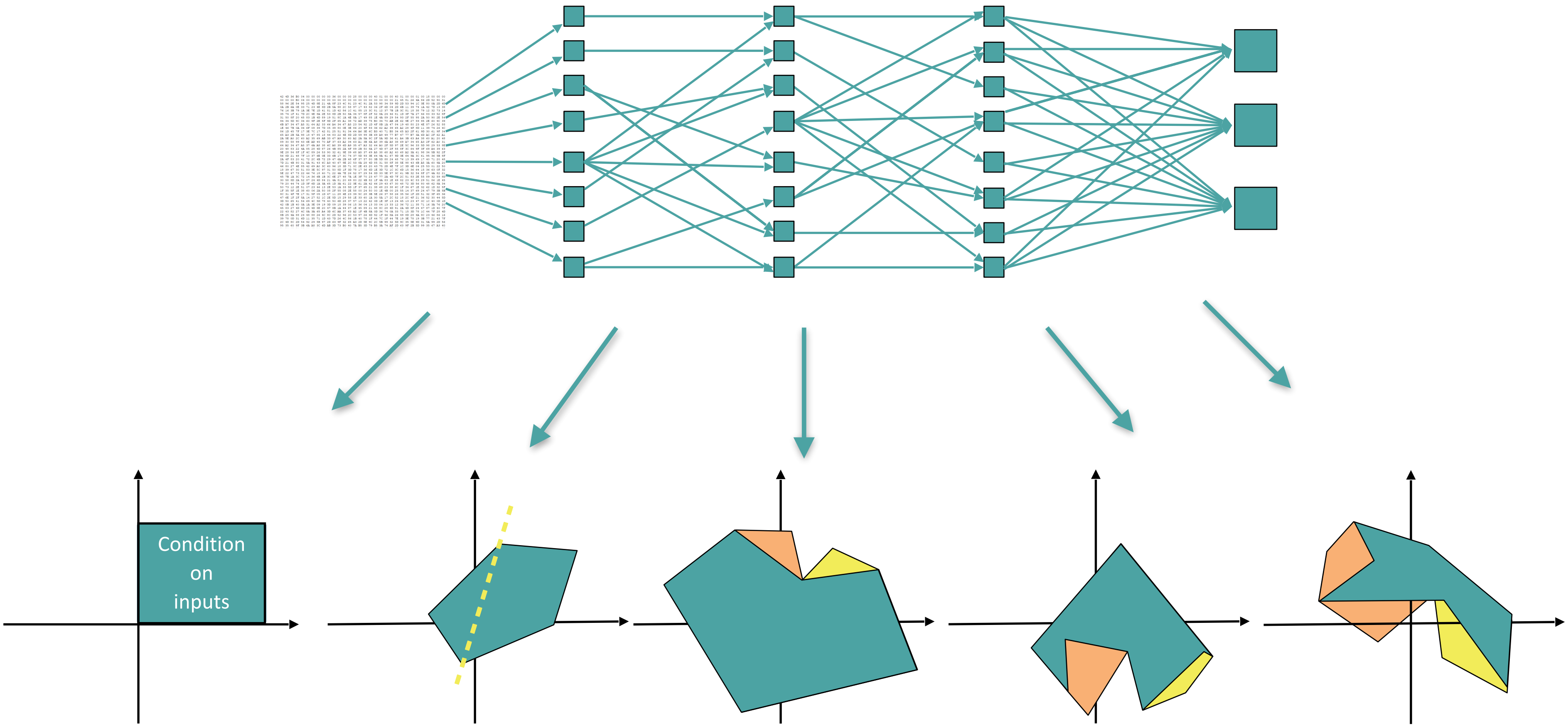


Image Deformations

Branch and Bound: Branching

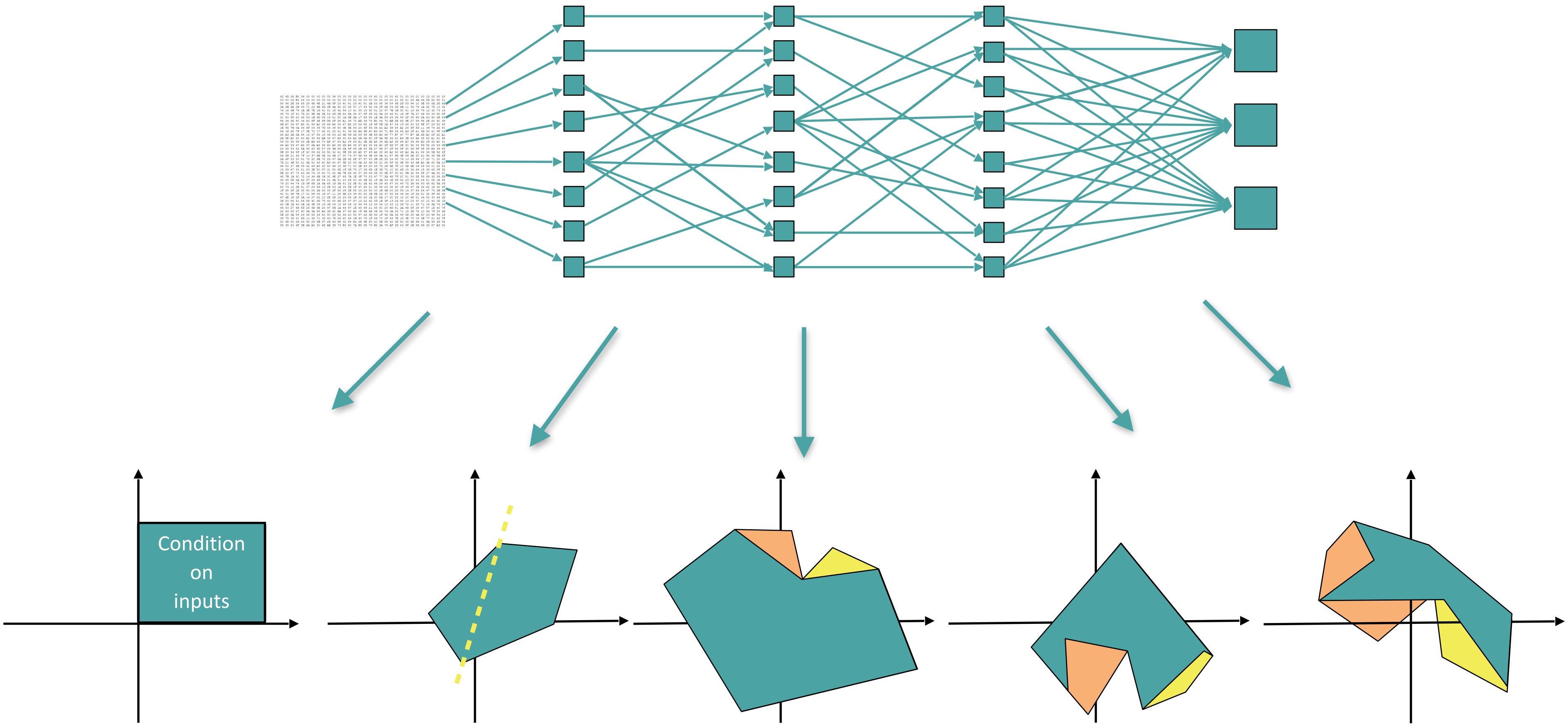
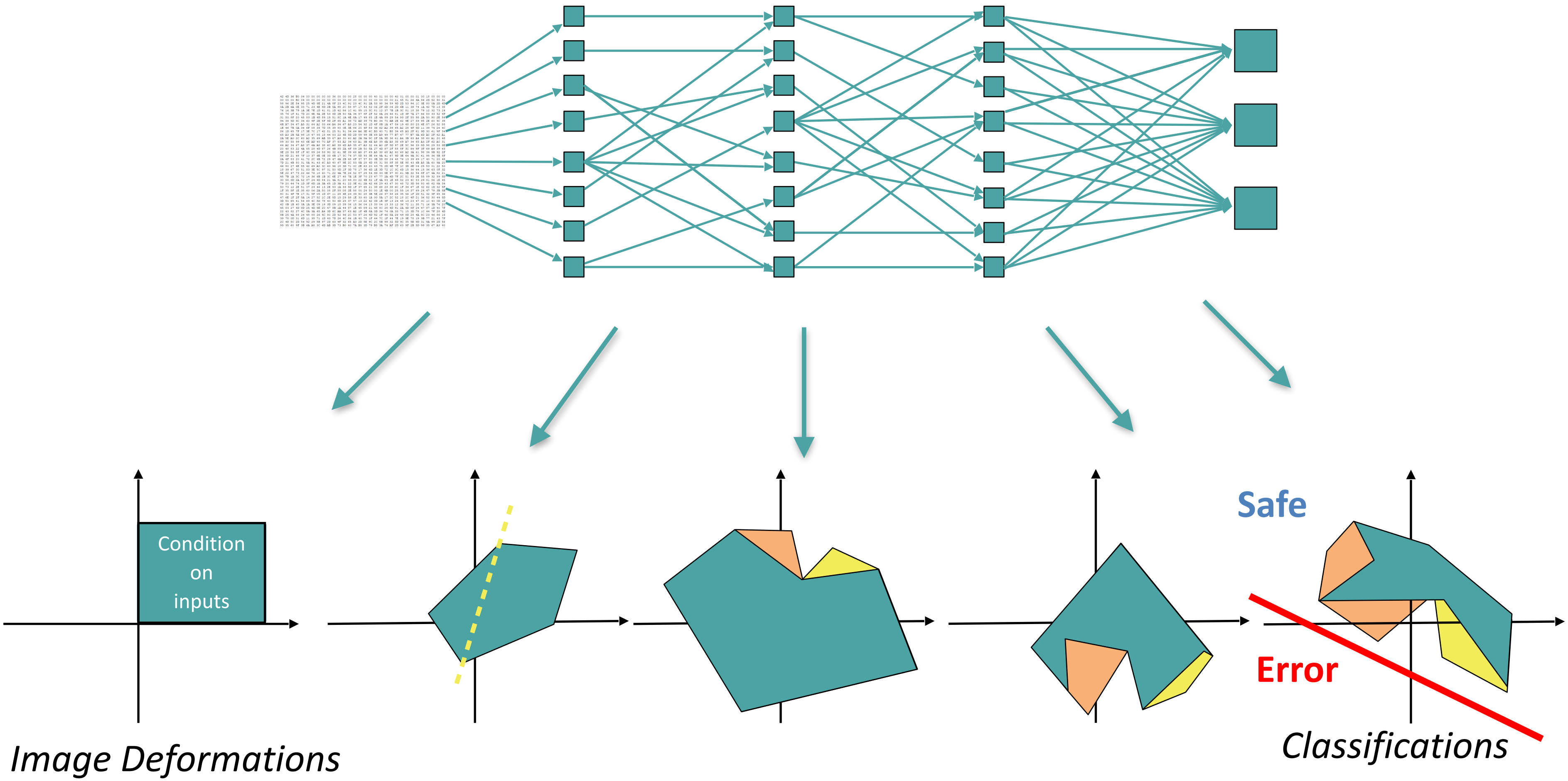


Image Deformations

Verify <u>all</u> properties via iterative branching

Branch and Bound: Branching

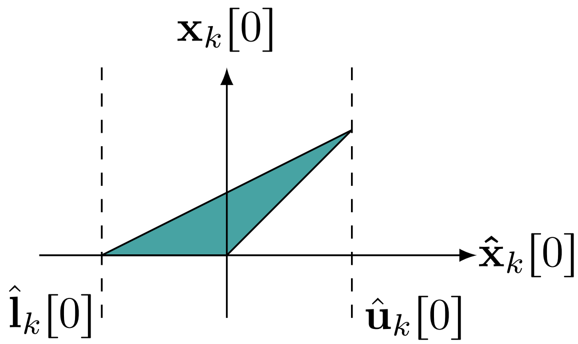


Verify all properties via iterative branching

Property is satisfied

Activation Splitting

