Verifying Global Two-Safety Properties in Neural Networks with Confidence

Anagha Athavale
anagha.athavale@tuwien.ac.at

Ezio Bartocci
ezio.bartocci@tuwien.ac.at

Maria Christakis
maria.christakis@tuwien.ac.at

Matteo Maffei
matteo.maffei@tuwien.ac.at

Dejan Nickovic
dejan.nickovic@ait.ac.at

Georg Weissenbacher
georg.wesissenbacher@tuwien.ac.at

36th International Conference on Computer Aided Verification
27 July 2024

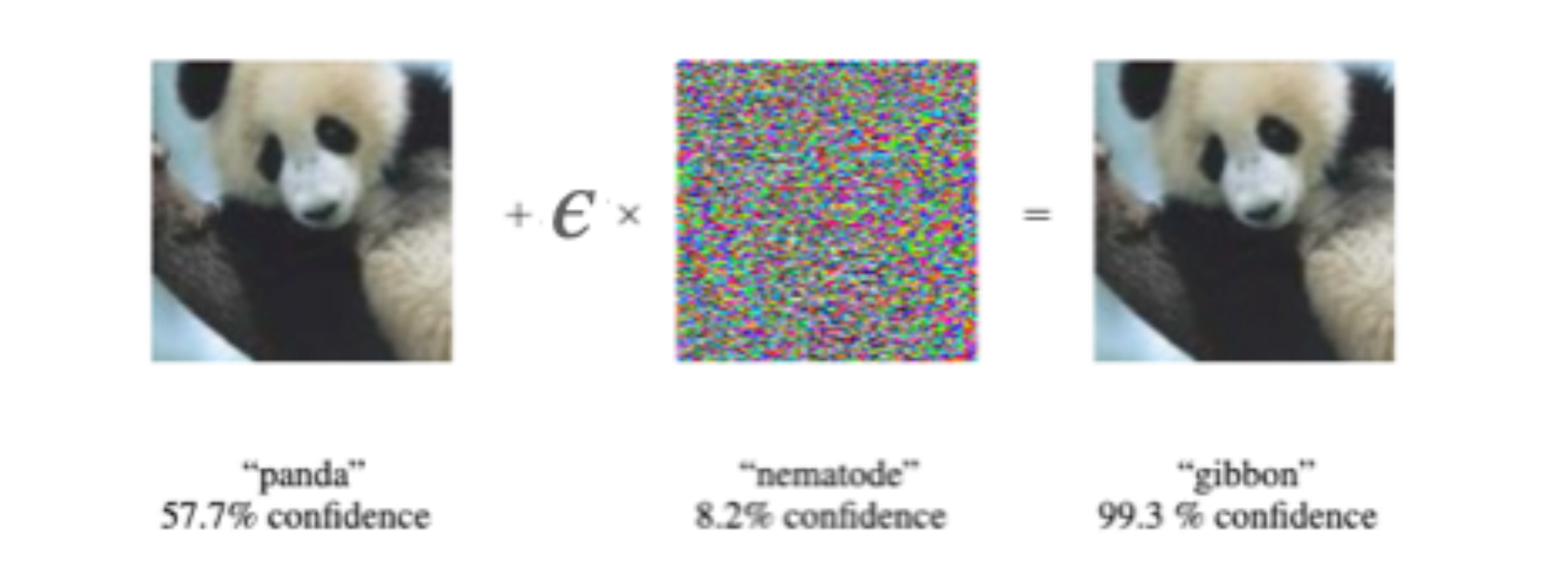
Why is verification of NNs important?

- Safety critical applications
  - Safety is a concern
  - Lives /money/ personal information at risk
- Applications such as:
  - Medical diagnosis
  - Self driving vehicles
  - Financial systems

Cannot trust ML-based systems where cost of error to be paid is huge!

Why is verification of NNs important?

Source: Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples.



- Szegedy et al. (2013), Goodfellow et al. (2014) observe a curious phenomenon
- By adding ϵ (an imperceptibly small vector) to the input vector, classification changes with a high confidence!

Why is verification of NNs important?

- Safety critical applications
  - Safety is a concern
  - Lives /money/ personal information at risk
- Applications such as:
  - Medical diagnosis
  - Self driving vehicles
  - Financial systems

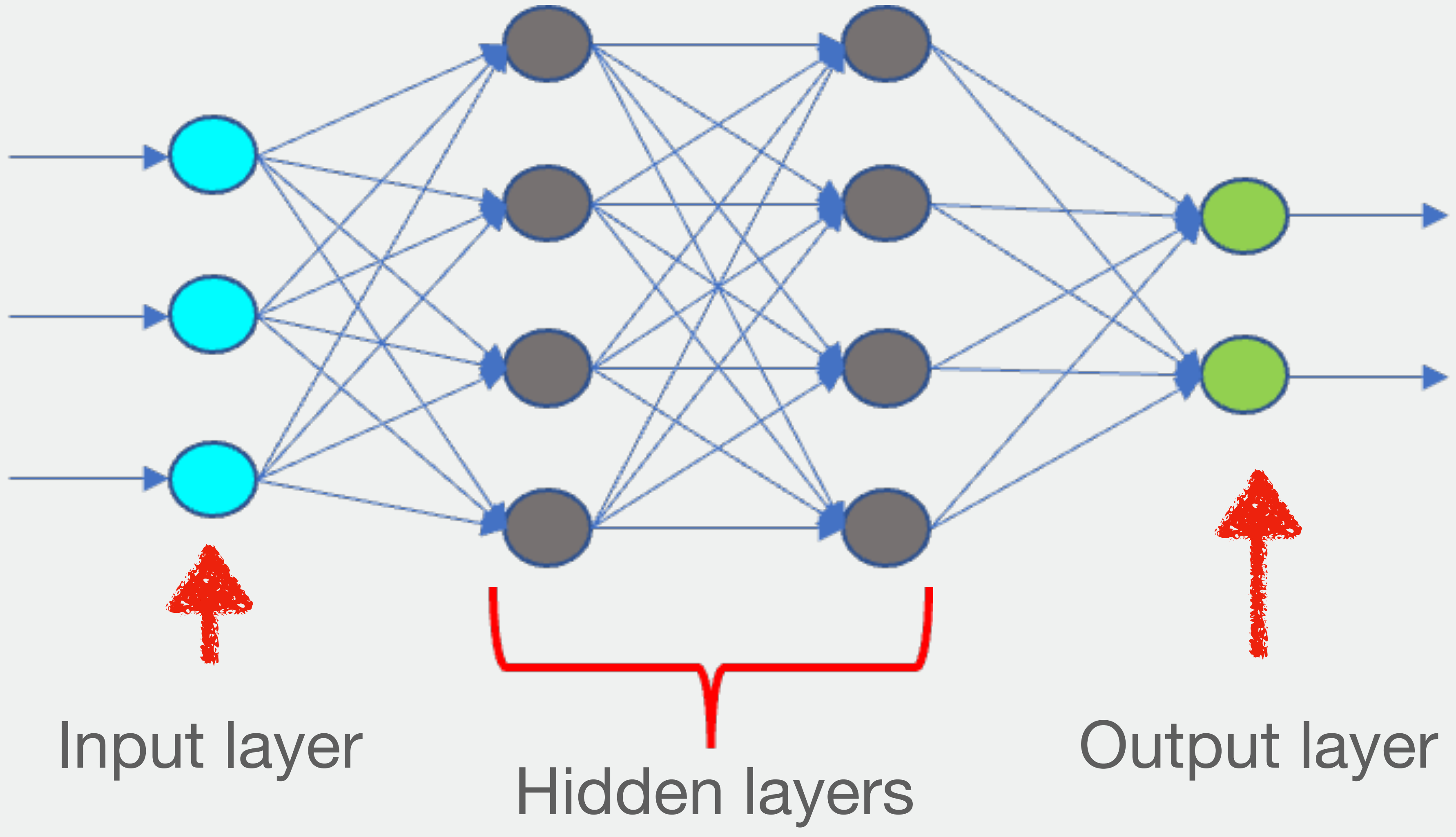
Cannot trust
ML-based
systems where cost of error
to be paid is huge!

