DTU

c3.ai Workshop:
Machine Learning for a Resilient,
Secure, Carbon-Free Electricity Supply

Remove barriers for Machine Learning Applications in Power Systems

Spyros Chatzivasileiadis
Associate Professor, DTU

Joint work with Andreas Venzke, Georgios Misyris, Jochen Stiasny, and with Guannan Qu, Steven Low

Slides available at www.chatziva.com/spyros_c3ai.pdf

Machine Learning for Power Systems: Why?

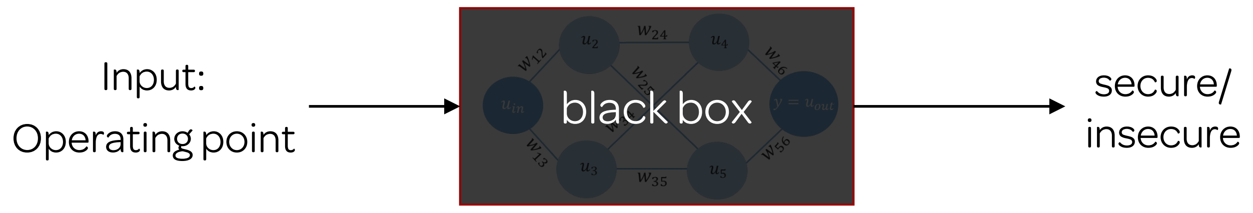
1. **Extremely fast**
   - computation within only a **few milliseconds**
     100x – 1000x faster than conventional methods → we can run 1'000 possible scenarios and get a good estimate vs. running only 1 scenario

2. Good alternative if we do not have full knowledge of the actual model
   - Handle **very complex systems**
   - **Infer** from incomplete data

But: Would an Operator ever trust AI in the Control Room?

Our work: Remove the barriers and build **trustworthy AI-tools**

ML Barriers for Power systems



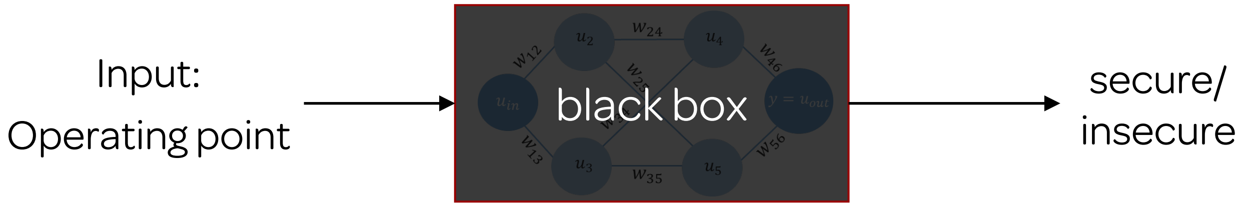
1. Why would we use a "**black box**" to decide about **a safety-critical application**?

2. Accuracy is a **purely statistical** performance metric. Who guarantees that the Neural Network can handle well previously unseen operating points?

3. Why would we depend on **discrete and incomplete data**, when we have developed **detailed physical models** over the past 100 years?

https://en.wikipedia.org/wiki/Garbage_in,_garbage_out

ML Barriers for Power systems



1. Why would we use a "**black box**" to decide about **a safety-critical application**?
2. **Accuracy is** a **purely statistical** performance metric. Who guarantees that the Neural Network can handle well previously unseen operating points?

→ **Neural Network verification**: <u>guarantees</u> for the NN performance!

3. Why would we depend on **discrete and incomplete data**, when we have developed **detailed physical models** over the past 100 years?

→ Physics-Informed Neural Networks

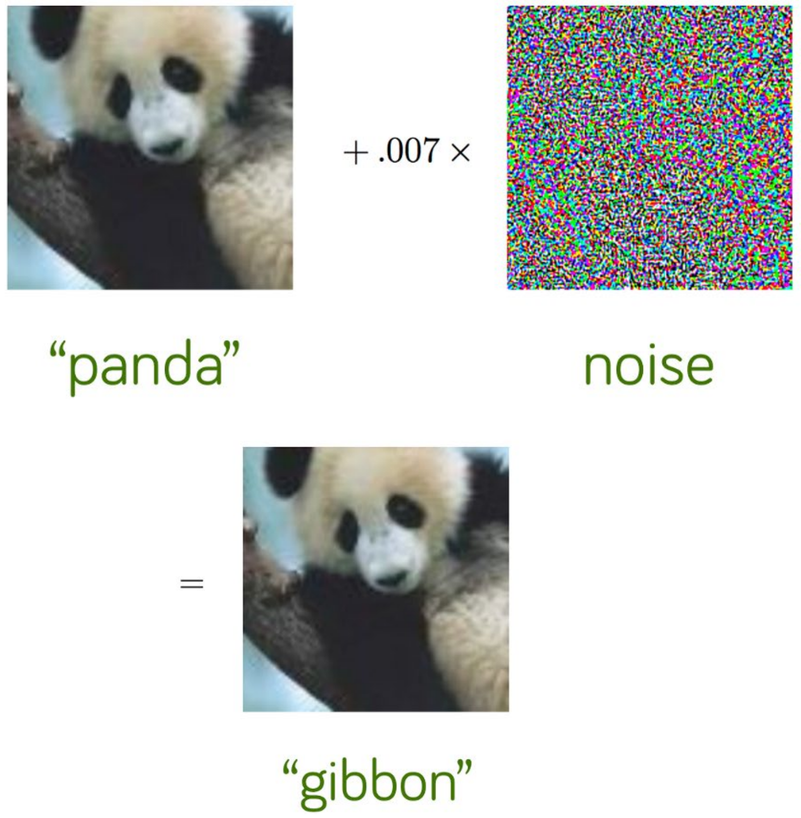
Accuracy is purely statistical!
(or why is Neural Network Verification important?)

NN accuracy: Until recently, the only way to assess the performance of neural networks

1. Challenge #1: No way to guarantee what the output is for a **continuous** range of inputs

2. Challenge #2: The **test database** determines the performance of the neural network
   - If the test data come from the same simulations as your training data → accuracy can be deceivingly high. Would it be equally high for the whole input domain?