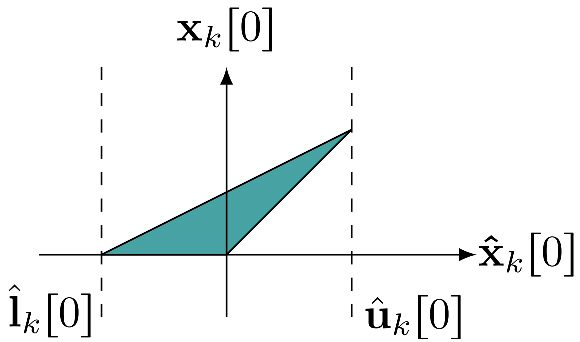
$$\mathrm{Conv}(\sigma(\hat{\mathbf{x}}_k), \hat{\mathbf{l}}_k, \hat{\mathbf{u}}_k)$$



[Bunel et al. 2020, De Palma et al. 2021, Mueller et al. 2022]

Activation Splitting

$$\mathrm{Conv}(\sigma(\hat{\mathbf{x}}_k), \hat{\mathbf{l}}_k, \hat{\mathbf{u}}_k)$$

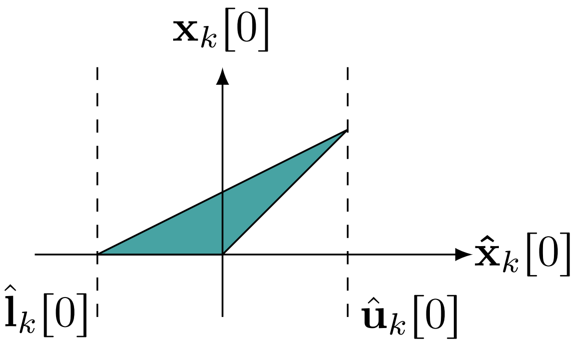


$\longrightarrow$

[Bunel et al. 2020, De Palma et al. 2021, Mueller et al. 2022]