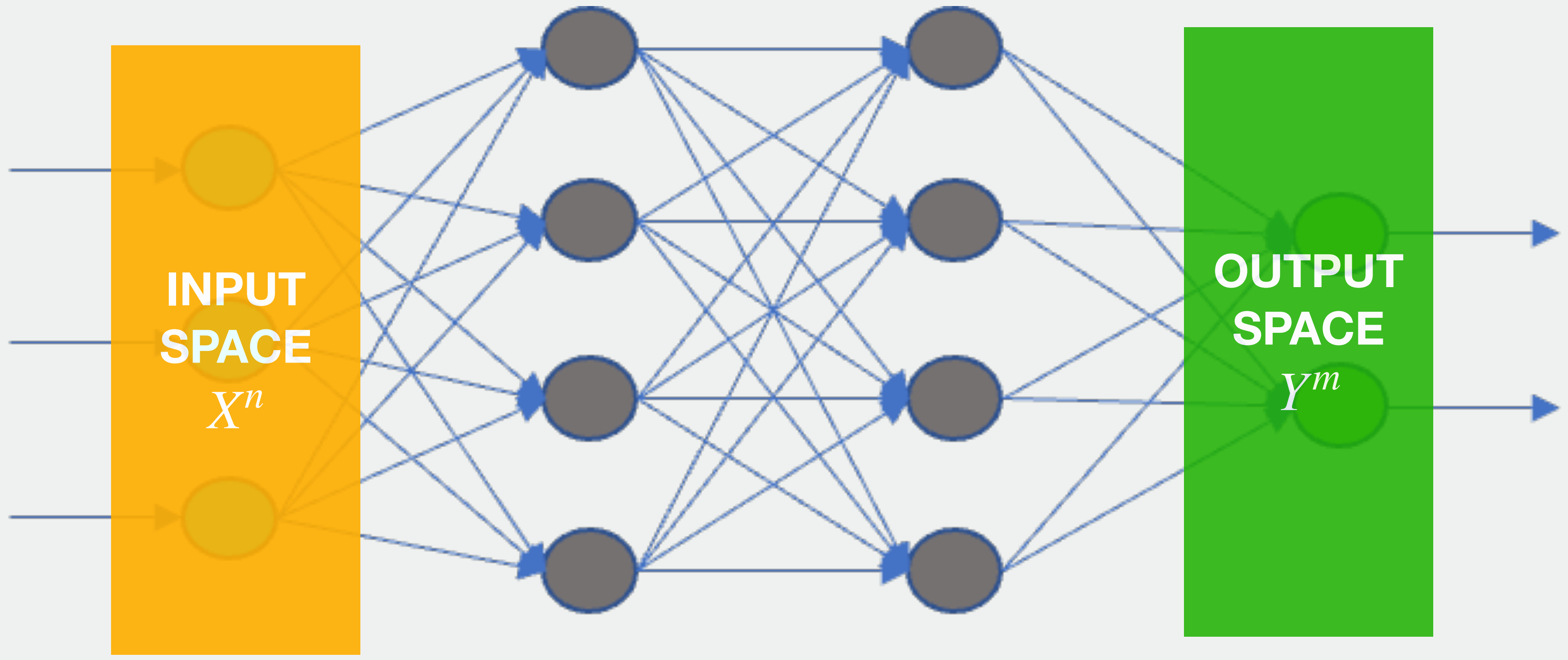
Solution -
Verification techniques with
formal guarantees

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

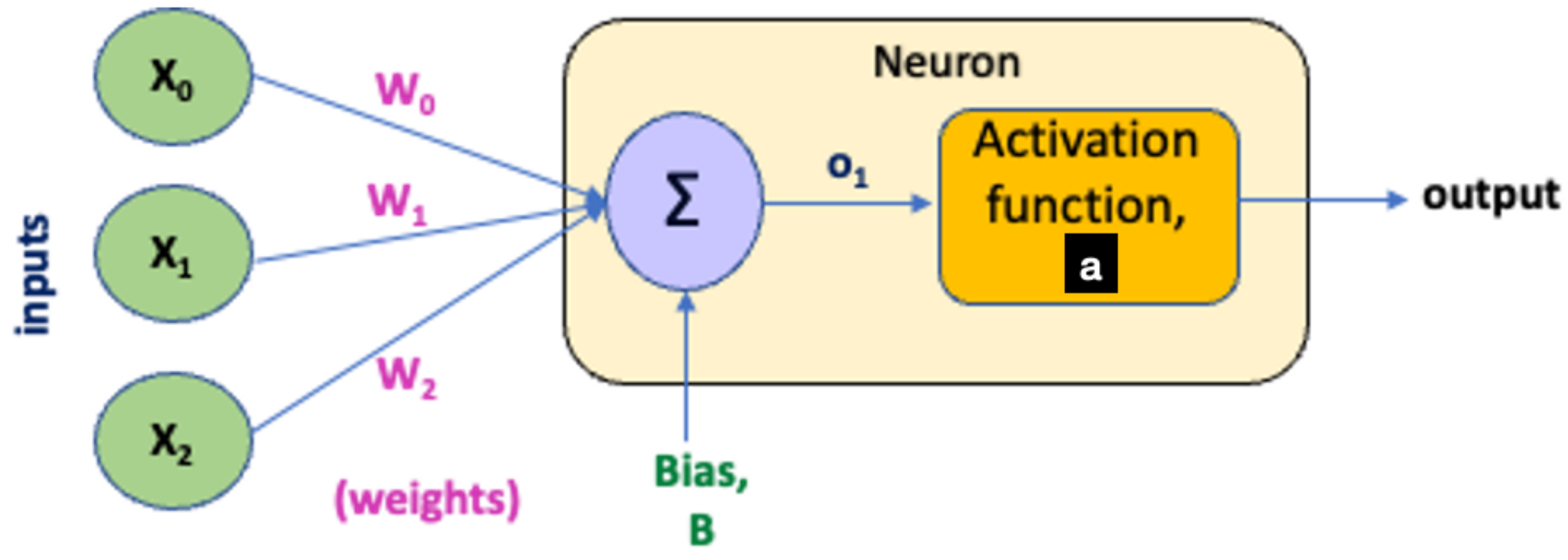
Feed-Forward Neural Networks

- Neural network $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ modeled as a DAG, $G = (V, E)$
  - V : finite set of nodes
  - $E \subseteq V \times V$: finite set of edges
- Nodes V partitioned into l layers V^i with $1 \leq i \leq l$
  - V^1: input layer
  - $V^2, \ldots, V^{l-1}$: hidden layers
  - V^l: output layer