3. Challenge #3: No way to systematically identify **adversarial examples**



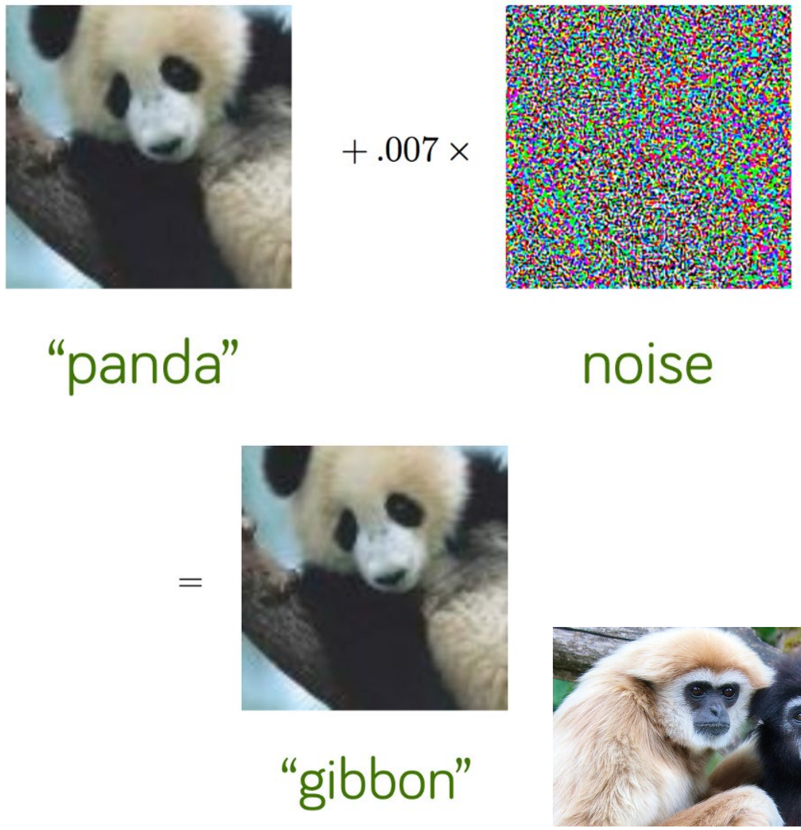
Accuracy is purely statistical!
(or why is Neural Network Verification important?)

NN accuracy: Until recently, the only way to assess the performance of neural networks

1. Challenge #1: No way to guarantee what the output is for a **continuous** range of inputs

2. Challenge #2: The **test database** determines the performance of the neural network
   - If the test data come from the same simulations as your training data → accuracy can be deceivingly high. Would it be equally high for the whole input domain?

3. Challenge #3: No way to systematically identify **adversarial examples**



Accuracy is purely statistical!
(or why is Neural Network Verification important?)

NN accuracy: Until recently, the only way to assess the performance of neural networks

1. Challenge #1: No way to guarantee what the output is for a **continuous** range of inputs
2. Challenge #2: The **test database** determines the performance of the neural network
   - If the test data come from the same simulations as your training data → accuracy can be deceivingly high. Would it be equally high for the whole input domain?
3. Challenge #3: No way to systematically identify **adversarial examples**

Neural network verification:

1. Can guarantee the output for a **continuous** range of inputs
2. Is **not dependent** on the quality of the test database
3. Can **systematically identify** adversarial examples

This talk:

1. Verification of **Classification Neural Networks** for Power Systems
2. Verification of **Regression Neural Networks**

---

3. **Physics-Informed** Neural Networks

Neural Network Verification
for classification NNs in Power Systems

A. Venzke, S. Chatzivasileiadis. Verification of Neural Network Behaviour: Formal Guarantees for Power System Applications. In *IEEE Transactions on Smart Grid*, vol. 12, no. 1, pp. 383-397, Jan. 2021, https://arxiv.org/pdf/1910.01624.pdf

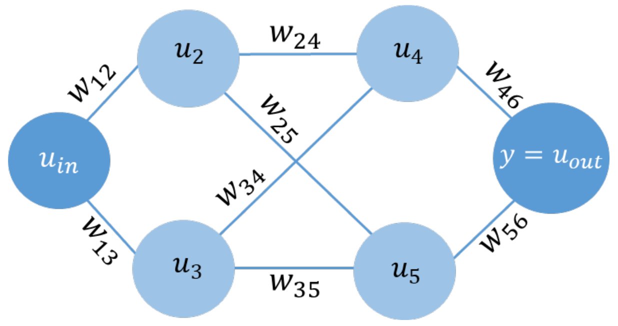
V. Tjeng, K. Y. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in International Conference on Learning Representations (ICLR 2019), 2019

Guiding Application: Security Assessment with Neural Networks

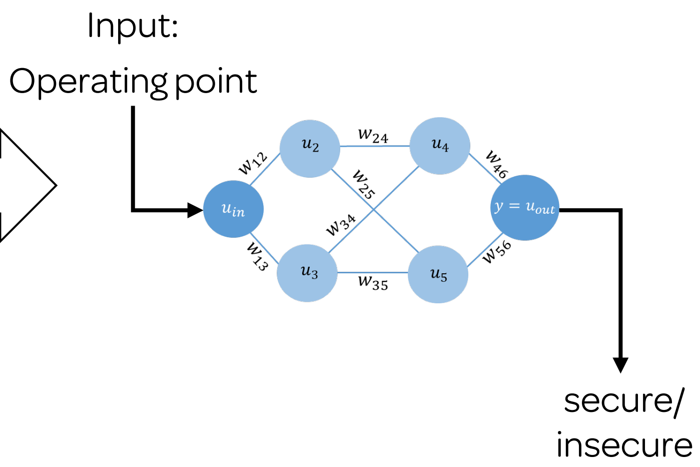
Approaches proposed up to now

Database of secure/insecure operating points

Any combination of **static/dynamic security** criteria
(e.g. N-1 & Small-signal stability)



5. Use the NN



1. Split the database in a training set and a test set

2. Train a neural network

3. Test the neural network

4. Is accuracy high enough?

NN Output:

Binary classification: **secure/insecure**

Extremely fast: up to 100x-1'000x faster

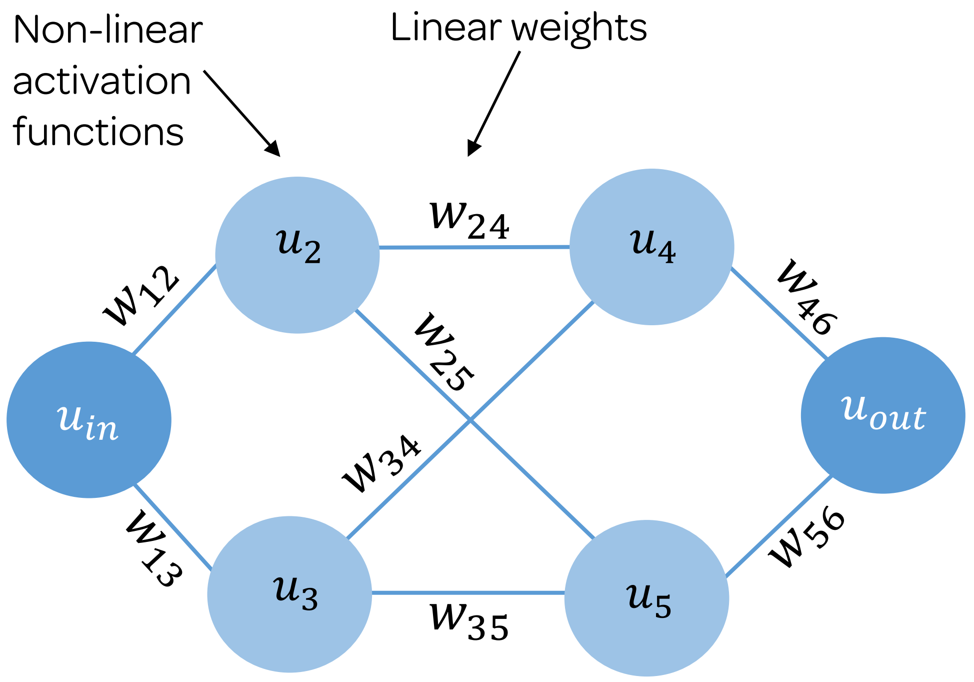
Neural Network Verification: HOW?

1. **Exact transformation:** Convert the neural network to a **set of linear equations with binaries**
   - The Neural Network can be included in a mixed-integer linear program

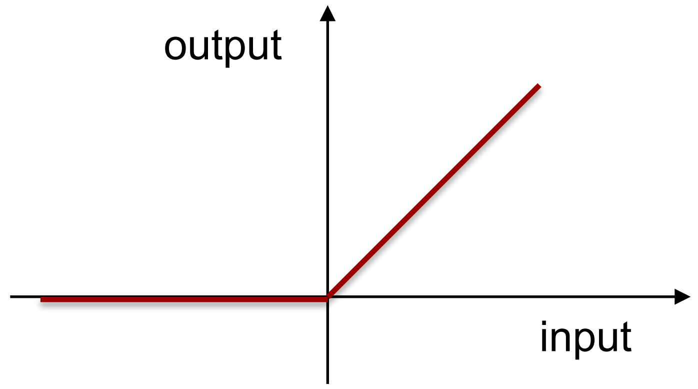
2. Formulate an **optimization** problem (MILP) and solve it → certificate for NN behavior

3. Assess if the neural network output complies with the ground truth

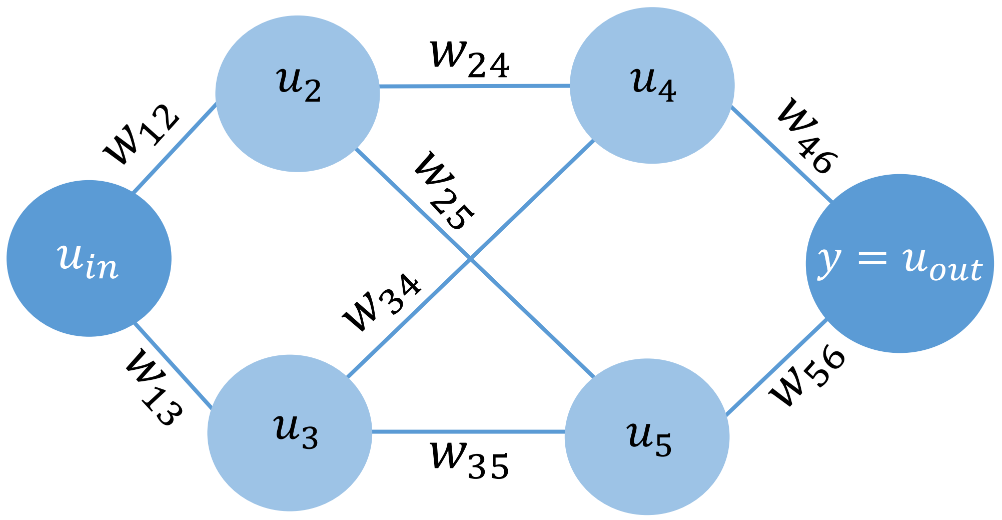
From Neural Networks to Mixed-Integer Linear Programming



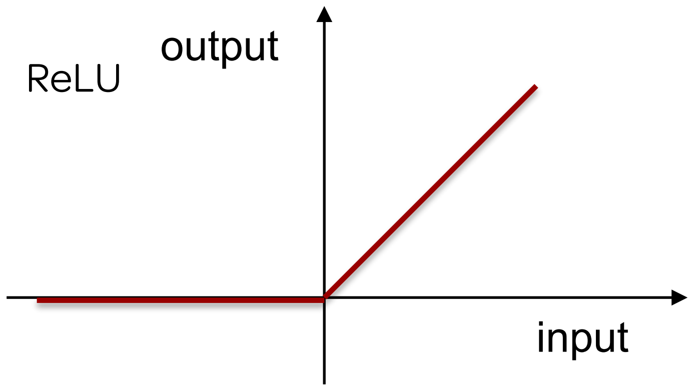
- Most usual activation function: ReLU
- ReLU: Rectifier Linear Unit