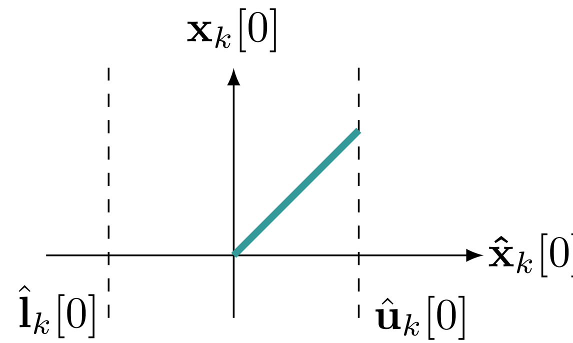
Activation Splitting

$$\mathrm{Conv}(\sigma(\hat{\mathbf{x}}_k), \hat{\mathbf{l}}_k, \hat{\mathbf{u}}_k)$$



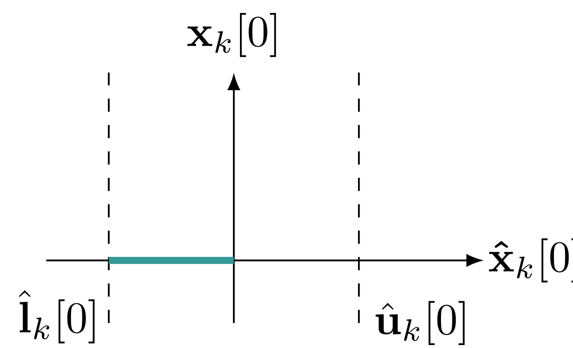
$$\longrightarrow$$

$$\mathrm{Conv}(\sigma(\hat{\mathbf{x}}_k), \mathbf{0}, \hat{\mathbf{u}}_k)$$



$$\bigcup$$

$$\mathrm{Conv}(\sigma(\hat{\mathbf{x}}_k), \hat{\mathbf{l}}_k, \mathbf{0})$$



[Bunel et al. 2020, De Palma et al. 2021, Mueller et al. 2022]

Outline

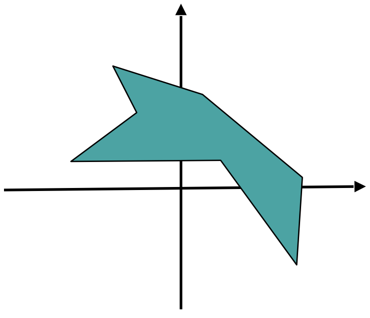
- Neural Network Verification
- **Training for Verified Robustness**
- NLP?
- Discussion

Robust Loss

$$\min_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \left[\max_{\mathbf{x}'\in\mathcal{C}(\mathbf{x})} \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}'), \mathbf{y}) \right]$$

Robust Loss

$$\min_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \left[\underbrace{\max_{\mathbf{x}'\in\mathcal{C}(\mathbf{x})} \mathcal{L}(f(\boldsymbol{\theta},\mathbf{x}'),\mathbf{y})}_{\mathcal{L}^*(f(\boldsymbol{\theta},\mathbf{x}),y)} \right]$$

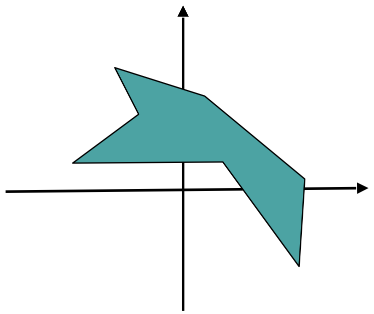


Adversarial Training

$$\text{Lower bound} \rightarrow \text{adversarial training}$$

[Madry et al. 2018]

$$\mathcal{L}^*(f(\boldsymbol{\theta}, \mathbf{x}), y)$$

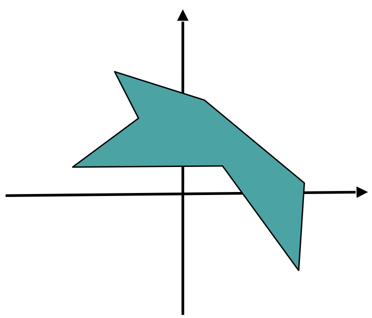


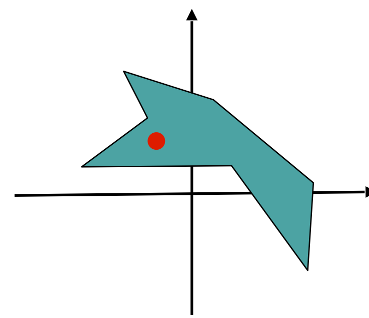
Adversarial Training

Lower bound $\rightarrow$ adversarial training

[Madry et al. 2018]

$$\mathcal{L}^*(f(\boldsymbol{\theta}, \mathbf{x}), y) \quad \geq \quad \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}_{\text{adv}}), y)$$



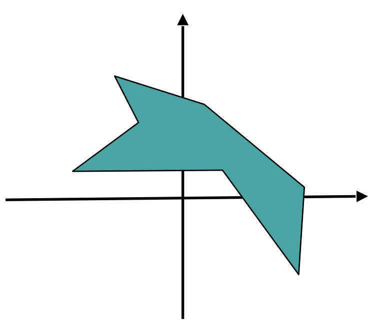


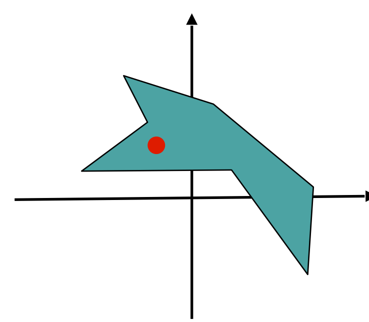
Adversarial Training

Lower bound $\rightarrow$ adversarial training

[Madry et al. 2018]

$$\mathcal{L}^*(f(\boldsymbol{\theta}, \mathbf{x}), y) \geq \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}_{\mathrm{adv}}), y)$$





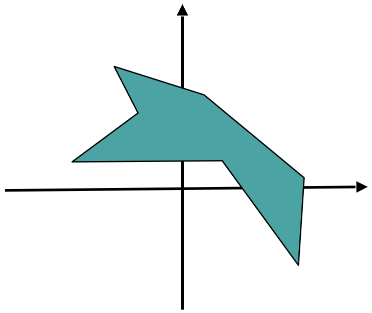
formal guarantees?