Feed-Forward Neural Networks



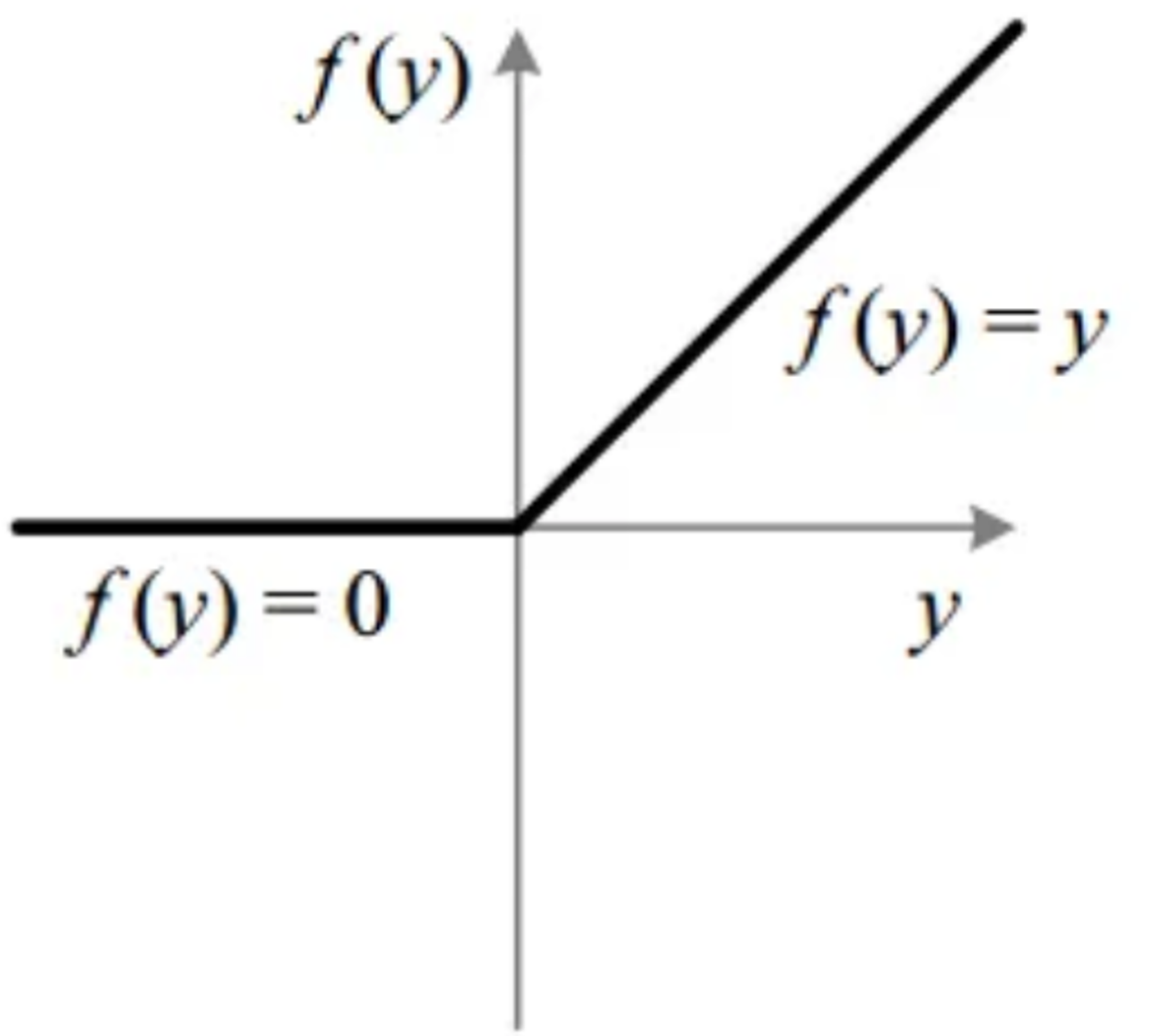
$$output = a(\sum_i W_i * x_i + B)$$

Activation Functions

Activation functions help introduce non-linearity which helps model complex functions, ones that cannot be modeled with plain linear regression

ReLU (Rectified Linear Unit)

- Most used activation function
- $ReLU(x) = max(0,x)$



Source: towardsdatascience.com

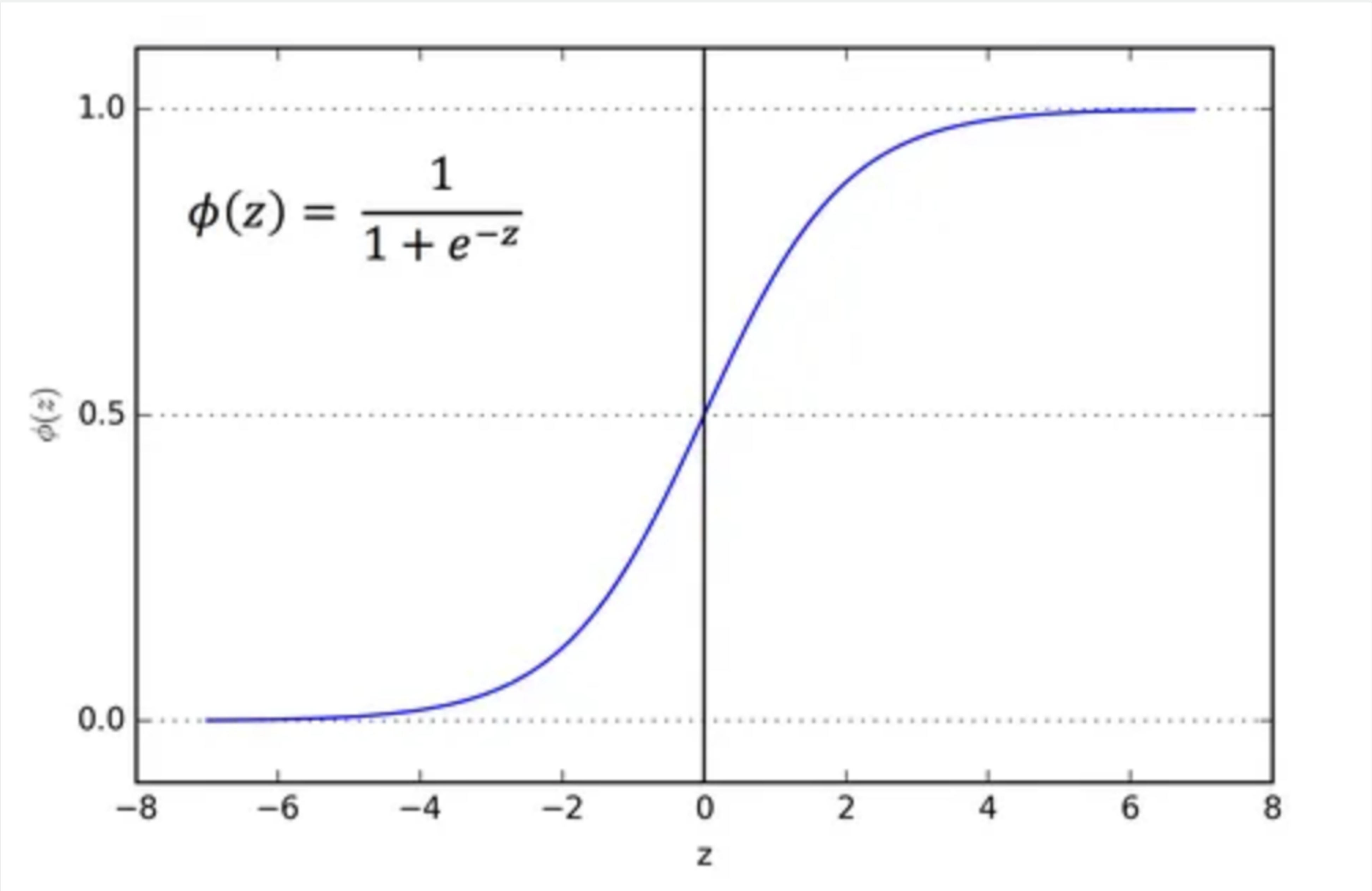
Activation Functions

Sigmoid

•The output of sigmoid lies between 0 and 1

•Therefore used for models where
a probability needs to be predicted as the output