From Neural Networks to Mixed-Integer Linear Programming

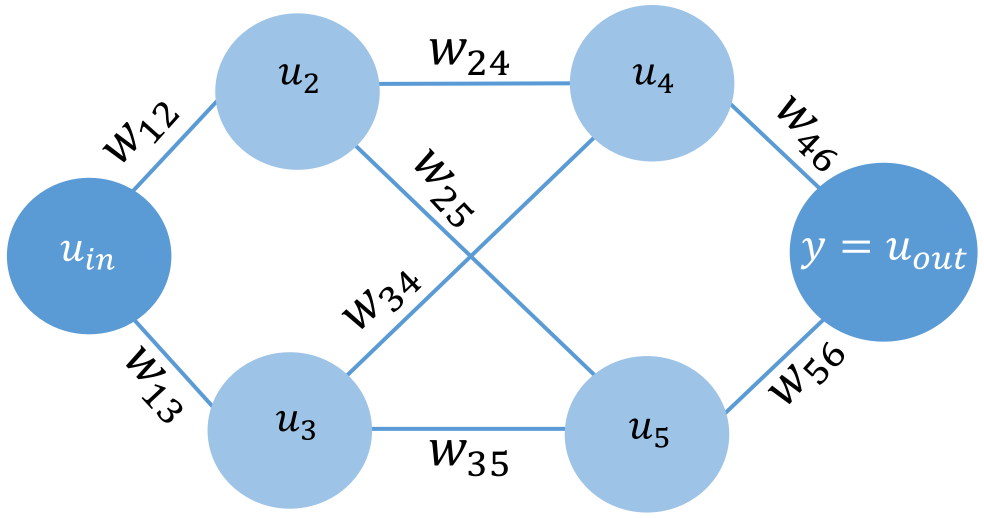


- Linear weights
- On every node: a non-linear activation function
  - ReLU: $\boldsymbol{u_j = \max(0, w_{ij}u_i + b_i)}$
- But ReLU can be transformed to a piecewise linear function with binaries



MILP

From Neural Networks to Mixed-Integer Linear Programming



- Input: Active power gen. setpoints

$$\mathbf{x} = [p_{g1}, p_{gi}, \ldots, p_{gN}]^T$$

- Output
  - Binary classification: safe/unsafe
  - Output vector y with two elements:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

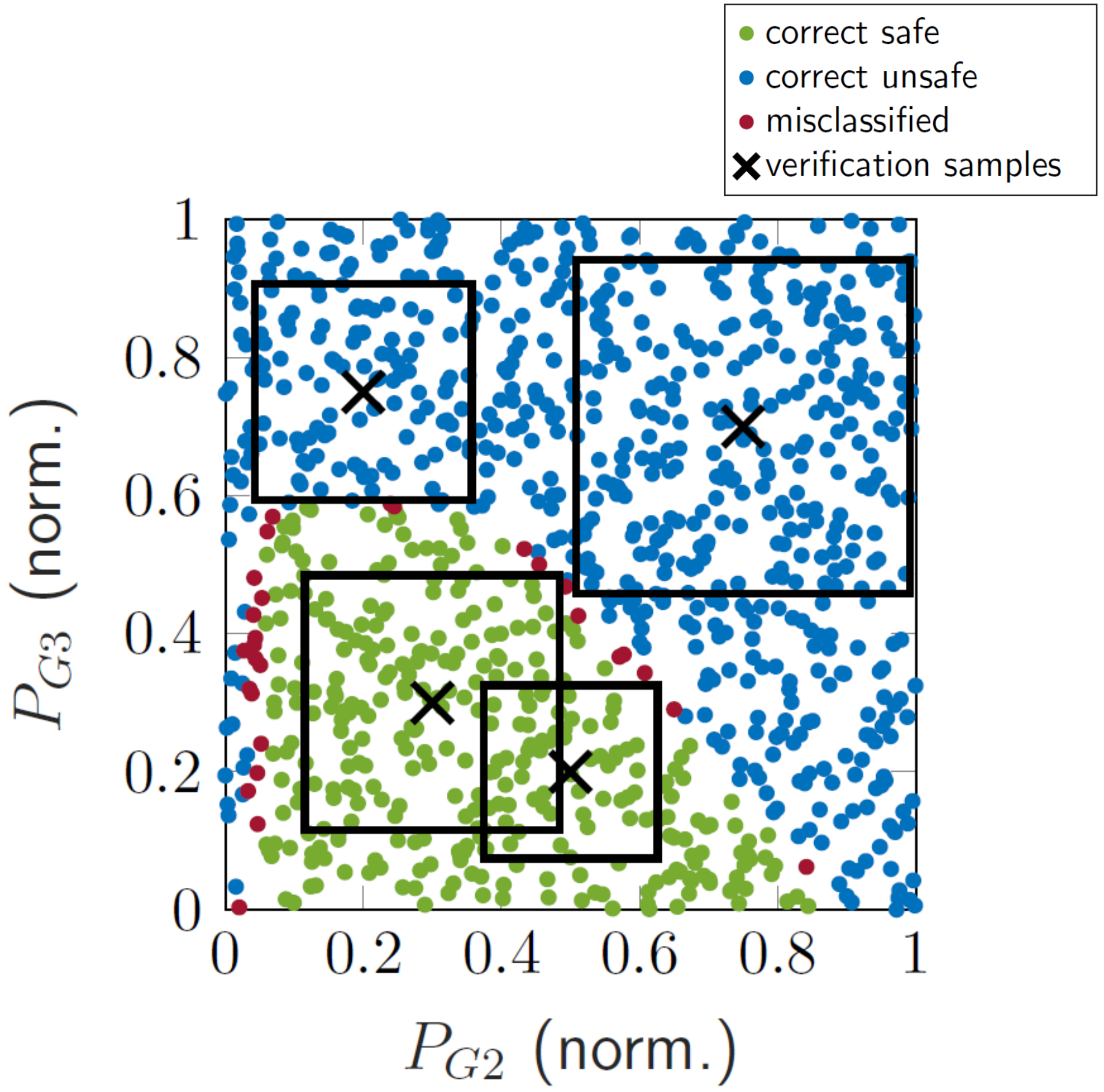
- $y_1 \geq y_2$: safe
- $y_1 < y_2$: unsafe