Verified Training

$\text{Upper bound} \rightarrow \text{certified training}$

[Wong and Kolter 2018, Gowal et al. 2018, Zhang et al. 2020, Shi et al. 2021]

$$\mathcal{L}^*(f(\boldsymbol{\theta}, \mathbf{x}), y)$$

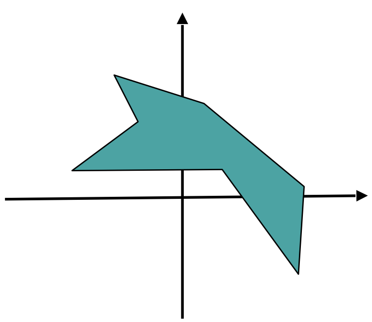


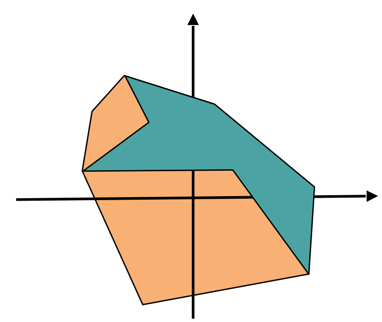
Verified Training

Upper bound $\rightarrow$ certified training

[Wong and Kolter 2018, Gowal et al. 2018, Zhang et al. 2020, Shi et al. 2021]

$$\mathcal{L}^*(f(\boldsymbol{\theta}, \mathbf{x}), y) \quad \leq \quad \mathcal{L}_{\text{ver}}(f(\boldsymbol{\theta}, \mathbf{x}), y)$$



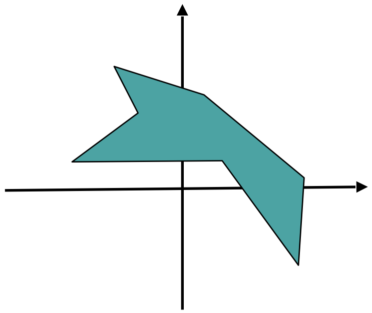


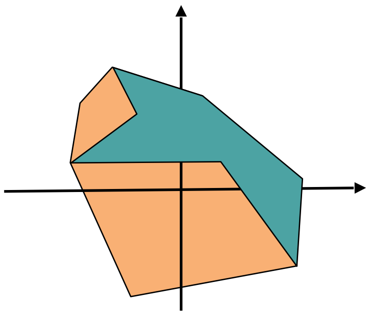
Verified Training

Upper bound $\rightarrow$ certified training

[Wong and Kolter 2018, Gowal et al. 2018, Zhang et al. 2020, Shi et al. 2021]

$$\mathcal{L}^*(f(\boldsymbol{\theta}, \mathbf{x}), y) \quad \leq \quad \mathcal{L}_{\text{ver}}(f(\boldsymbol{\theta}, \mathbf{x}), y)$$

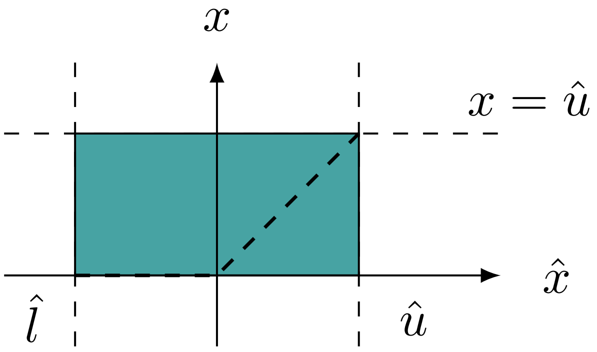




verification via cheap incomplete verifiers

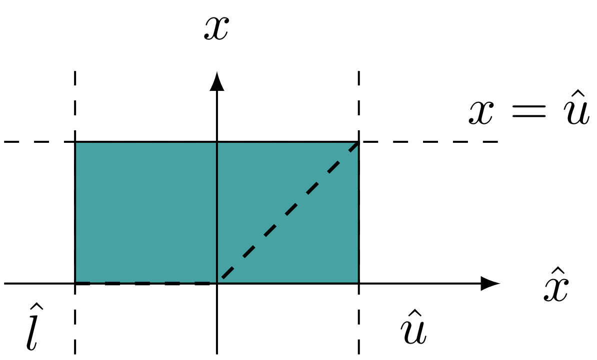
Verified Training

IBP

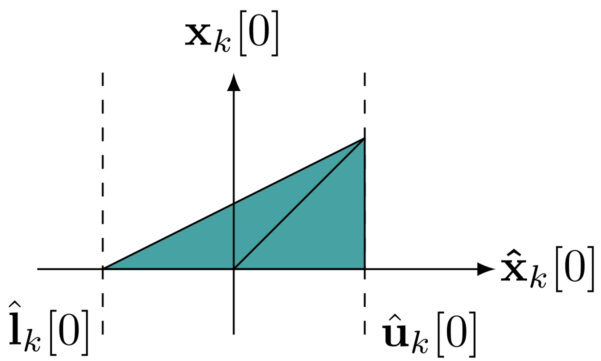


Verified Training

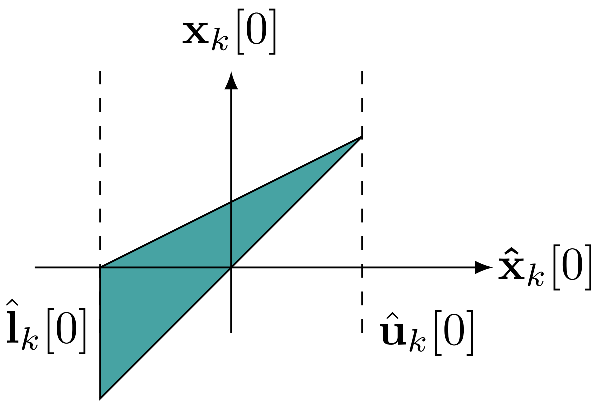
IBP



CROWN

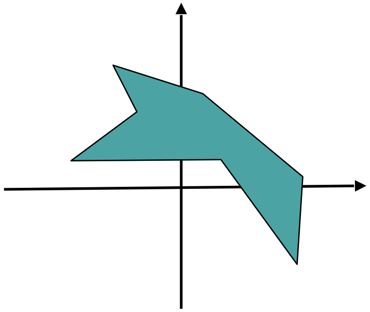


OR



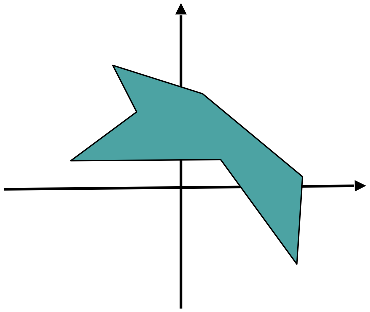
Loss Expressivity

$$\mathcal{L}^*(f(\boldsymbol{\theta}, \mathbf{x}), y)$$



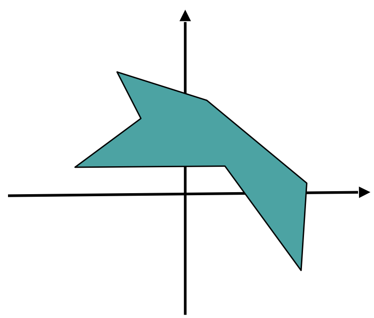
Loss Expressivity

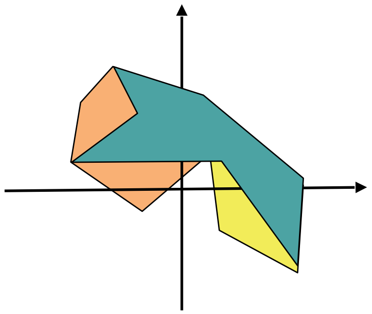
$$\mathcal{L}^*(f(\boldsymbol{\theta}, \mathbf{x}), y) \qquad \leq$$