Sigmoid (z) = 1/(1+e^(-z))



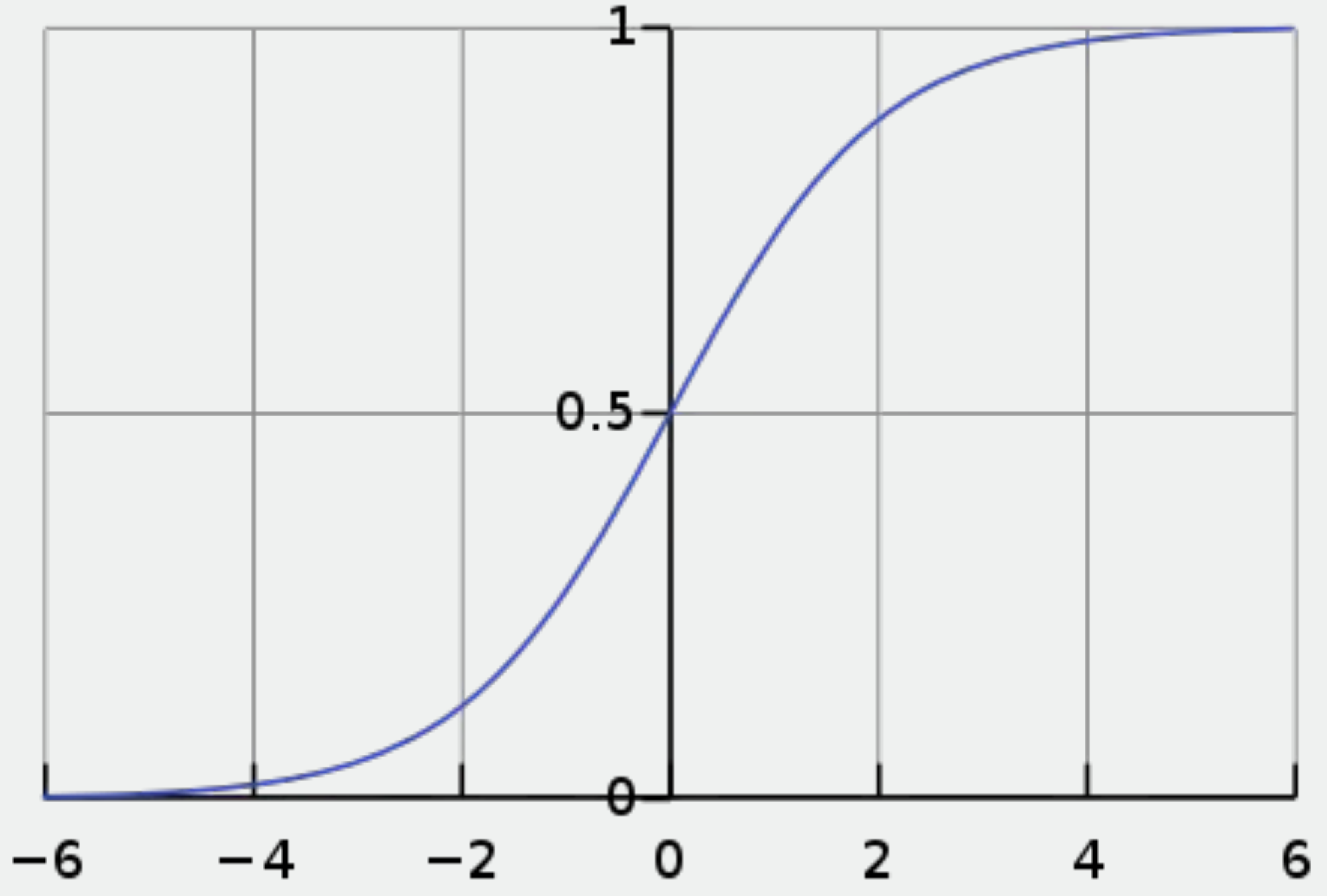
Source: towardsdatascience.com

Activation Functions

Softmax

- Softmax turns a vector of K real values into a vector of K real values that sum to 1
- It is used to obtain the confidence scores for NN output labels

$$Softmax(z) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$



Source: medium.com

Confidence score

- In neural networks used as classifiers, input $\vec{x}$ mapped to one of m classes
- Final layer typically employs a softmax function to represent output as normalized probabilities
- We use the term confidence, $conf(f(\vec{x}))$ to refer to the highest probability

$$conf(f(\vec{x})) = max(out(v_{l,1}), \ldots, out(v_{l,n}))$$

State-of-art

Local robustness - NN's ability to withstand adversarial inputs in the vicinity of a specific point in the input space

"Small" changes in input -> "Small" changes in output

$$\|\vec{x} - \vec{x_0}\| \leq \epsilon \rightarrow class(f(\vec{x})) = class(f(\vec{x_0}))$$

This is called local robustness because we are checking for perturbations in an ϵ-radius circle around a fixed $\vec{x_o}$

State-of-art

- SMT-based NN verification tools, such as Marabou [M1], take a fully-connected feed-forward neural network along with the local robustness property that needs to be verified
- Network encoded as a set of linear constraints representing weighted sum of neurons' outputs and a set of non-linear constraints defining activation functions
- The property is a set of constraints on the network's inputs and outputs
- The neurons are treated as variables - the verification problem thus involves identifying a variable assignment that satisfies all constraints at the same time
- Pass the constraints corresponding to the NN and the property (in negated form) to an SMT solver to find a satisfying assignment

Local robustness - Limitations

- Local robustness is defined only for a specific input
- Consequently, it does not provide any guarantees for any other input
- It follows that the robustness of the entire neural network cannot be assessed with local robustness only

Narrow, limited approach

Need a broader perspective when certifying NNs - one that covers the entire input space

Global robustness - General Definition

- Global robustness is not limited to analyzing robustness around a fixed point
- It is a measure of a NN's robustness over the entire input space, rather than specific points
- The local robustness definition that we saw can be generalized over all inputs to get global robustness

$$\forall \vec{x}, \overrightarrow{x'} \;\; \|\vec{x} - \overrightarrow{x'}\| \leq \epsilon \;\rightarrow\; class(f(\vec{x})) = class(f(\overrightarrow{x'}))$$