Certify the output for a continuous range of inputs

- We assume a given input x_{ref} with classification y: $y_1 > y_2$

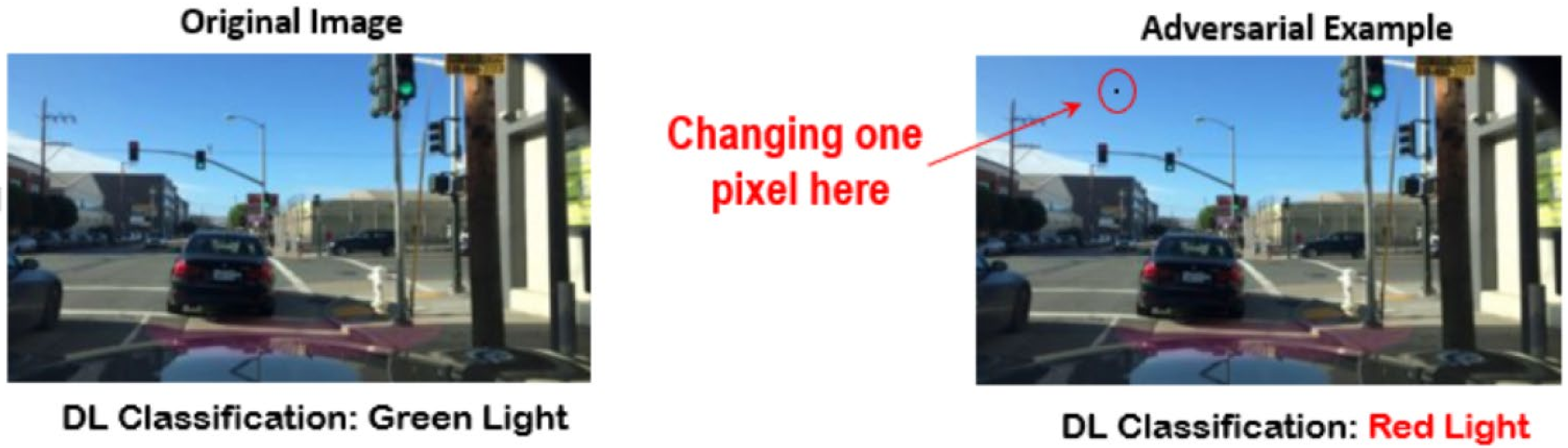
1. For distance ϵ evaluate if input x exists with different classification y_2

$$\max_{x,y} \quad \mathbf{y_2} - \mathbf{y_1}$$
$$\text{s.t.} \quad y = NN(x)$$
$$|\mathbf{x} - \mathbf{x}_{\text{ref}}|_\infty \leq \epsilon$$



A. Venzke, S. Chatzivasileiadis. Verification of Neural Network Behaviour: Formal Guarantees for Power System Applications. *IEEE Transactions on Smart Grid*, Jan. 2021. https://arxiv.org/pdf/1910.01624.pdf

Adversarial examples in safety-critical systems



source: Wu et al. A game-based approximate verification of deep neural networks with provable guarantees. arXiv:1807.03571.

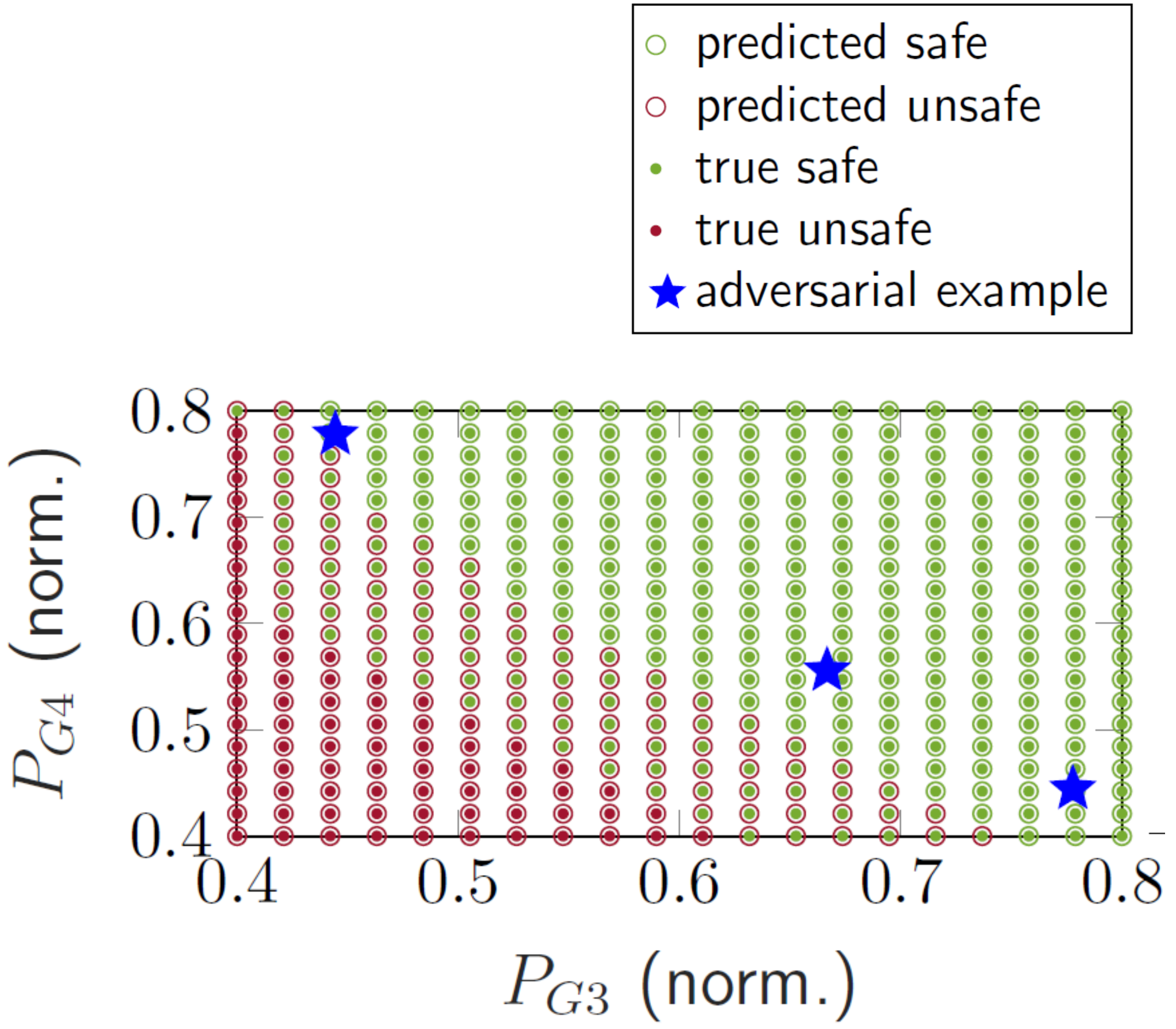
- Adversarial examples exist in many (deep) learning applications
- Major barrier for adoption of machine learning techniques in safety-critical systems!