Loss Expressivity

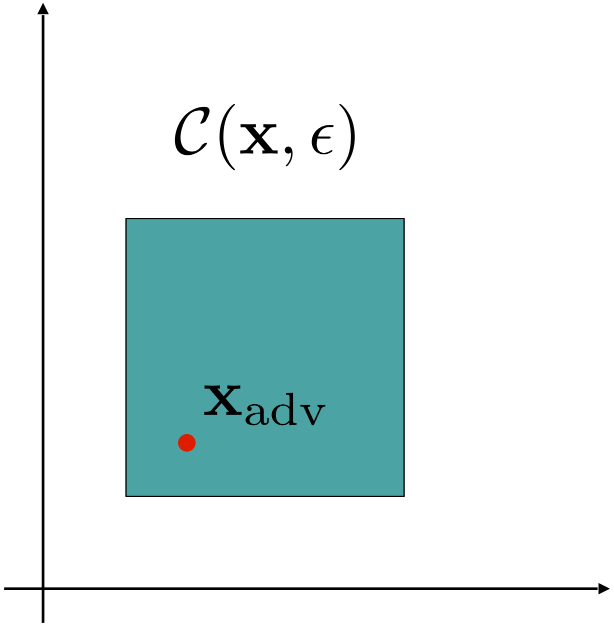
$$\mathcal{L}^*(f(\boldsymbol{\theta}, \mathbf{x}), y) \qquad \leq \qquad ?$$





Hybrid Training Methods: SABR

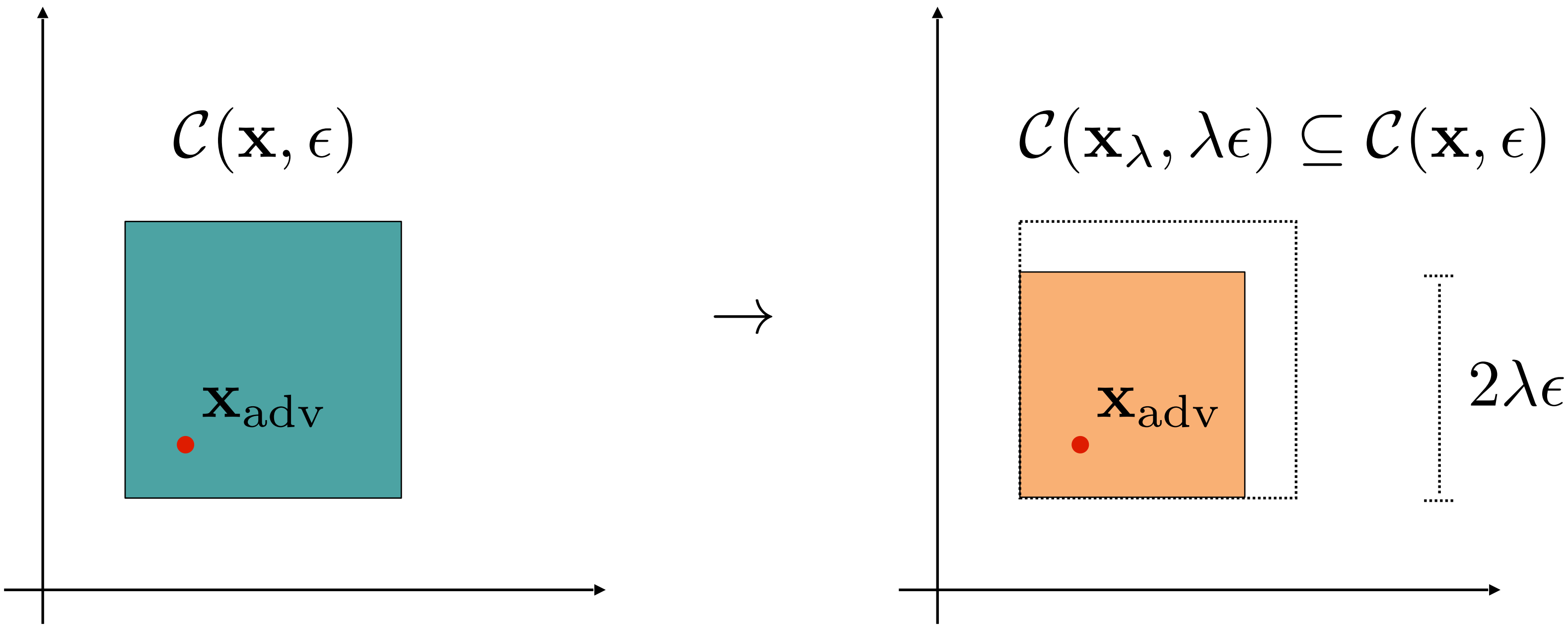
Compute over-approximation over a parametrized subset of the input domain that includes an adversarial attack.



[Mueller et al., 2023]