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

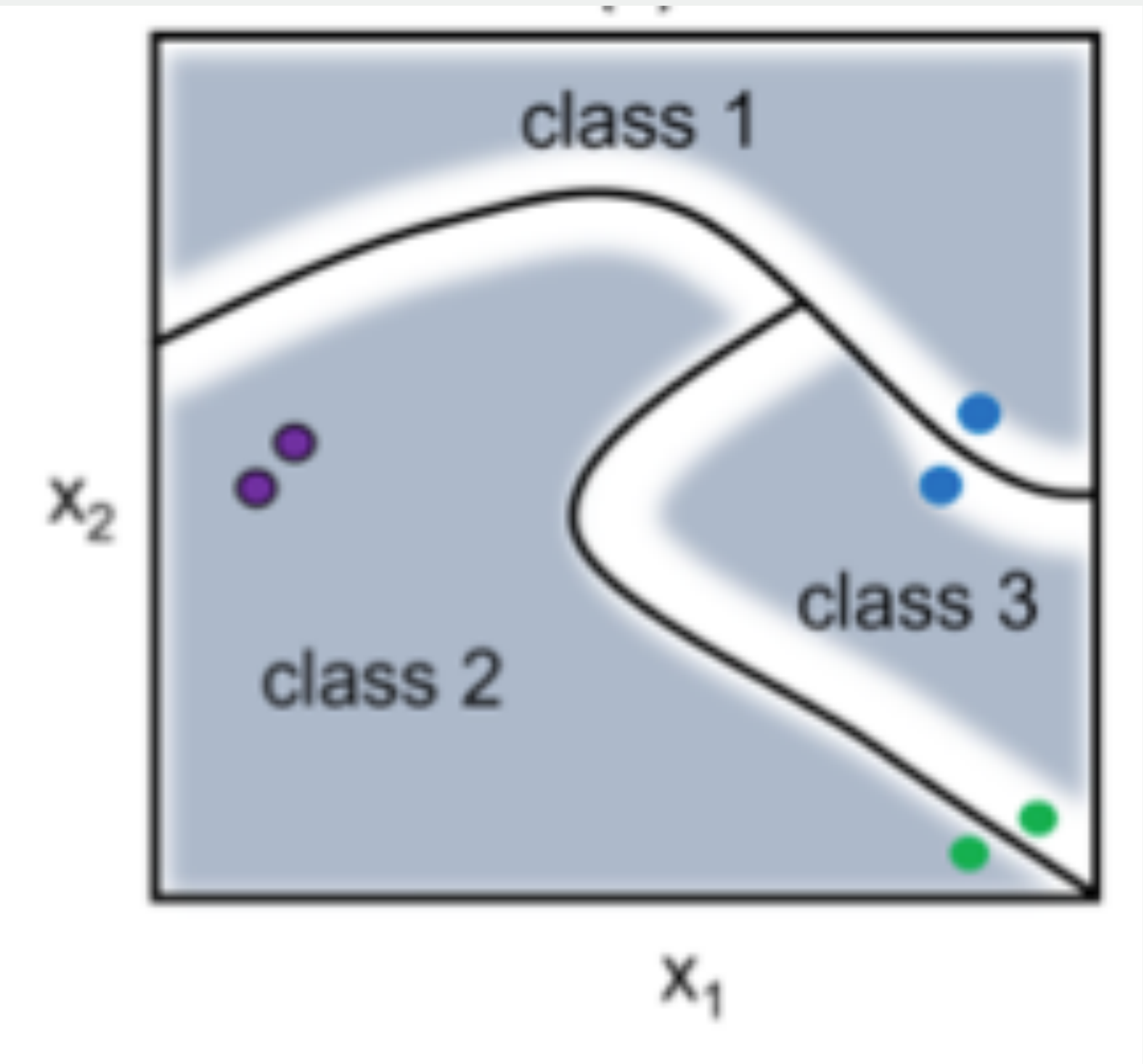
Global robustness - Limitation

$$\forall \vec{x}, \overrightarrow{x'} \ \ \|\vec{x} - \overrightarrow{x'}\| \leq \epsilon \rightarrow \ class(f(\vec{x})) = class(f(\overrightarrow{x'}))$$

- This definition of global robustness, however, can only be satisfied by trivial models that map all inputs to a single class
- This needs to be relaxed in order to make it practically useful for verification of NNs

Confidence-based global robustness

- At decision boundaries, the confidence for one label gradually decreases whereas the confidence in another one gradually increases
- At transition point, confidences in both labels lower and close to each other
- Hence, we do not consider pairs of inputs whose labels differ, but with a low level of confidence, to be counterexamples to robustness
- The idea is to compare all input pairs which are sufficiently close and for which at least one of them yields a high-confidence classification



TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

Global Fairness

- CertiFair [C1] and Fairify [F1] address a similar problem, which arises in the context of fairness, by partitioning the input space based on categorical features

- In general, if the input to a decision-making neural network comprises of certain sensitive attributes, say age or gender, the network is said to be fair if the sensitive attributes do not influence its decision

- For example, a hiring algorithm that discriminates against certain groups of job applicants based on their race or gender could perpetuate existing biases and inequalities in the workplace

$$\forall \vec{x} = (x_s, \overrightarrow{x_n}), \overrightarrow{x'} = (x'_s, \overrightarrow{x'_n}) . \ \ ||\overrightarrow{x_n} - \overrightarrow{x'_n}|| \leq \epsilon \ \wedge (x_s \neq x'_s) \rightarrow class(f(\vec{x})) = class(f(\overrightarrow{x'}))$$

where x_s and x_n: sensitive and non-sensitive attributes of $\vec{x}$, respectively

Global robustness and fairness are hyperproperties

- Observe that global robustness and fairness are hyper properties
- Properties capture relationships between multiple execution traces are known as hyperproperties
- Traditional properties, in contrast, are evaluated over individual traces
- A hyperproperty on the other hand quantifies over more than one trace

$\forall \vec{x} \, . \, conf(f(\vec{x})) \geq \kappa$ ← Traditional property

$$\forall \vec{x}, \overrightarrow{x'} \, . \, \frac{f(\vec{x})_i - f(\overrightarrow{x'})_i}{||\vec{x} - \overrightarrow{x'}||} \leq \kappa$$ ← Hyperproperty

Global robustness and fairness are hyperproperties

- Observe that global robustness and fairness are hyperproperties

Global robustness

$$\forall \vec{x}, \overrightarrow{x'} \;\; \|\vec{x}-\vec{x}\| \leq \epsilon \rightarrow$$
$$class(f(\vec{x})) = class(f(\overrightarrow{x'}))$$

Global fairness

$$\forall \vec{x} = (x_s, \overrightarrow{x_n}), \overrightarrow{x'} = (x'_s, \overrightarrow{x'_n}) . \;\; || \overrightarrow{x_n} - \overrightarrow{x'_n} || \leq \epsilon \; \wedge (x_s \neq x'_s) \rightarrow$$
$$class(f(\vec{x})) = class(f(\overrightarrow{x'}))$$

where x_s and x_n: sensitive and non-sensitive attributes of $\vec{x}$, respectively

- We used this striking similarity of the two properties to formalize the first definition of confidence-based 2-safety property that unifies global robustness and fairness for DNNs

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

Confidence-based global 2-safety - Definition

- A model f is said to be globally 2-safe for confidence κ > 0 and tolerance ϵ iff:

$$\forall \vec{x}, \vec{x}' \,.\, cond(\vec{x}, \vec{x}', \vec{\epsilon}) \land conf(f(\vec{x})) > \kappa \implies class(f(\vec{x})) = class(f(\vec{x}'))$$

For confidence-based global robustness:

$$cond(\vec{x}, \vec{x}', \vec{\epsilon}) = \bigwedge_{i \in [1,n]} d(x_i, x_i') \leq \epsilon_i$$

For confidence-based global fairness:

$$cond(\vec{x}, \vec{x}', \vec{\epsilon}) = \bigwedge_{x_i \in \vec{x_s}} d(x_i, x_i') > 0 \;\land \bigwedge_{x_i \in \vec{x_n}} d(x_i, x_i') \leq \epsilon_i$$

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

Confidence-based global 2-safety - Challenges

- We have now crossed the first hurdle by defining the property that we want to check

- However, there are several challenges:

  1. How to verify a 2-safety property?

  2. The presence of confidence in the definition of the property means we have to deal with non-linear softmax

1. How to verify a 2-safety property?

Self-composition

- 2-safety properties can be verified using self-composition

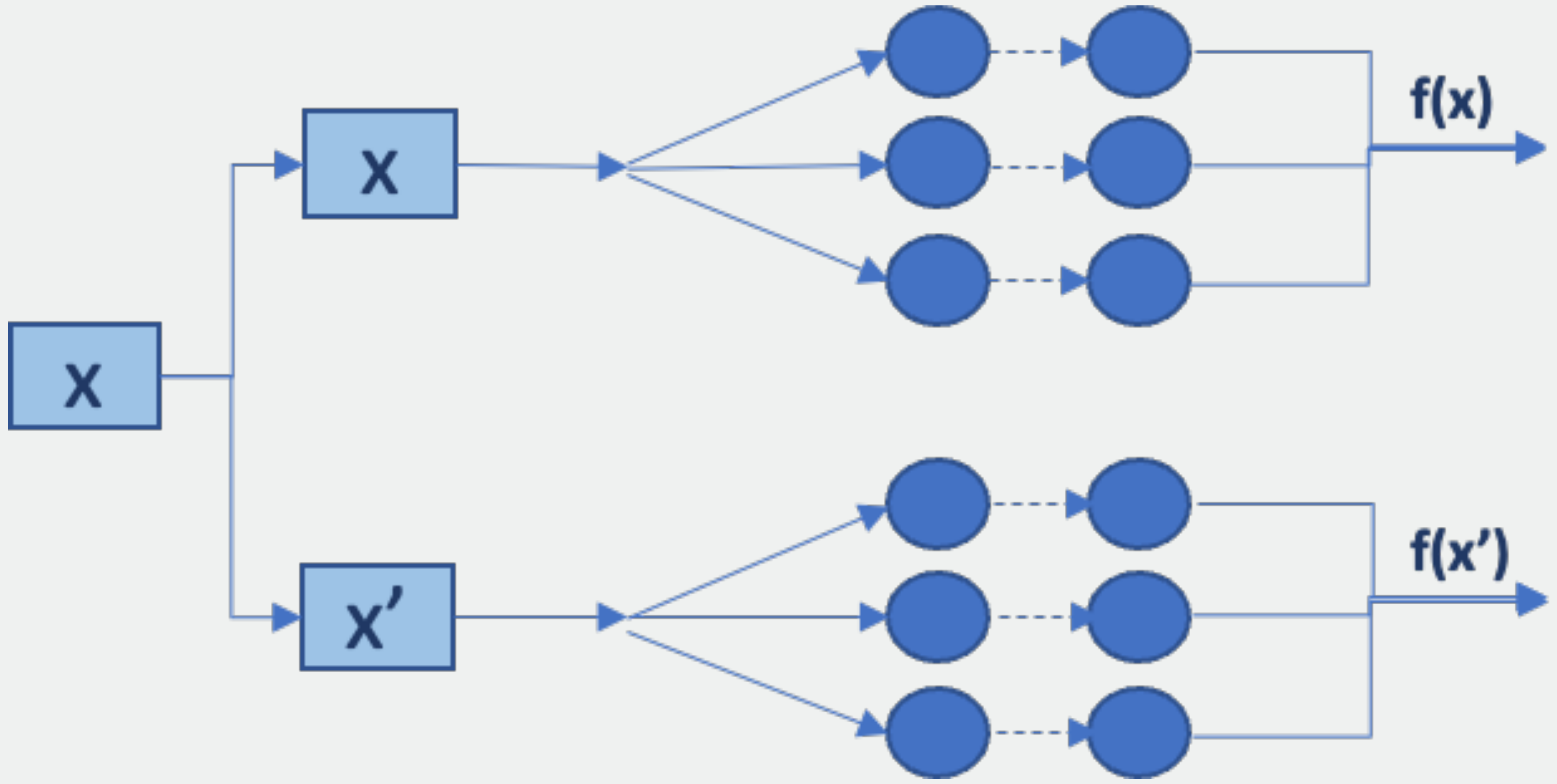
- The idea is to compose the program with itself and relate the two executions

$$f(\vec{x}) \times f(\overrightarrow{x'}) = \lambda(\vec{x}, \overrightarrow{x'}) \,.\, (f(\vec{x}), f(\overrightarrow{x'}))$$

where, $(\vec{x}, \overrightarrow{x'})$: concatenation of vectors $\vec{x}$ and $\overrightarrow{x'}$

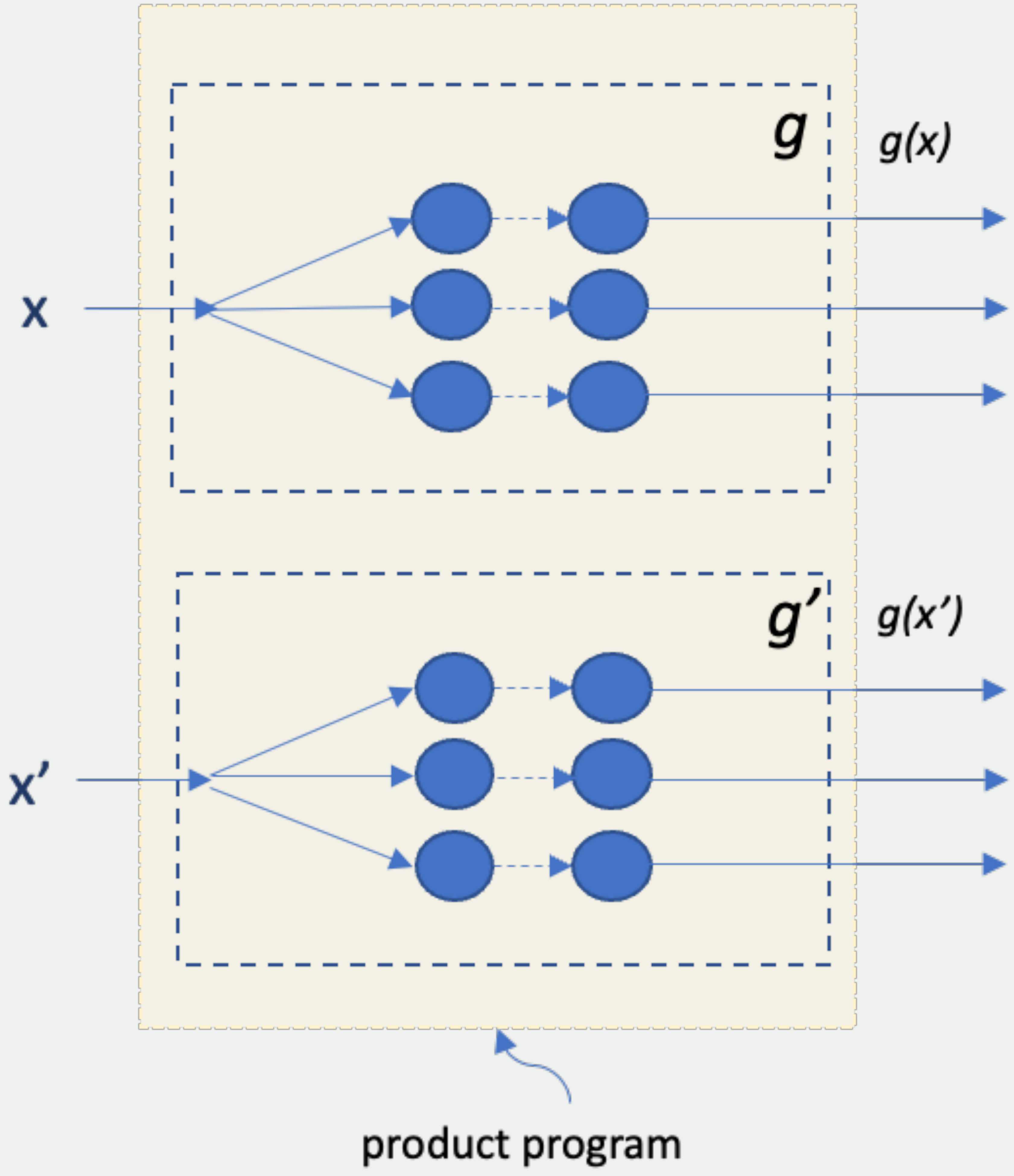
$\lambda\vec{x} \,.\, f(\vec{x})$: lambda term that binds $\vec{x}$ in $f(\vec{x})$