Systematically identify adversarial examples

- We assume a given input x_{ref} with classification y: $y_1 > y_2$

2. Minimize distance ϵ from $\mathbf{x}_{\text{ref}}$ to input $\mathbf{x}$ with classification y_2

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\epsilon} \quad & \epsilon \\
\text{s.t.} \quad & \mathbf{y} = NN(\mathbf{x}) \\
& |\mathbf{x} - \mathbf{x}_{\text{ref}}|_\infty \le \epsilon \\
& y_2 \ge y_1
\end{aligned}
$$



Learning OPF: **Worst-case Guarantees for Regression Neural Networks**

A. Venzke, G. Qu, S. Low, S. Chatzivasileiadis, Learning Optimal Power Flow: Worst-case Guarantees for Neural Networks. **Best Student Paper Award** at IEEE SmartGridComm 2020. https://arxiv.org/pdf/2006.11029.pdf

Quick Reminder: DC Optimal Power Flow

- **Objective**: find the minimum cost generation dispatch
- **Input**: Varying load demand at different nodes

- Considered constant: generator costs; system topology

$$
\begin{aligned}
\min_{\mathbf{p}_{\mathrm{g}},\boldsymbol{\theta}} \quad & \mathbf{c}^T\mathbf{p}_{\mathrm{g}} && \text{Minimizes generation cost} \\
\text{s.t.} \quad & \mathbf{M}_{\mathrm{g}}\mathbf{p}_{\mathrm{g}} - \mathbf{M}_{\mathrm{d}}\mathbf{p}_{\mathrm{d}} = \mathbf{B}_{\mathrm{bus}}\boldsymbol{\theta} && \text{Nodal power balance} \\
& -\mathbf{p}_{\mathrm{line}}^{\mathrm{max}} \leq \mathbf{B}_{\mathrm{line}}\boldsymbol{\theta} \leq \mathbf{p}_{\mathrm{line}}^{\mathrm{max}} && \text{Transmission line limits} \\
& \mathbf{p}_{\mathrm{g}}^{\mathrm{min}} \leq \mathbf{p}_{\mathrm{g}} \leq \mathbf{p}_{\mathrm{g}}^{\mathrm{max}} && \text{Generator limits}
\end{aligned}
$$

Quick Reminder: DC Optimal Power Flow

- **Objective**: find the minimum cost generation dispatch
- **Input**: Varying load demand at different nodes

- Considered constant: generator costs; system topology

Several recent approaches in the literature that apply Neural Networks for solving the DC-OPF

- Demonstrate up to **100x speedup**
- But **no performance guarantees**

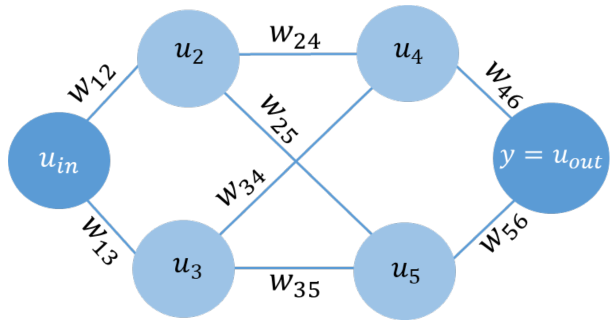
$\min\limits_{\mathbf{p}_{\mathrm{g}},\boldsymbol{\theta}}$	$\mathbf{c}^T\mathbf{p}_{\mathrm{g}}$	Minimizes generation cost
s.t.	$\mathbf{M}_{\mathrm{g}}\mathbf{p}_{\mathrm{g}} - \mathbf{M}_{\mathrm{d}}\mathbf{p}_{\mathrm{d}} = \mathbf{B}_{\mathrm{bus}}\boldsymbol{\theta}$	Nodal power balance
	$-\mathbf{p}_{\mathrm{line}}^{\mathrm{max}} \leq \mathbf{B}_{\mathrm{line}}\boldsymbol{\theta} \leq \mathbf{p}_{\mathrm{line}}^{\mathrm{max}}$	Transmission line limits
	$\mathbf{p}_{\mathrm{g}}^{\mathrm{min}} \leq \mathbf{p}_{\mathrm{g}} \leq \mathbf{p}_{\mathrm{g}}^{\mathrm{max}}$	Generator limits

Guiding Application for Regression NN: Learning OPF

Approaches proposed up to now

Database of varying loads and the related cost-optimal generation dispatch

1. Split the database in a training set and a test set

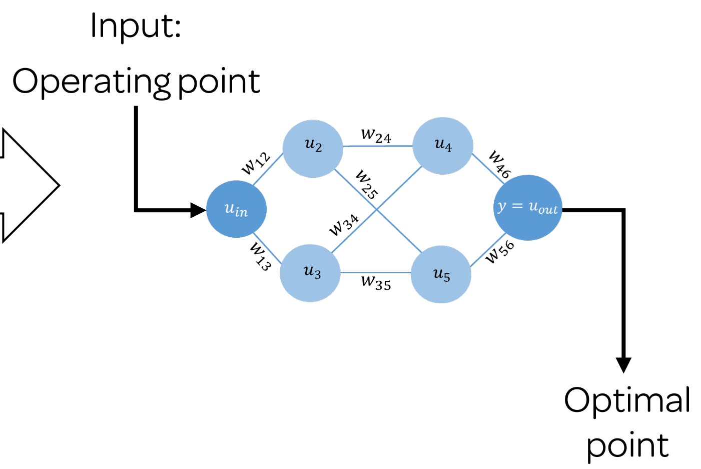


2. Train a neural network

3. Test the neural network

4. Is accuracy high enough?

5. Use the NN



NN Output:

Optimal generation dispatch

Extremely fast:

up to 100x faster

Part I: Maximum limit-violations

1. Maximum violation of generator limits

$$\nu_{\mathrm{g}} = \max(\hat{\mathbf{p}}_{\mathrm{g}} - \mathbf{p}_{\mathrm{g}}^{\mathrm{max}}, \mathbf{p}_{\mathrm{g}}^{\mathrm{min}} - \hat{\mathbf{p}}_{\mathrm{g}}, \mathbf{0})$$

$$\begin{aligned} \max \quad & \nu_{\mathrm{g}} & \\ \text{s.t.} \quad & \mathbf{A}_{\mathrm{d}}\mathbf{p}_{\mathrm{d}} \leq \mathbf{b}_{\mathrm{d}} & \text{Convex polytope as input domain } \mathcal{D} \\ & \hat{\mathbf{p}}_{\mathrm{g}} = NN(\mathbf{p}_{\mathrm{d}}) & \text{Mixed-integer reformulation of trained NN} \end{aligned}$$

Example:

$$0.6\, \boldsymbol{p}_{\mathrm{d}}^{\mathrm{max}} \leq \boldsymbol{p}_{\mathrm{d}} \leq 1.0\, \boldsymbol{p}_{\mathrm{d}}^{\mathrm{max}}$$

Part I: Maximum limit-violations

1. Maximum violation of generator limits

$$\nu_{\mathrm{g}} = \max(\hat{\mathbf{p}}_{\mathrm{g}} - \mathbf{p}_{\mathrm{g}}^{\mathrm{max}}, \mathbf{p}_{\mathrm{g}}^{\mathrm{min}} - \hat{\mathbf{p}}_{\mathrm{g}}, \mathbf{0})$$

$$\boxed{\begin{array}{lll} \max & \nu_{\mathrm{g}} & \\ \text{s.t.} & \mathbf{A}_{\mathrm{d}}\mathbf{p}_{\mathrm{d}} \leq \mathbf{b}_{\mathrm{d}} & \text{Convex polytope as input domain } \mathcal{D} \\ & \hat{\mathbf{p}}_{\mathrm{g}} = NN(\mathbf{p}_{\mathrm{d}}) & \text{Mixed-integer reformulation of trained NN} \end{array}}$$

Example:

$$0.6\, \boldsymbol{p}_{\mathrm{d}}^{\mathrm{max}} \leq \boldsymbol{p}_{\mathrm{d}} \leq 1.0\, \boldsymbol{p}_{\mathrm{d}}^{\mathrm{max}}$$

2. Maximum violation of line limits

$$\nu_{\mathrm{line}} = \max(\underbrace{|\mathbf{B}_{\mathrm{line}}\tilde{\mathbf{B}}_{\mathrm{bus}}^{-1}(\mathbf{M}_{\mathrm{g}}\hat{\mathbf{p}}_{\mathrm{g}} - \mathbf{M}_{\mathrm{d}}\mathbf{p}_{\mathrm{d}})^{\mathrm{nsb}}|}_{\text{Line flow equations for DC-OPF based on PTDFs}} - \mathbf{p}_{\mathrm{line}}^{\mathrm{max}}, \mathbf{0})$$

$$\boxed{\begin{array}{lll} \max & \nu_{\mathrm{line}} & \\ \text{s.t.} & \mathbf{A}_{\mathrm{d}}\mathbf{p}_{\mathrm{d}} \leq \mathbf{b}_{\mathrm{d}} & \text{Convex polytope as input domain } \mathcal{D} \\ & \hat{\mathbf{p}}_{\mathrm{g}} = NN(\mathbf{p}_{\mathrm{d}}) & \text{Mixed-integer reformulation of trained NN} \end{array}}$$

Worst violation over the **whole training dataset** (training+test set)

Our algorithm: **provable** worst-case guarantee over the **whole input domain**

	Empirical lower bound		Exact worst-case guarantee	
Test cases	ν_g (MW)	ν_{line} (MW)	ν_g (MW)	ν_{line} (MW)
case9				
case30				
case39				
case57				
case118				
case162				
case300				

ν_g	Maximum violation of generator limits
ν_{line}	Maximum violation of line limits

Worst violation over the **whole training dataset** (training+test set)

Our algorithm: **provable** worst-case guarantee over the **whole input domain**

	Empirical lower bound		Exact worst-case guarantee	
Test cases	ν_g (MW)	ν_{line} (MW)	ν_g (MW)	ν_{line} (MW)
case9	2.5	1.8	2.8	1.9
case30	1.7	0.6	3.6	3.1
case39	51.9	37.2	270.6	120.0
case57	4.2	0.0	23.7	0.0
case118	149.4	15.6	997.8	510.8
case162	228.0	180.0	1563.3	974.1
case300	474.5	692.7	3658.5	3449.3

ν_g	Maximum violation of generator limits
ν_{line}	Maximum violation of line limits

Over the whole input domain **violations can be much larger** (here ~7x) compared to what has been estimated empirically on the dataset

Worst violation over the **whole training dataset** (training+test set)

Our algorithm: **provable** worst-case guarantee over the **whole input domain**

Test cases	Empirical lower bound ν_g (MW)	Empirical lower bound ν_{line} (MW)	Exact worst-case guarantee ν_g (MW)	Exact worst-case guarantee ν_{line} (MW)
case9	2.5	1.8	2.8	1.9
case30	1.7	0.6	3.6	3.1
case39	51.9	37.2	270.6	120.0
case57	4.2	0.0	23.7	$\boxed{0.0}$
case118	149.4	15.6	997.8	510.8
case162	228.0	180.0	1563.3	974.1
case300	474.5	692.7	3658.5	3449.3

ν_g	Maximum violation of generator limits
ν_{line}	Maximum violation of line limits

Our method provides **guarantees that no NN output will violate the line limits** over the whole input domain