Hybrid Training Methods: SABR

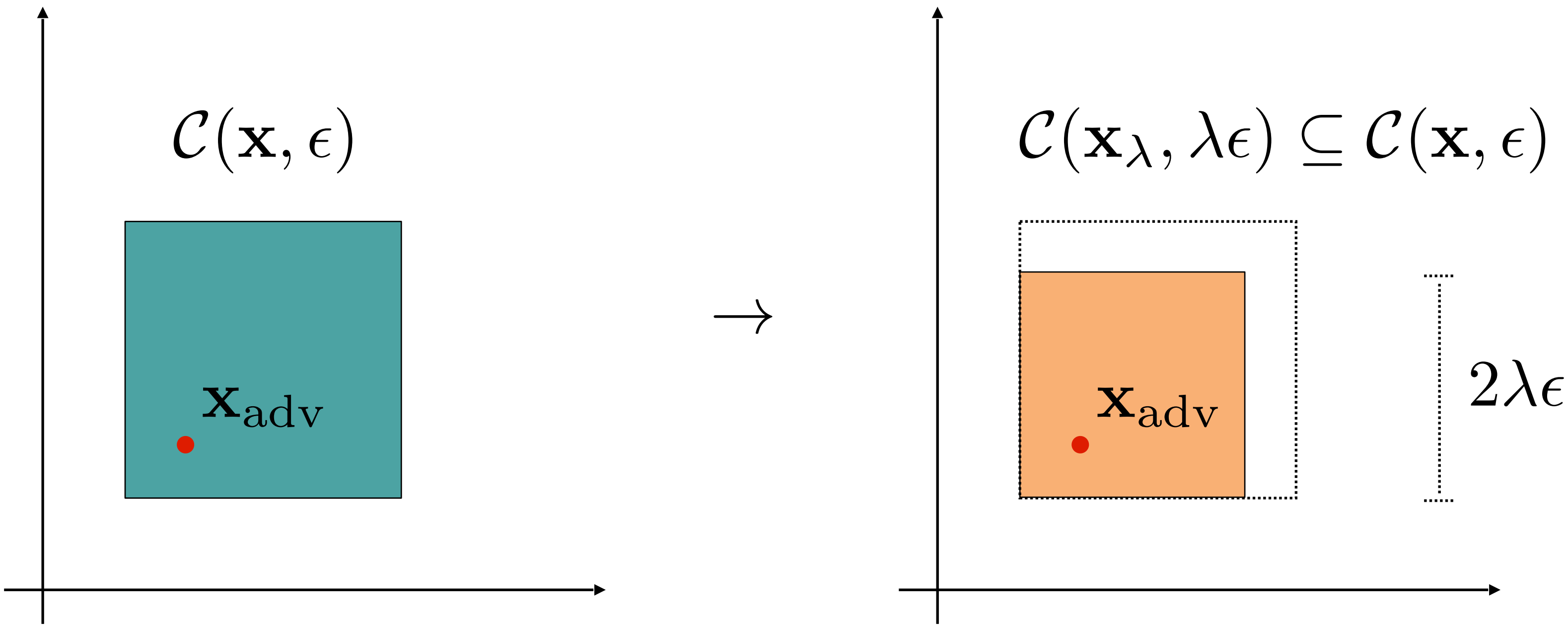
Compute over-approximation over a parametrized subset of the input domain that includes an adversarial attack.



[Mueller et al., 2023]

Hybrid Training Methods: SABR

Compute over-approximation over a parametrized subset of the input domain that includes an adversarial attack.



verification via BaB

[Mueller et al., 2023]

Expressive Losses for Verified Robustness via Convex Combinations

Alessandro De Palma
Imperial College London[1]
adepalma@ic.ac.uk

Rudy Bunel
Google DeepMind

Krishnamurthy (Dj) Dvijotham
Google DeepMind

M. Pawan Kumar
Google DeepMind

Robert Stanforth
Google DeepMind

Alessio Lomuscio
Imperial College London

https://arxiv.org/abs/2305.13991

Loss Expressivity

A parametrized family of losses $\mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y)$ is *expressive* if:

Loss Expressivity

A parametrized family of losses $\mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y)$ is *expressive* if:

- $\mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}_{\text{adv}}), y) \leq \mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y) \leq \mathcal{L}_{\text{ver}}(f(\boldsymbol{\theta}, \mathbf{x}), y) \ \forall \ \alpha \in [0, 1]$;

Loss Expressivity

A parametrized family of losses $\mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y)$ is *expressive* if:

- $\mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}_{\text{adv}}), y) \leq \mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y) \leq \mathcal{L}_{\text{ver}}(f(\boldsymbol{\theta}, \mathbf{x}), y) \; \forall \; \alpha \in [0, 1]$;
- $\mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y)$ is monotonically increasing with α;

Loss Expressivity

A parametrized family of losses $\mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y)$ is *expressive* if:

- $\mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}_{\text{adv}}), y) \leq \mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y) \leq \mathcal{L}_{\text{ver}}(f(\boldsymbol{\theta}, \mathbf{x}), y) \ \forall \ \alpha \in [0, 1]$;
- $\mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y)$ is monotonically increasing with α;
- $\mathcal{L}_0(\boldsymbol{\theta}, \mathbf{x}, y) = \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}_{\text{adv}}), y)$;

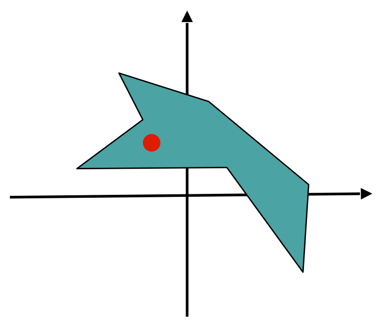
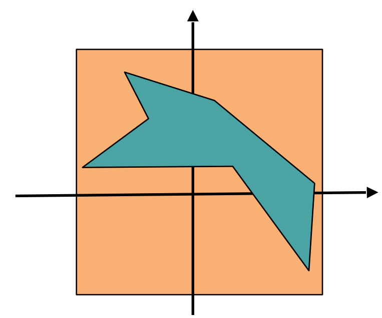
Loss Expressivity

A parametrized family of losses $\mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y)$ is *expressive* if:

- $\mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}_{\text{adv}}), y) \leq \mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y) \leq \mathcal{L}_{\text{ver}}(f(\boldsymbol{\theta}, \mathbf{x}), y) \ \forall \ \alpha \in [0, 1]$;
- $\mathcal{L}_\alpha(\boldsymbol{\theta}, \mathbf{x}, y)$ is monotonically increasing with α;
- $\mathcal{L}_0(\boldsymbol{\theta}, \mathbf{x}, y) = \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}_{\text{adv}}), y)$;
- $\mathcal{L}_1(\boldsymbol{\theta}, \mathbf{x}, y) = \mathcal{L}_{\text{ver}}(f(\boldsymbol{\theta}, \mathbf{x}), y)$.

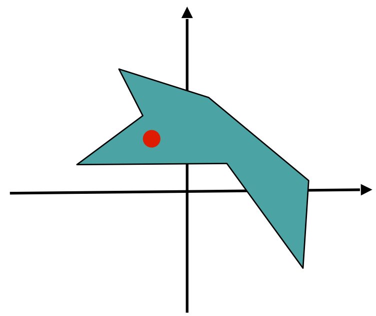
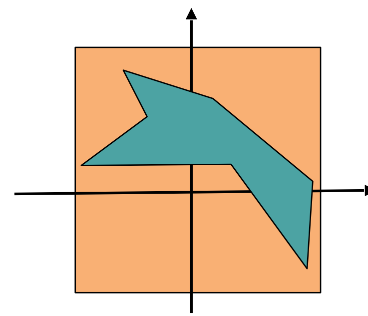
Expressivity via Convex Combinations