- A counterexample to a 2-safety property comprises of a pair of traces



Encoding 2-safety as Product Neural Network

- Compose a copy of the neural network with itself to get a product neural network
- The self-composed neural network consists of two copies of the original neural network, each with its own copy of the variables
- To encode the self-composition, we duplicate all variables and constraints by introducing primed counterparts $in'_{i,j}$ and $out'_{i,j}$ for $in_{i,j}$ and $out_{i,j}$
- Checking 2-safety then reduces to checking an ordinary safety property
- A product network allows the reduction of a 2-safety to a trace property, a problem, which can be solved using an existing standard verification technique
-



TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

2. How to deal with non-linear softmax to model confidence

Confidence - 2-step approach

Step 1

- To model confidence, we needed a way to find an abstraction of the softmax, which is amenable to automated verification

- Our approximation of the softmax involves a 2-step approach

- In the first part of the approximation, we express softmax in terms of log-sum-exp (LSE) and sigmoid

$$softmax(\vec{z}_i) = sigmoid(z_i - LSE^{n}_{1\,j \neq i}(z_j))$$

- Also, from [P1] we know that:

$$max^n_1(z_i) \leq LSE^n_1(z_i) \leq max^n_1(z_i) + log(n)$$

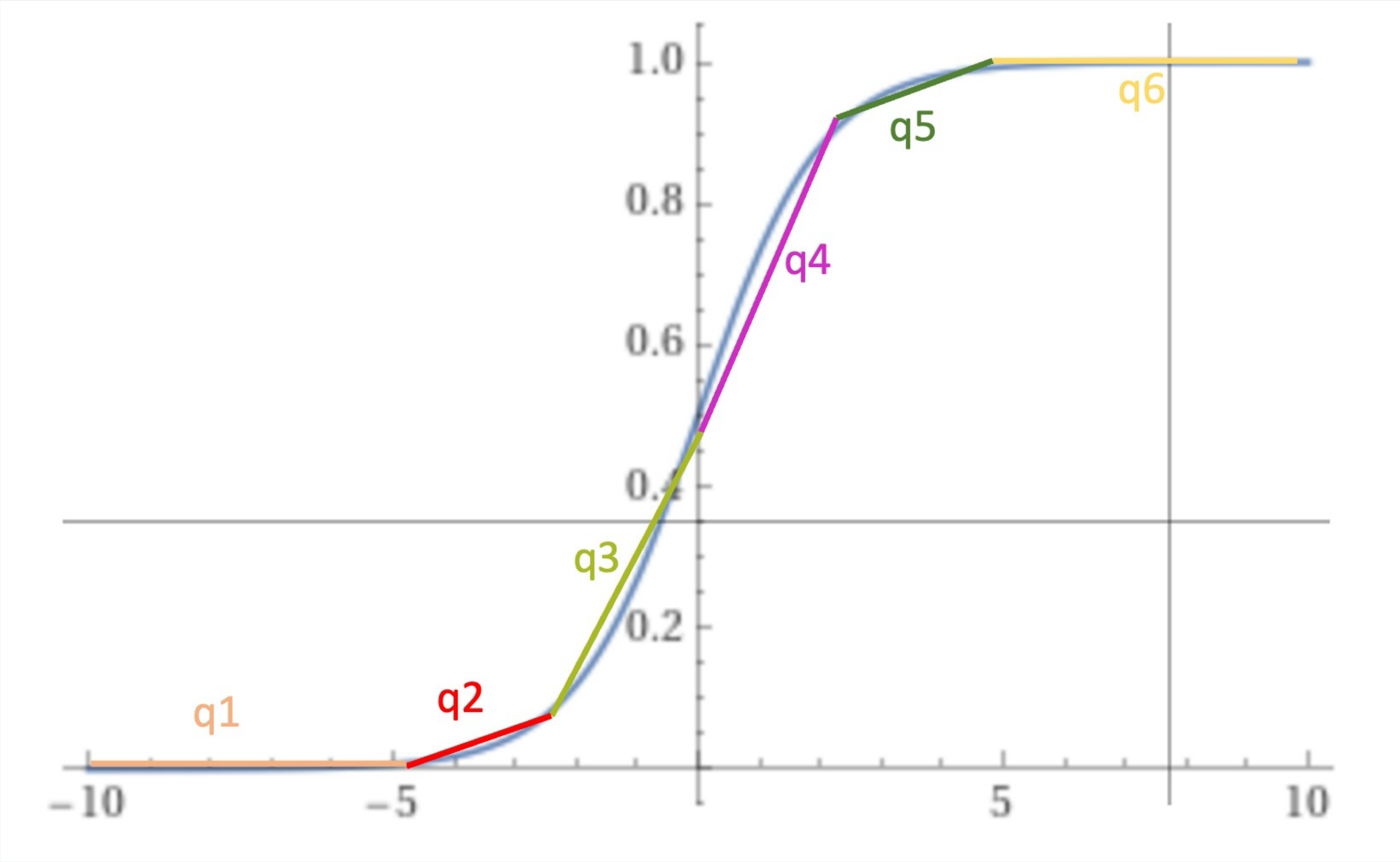
- We use these equations to express softmax in terms of sigmoid and max

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

Confidence - 2-step approach

Step 2

- We still do not know how to deal with sigmoid
- We approximate sigmoid as a piece-wise linear function using the Remez exchange algorithm [R1].
- Remez algorithm - iterative algorithm that finds simpler approximations to functions
- Set error to 0.005 -> obtain 35 segments -> encode each segment as an equation and represent using variable q_j
- Select applicable segment



Soundness

- For our confidence-based 2-safety property, our analysis provides a soundness guarantee

- This means that whenever the analysis reports that the property holds, then the property also holds true in the concrete execution

“ **Theorem 3.** *(Soundness) Let f and $\hat{f}$ be the original neural network and over-approximated neural network, respectively. Let $b_{n,\delta}$ be the error bound of the approximated softmax ($b_{n,\delta} = \frac{n-2}{(\sqrt{n-1}+1)^2} + 2\delta$ (see Theorem $\boxed{1}$)). Then we have the following soundness guarantee: Whenever the approximated neural network is 2-safe for $\mathrm{conf}(\hat{f}(\vec{x})) > (\kappa - b_{n,\delta})$, the real neural network is 2-safe for $\mathrm{conf}(f(\vec{x})) > \kappa$, given $\mathrm{conf}(\hat{f}(\vec{x})) > \frac{1}{2}$. Formally:*