CC-IBP

$\mathcal{L}(-[(1-\alpha)$  $+\alpha$  $], y)$

Expressivity via Convex Combinations

CC-IBP

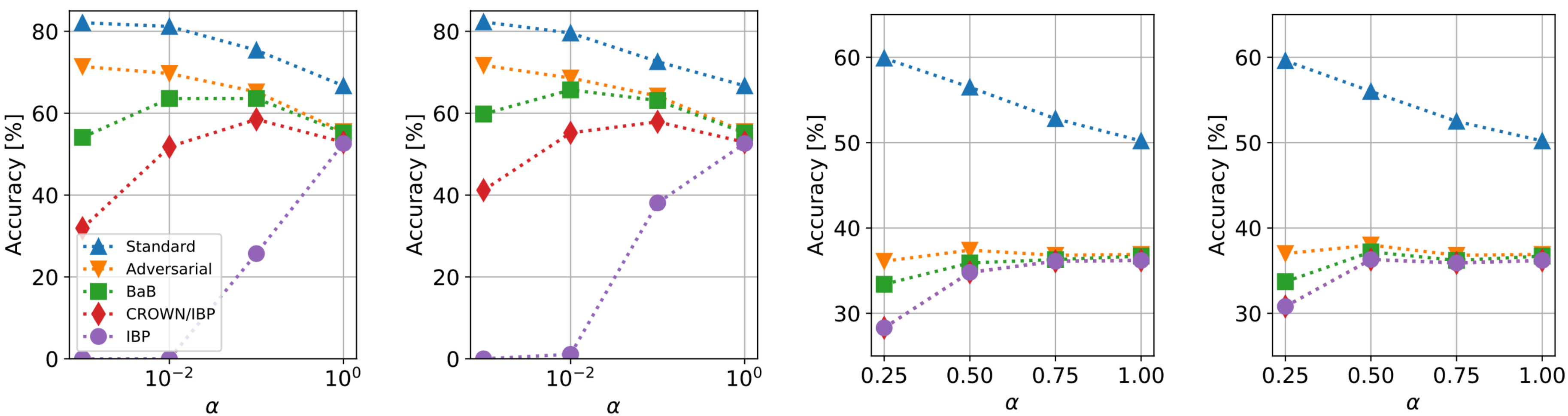
$\mathcal{L}(-\,[(1-\alpha)$  $+\,\alpha$  $],\, y)$

MTL-IBP

$$(1-\alpha)\mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}_{\text{adv}}), y) + \alpha\ \mathcal{L}_{\text{ver}}(f(\boldsymbol{\theta}, \mathbf{x}), y)$$

Loss Sensitivity

Sensitivity of CC-IBP and MTL-IBP to the convex combination coefficient α on the first 1000 CIFAR-10 test images.



(a) CC-IBP, $\epsilon = {}^{2}/_{255}$. (b) MTL-IBP, $\epsilon = {}^{2}/_{255}$. (c) CC-IBP, $\epsilon = {}^{8}/_{255}$. (d) MTL-IBP, $\epsilon = {}^{8}/_{255}$.

Experimental Results

Performance of different verified training algorithms under ℓ_∞ norm perturbations on the CIFAR-10 dataset.

Dataset	ϵ	Method	Standard acc. [%]	Verified rob. acc. [%]	Training time [s]
CIFAR-10	$\frac{2}{255}$	CC-IBP	<u>80.09</u>	**63.78**	1.77×10^4
		MTL-IBP	**80.11**	<u>63.24</u>	1.76×10^4
		STAPS	79.76	62.98	1.41×10^5
		SABR	79.24	62.84	2.56×10^4
		SortNet	67.72	56.94	4.04×10^4
		IBP-R	78.19	61.97	9.34×10^3
		CROWN-IBP	71.52	53.97	9.13×10^4
	$\frac{8}{255}$	CC-IBP	<u>53.71</u>	<u>35.27</u>	1.72×10^4
		MTL-IBP	<u>53.35</u>	<u>35.44</u>	1.70×10^4
		STAPS	52.82	34.65	2.70×10^4
		SABR	52.38	35.13	2.64×10^4
		SortNet	**54.84**	**40.39**	4.04×10^4
		IBP-R	52.74	27.55	5.89×10^3
		IBP	48.94	34.97	9.51×10^3

Experimental Results

Performance of different verified training algorithms under ℓ_∞ norm perturbations on the TinyImageNet and downscaled (64×64) ImageNet datasets.

Dataset	ϵ	Method	Standard acc. [%]	Verified rob. acc. [%]	Training time [s]
TinyImageNet	$\frac{1}{255}$	CC-IBP	<u>32.71</u>	<u>23.10</u>	6.58×10^4
		MTL-IBP	**32.76**	**24.14**	6.56×10^4
		STAPS	28.98	22.16	3.06×10^5
		SABR	28.85	20.46	2.07×10^5
		SortNet	25.69	18.18	1.56×10^5
		IBP	25.92	17.87	3.53×10^4
ImageNet64	$\frac{1}{255}$	CC-IBP	<u>19.62</u>	<u>11.87</u>	3.26×10^5
		MTL-IBP	**20.15**	**12.13**	3.52×10^5
		SortNet	14.79	9.54	6.58×10^5
		CROWN-IBP	16.23	8.73	/

Future Work

- Theoretical understanding of relative method performance;

Future Work

- Theoretical understanding of relative method performance;
- Would more network capacity help?

Future Work

- Theoretical understanding of relative method performance;
- Would more network capacity help?
- Examine applications to different data domains.

Outline

- Neural Network Verification
- Training for Verified Robustness
- **NLP?**
- Discussion

NLP Challenges

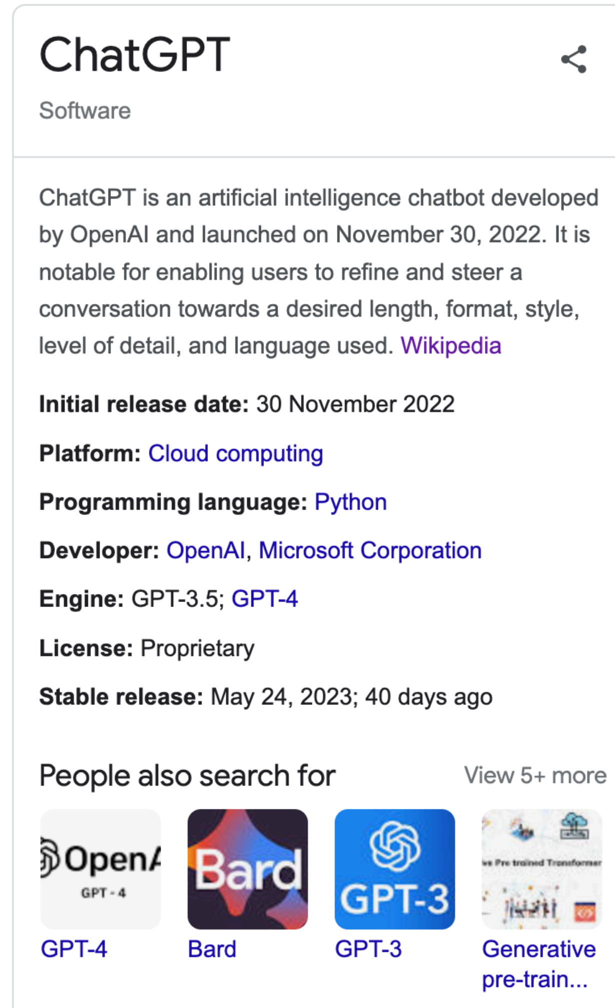
- Threat model: robustness to synonym-based perturbations;

NLP Challenges

- Threat model: robustness to synonym-based perturbations;
- More complexity than PLNN;

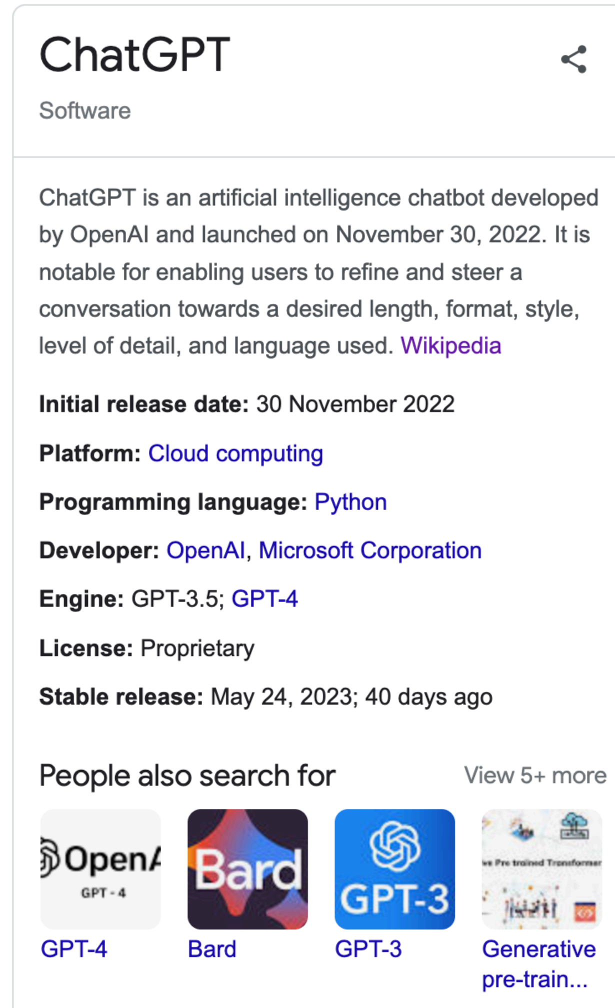
NLP Challenges

- Threat model: robustness to synonym-based perturbations;
- More complexity than PLNN;
- LLMs: very large, pre-trained;



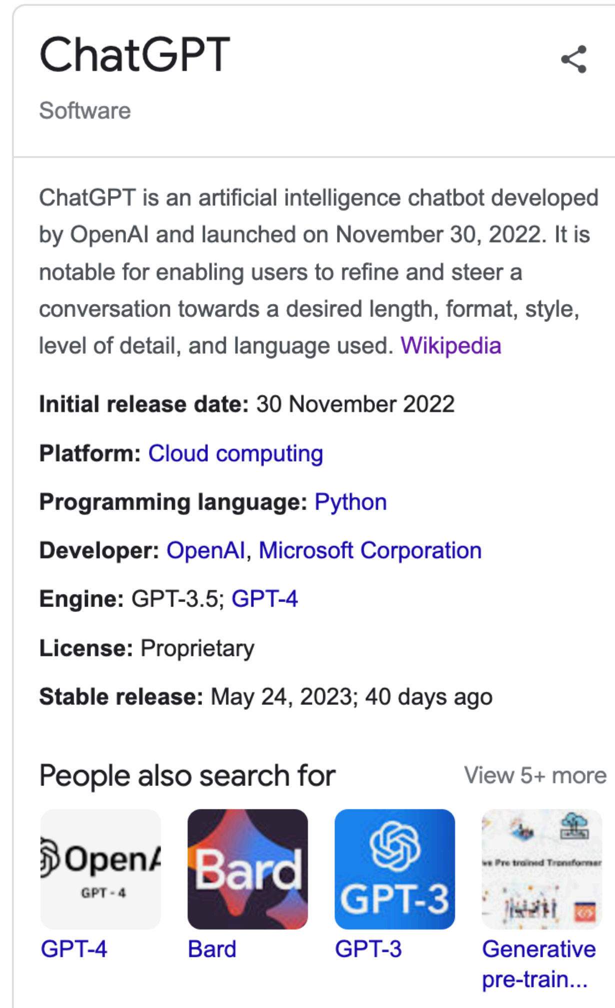
NLP Challenges

- Threat model: robustness to synonym-based perturbations;
- More complexity than PLNN;
- LLMs: very large, pre-trained;
- Interaction: black-box, prompt-based (zero) few-shot learning;



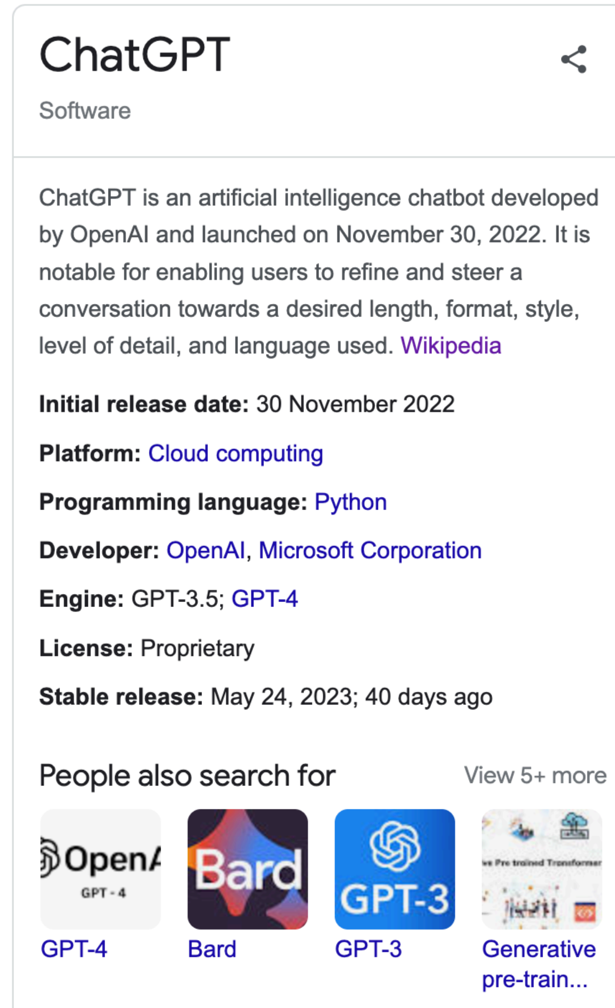
NLP Challenges

- Threat model: robustness to synonym-based perturbations;
- More complexity than PLNN;
- LLMs: very large, pre-trained;
- Interaction: black-box, prompt-based (zero) few-shot learning;
- Hallucinations?



NLP Challenges

- Threat model: robustness to synonym-based perturbations;
- More complexity than PLNN;
- LLMs: very large, pre-trained;
- Interaction: black-box, prompt-based (zero) few-shot learning;
- Hallucinations?
- Textual inputs are discrete.



Discussion