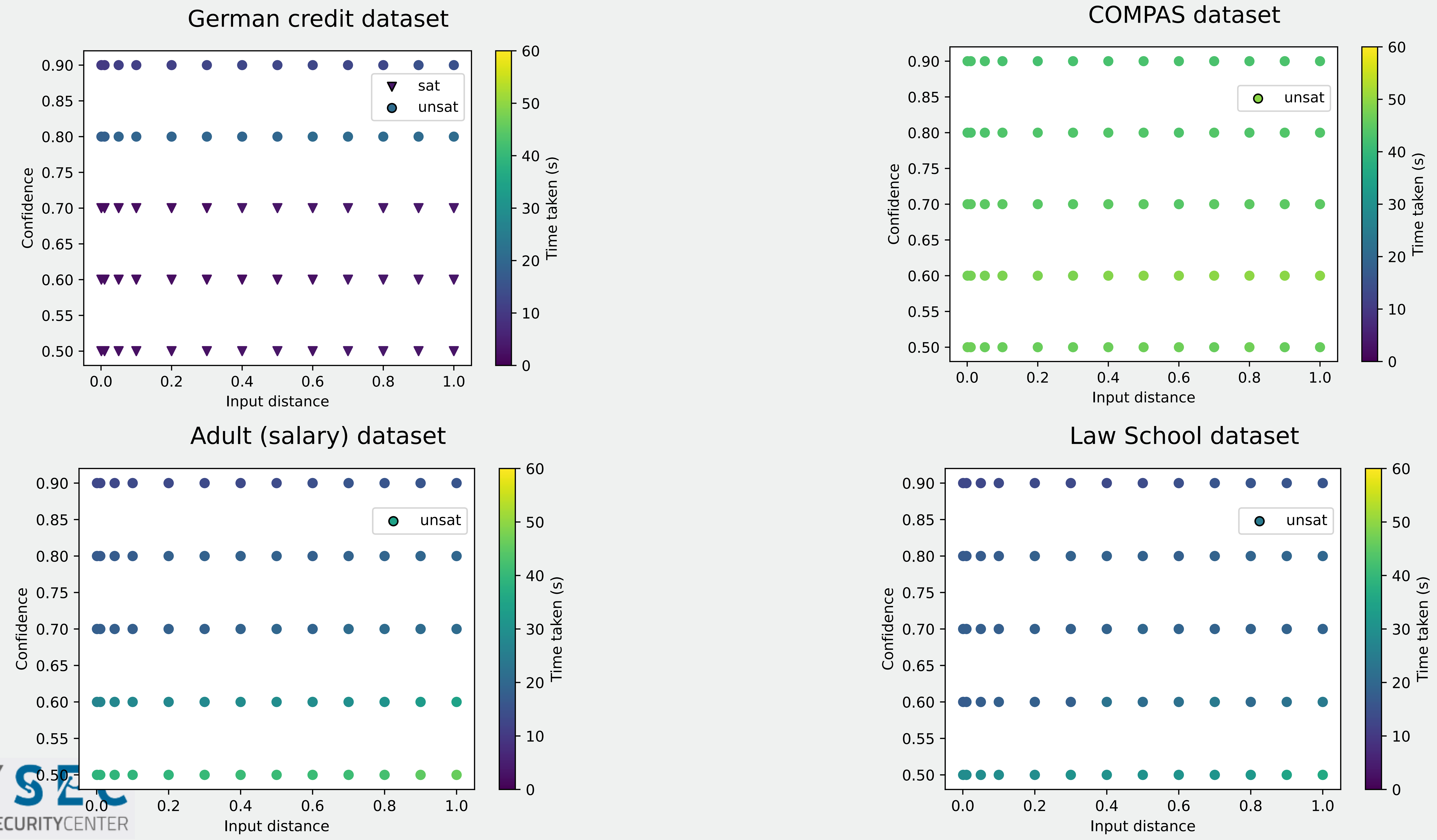
$$\begin{pmatrix} \forall \vec{x}, \vec{x}'. \ \mathrm{cond}(\vec{x}, \vec{x}', \vec{\epsilon}) \wedge \mathrm{conf}(\hat{f}(\vec{x})) > (\kappa - b_{n,\delta}) \\ \implies \mathrm{class}(\hat{f}(\vec{x})) = \mathrm{class}(\hat{f}(\vec{x}')) \end{pmatrix} \implies$$
$$\begin{pmatrix} \forall \vec{x}, \vec{x}'. \ \mathrm{cond}(\vec{x}, \vec{x}', \vec{\epsilon}) \wedge \mathrm{conf}(f(\vec{x})) > \kappa \\ \implies \mathrm{class}(f(\vec{x})) = \mathrm{class}(f(\vec{x}')) \end{pmatrix}, \ \textit{with} \ \ \mathrm{conf}(\hat{f}(\vec{x})) > \frac{1}{2}$$
”

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

Implementation

- Our method is applicable to any off-the-shelf static analysis tool
- As a proof of concept, we implement it on the state-of-the-art NN verification tool - Marabou
- Simplex-based, linear programming verification tool
- Capable of addressing queries about network’s properties (such as local robustness) — by encoding them into constraint satisfaction problem
- Can only handle traditional safety properties

Experimental evaluation - Confidence-based global robustness



Experimental evaluation - Confidence-based global fairness

Dataset	Sensitive attribute	Confidence threshold	Result	Time taken
German credit	Gender	0.5	unsat	10.232 sec
German credit	Age	0.5	unsat	11.478 sec
COMPAS	Gender	0.5	sat	7.423 sec
COMPAS	Ethnicity	0.5	sat	18.293 sec
COMPAS	Ethnicity	0.99	sat	25.846 sec
COMPAS	Ethnicity	0.999	unsat	171 min 15 sec

Global fairness on German credit/COMPAS datasets for various criteria

Exploring the space of property parameters

- We combined our method with binary search, to synthesize the minimum confidence for which the DNN is globally robust or fair
- We perform the binary search:
    - Start with confidence 0.5
    - If the model is unsat, done!
    - Else, check for confidence *mid = (0.5 + 1)/2*, and continue in this way till we find the minimum confidence accurate to the nearest 0.05
- For instance, binary search combined with our method, on German credit gave us 0.75 (in 45 seconds) to be the minimum confidence for which the DNN is globally robust

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

Current and Future Work

- Scalablility
  - Pruning
  - Knowledge distillation
- Tighter softmax approximation
- A hybrid approach that leverages the strengths of both testing and verification
- Property-based testing for our 2-safety confidence based property

References

[C1] Khedr, H., & Shoukry, Y. (2023, June). Certifair: A framework for certified global fairness of neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 37, No. 7, pp. 8237-8245)Khedr, H., & Shoukry, Y. (2023, June). Certifair: A framework for certified global fairness of neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 37, No. 7, pp. 8237-8245).

[F1] Biswas, S., & Rajan, H. (2023, May). Fairify: Fairness verification of neural networks. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 1546-1558). IEEE.

[M1] Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., ... & Barrett, C. (2019). The marabou framework for verification and analysis of deep neural networks. In Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31 (pp. 443-452). Springer International Publishing.

[R1] Chalmers, B. L. (1976). The Remez exchange algorithm for approximation with linear restrictions. Transactions of the American Mathematical Society, 223, 103-131.

References

[G1] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples.

[P1] Pant, Y. V., Abbas, H., & Mangharam, R. (2017, August). Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications (CCTA)* (pp. 1235-1240). IEEE.

[SC1] Chen, Y., Wang, S., Qin, Y., Liao, X., Jana, S., & Wagner, D. (2021, November). Learning security classifiers with verified global robustness properties. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (pp. 477-494).

Verifying Global Two-Safety Properties in Neural Networks with Confidence

Thank You!

Questions?

Anagha Athavale
TU Wien and AIT Austrian Institute of Technology
anagha.athavale@tuwien.ac.at