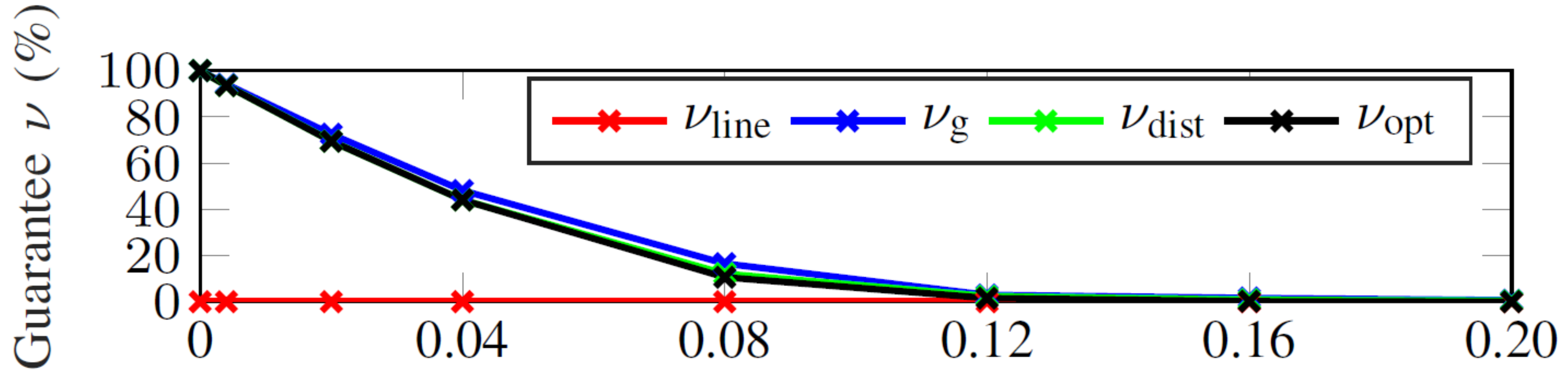
How can we reduce the worst-case violations?

- From our experiments with DC-OPF in 7 different test power systems, we observed that the **worst-case violations occur at the boundary of the input domain**

- Possible solution:
  1. Train on a larger input domain
  2. Use the NN on a subdomain of the original training input

Reducing the worst-case violations

100% on y-axis = Worst-case violation for $\delta = 0$



(b) *case57*: Input domain reduction δ (–)

- Input domain used for training

$$0.6\,\boldsymbol{p}_{\text{d}}^{\text{max}} \leq \boldsymbol{p}_{\text{d}} \leq 1.0\,\boldsymbol{p}_{\text{d}}^{\text{max}}$$

- Input domain for using the NN (and where worst-case violations were evaluated)

$$(0.6+\delta)\mathbf{p}_{\text{d}}^{\text{max}} \leq \mathbf{p}_{\text{d}} \leq (1.0-\delta)\mathbf{p}_{\text{d}}^{\text{max}}$$

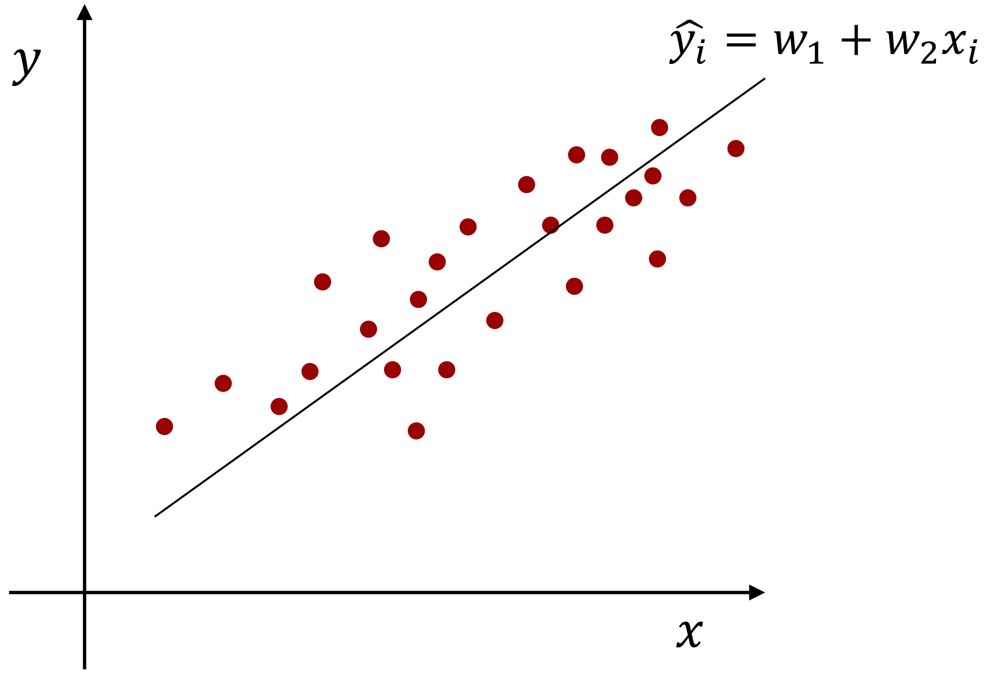
Example: If $\delta = 0.1$ then $0.7\,\boldsymbol{p}_{\text{d}}^{\text{max}} \leq \boldsymbol{p}_{\text{d}} \leq 0.9\,\boldsymbol{p}_{\text{d}}^{\text{max}}$

Physics-Informed Neural Networks for Power Systems

Neural Networks: An advanced form of non-linear regression

y_i: actual/correct value

$\hat{y}_i$: estimated value



Loss function: Estimate best w_1, w_2 to fit the training data

$$\min_{w_1,w_2} \quad \|y_i - \hat{y}_i\|$$

s.t.

$$\hat{y}_i = w_1 + w_2 x_i \qquad \forall i$$

Traditional training of neural networks required no information about the underlying physical model. Just data!

Physics Informed Neural Networks

- Automatic differentiation: derivatives of the neural network output with respect to the input can be computed during the training procedure
- A differential-algebraic model of a physical system can be included in the neural network training*
- Neural networks can now exploit knowledge of the actual physical system
- Machine learning platforms such as Tensorflow enable these capabilities

*M. Raissi, P. Perdikaris, and G. Karniadakis, Physics-Informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations", Journal of Computational Physics, vol.378, pp. 686-707, 2019

Physics-Informed Neural Networks for Power Systems

"Original" Loss function

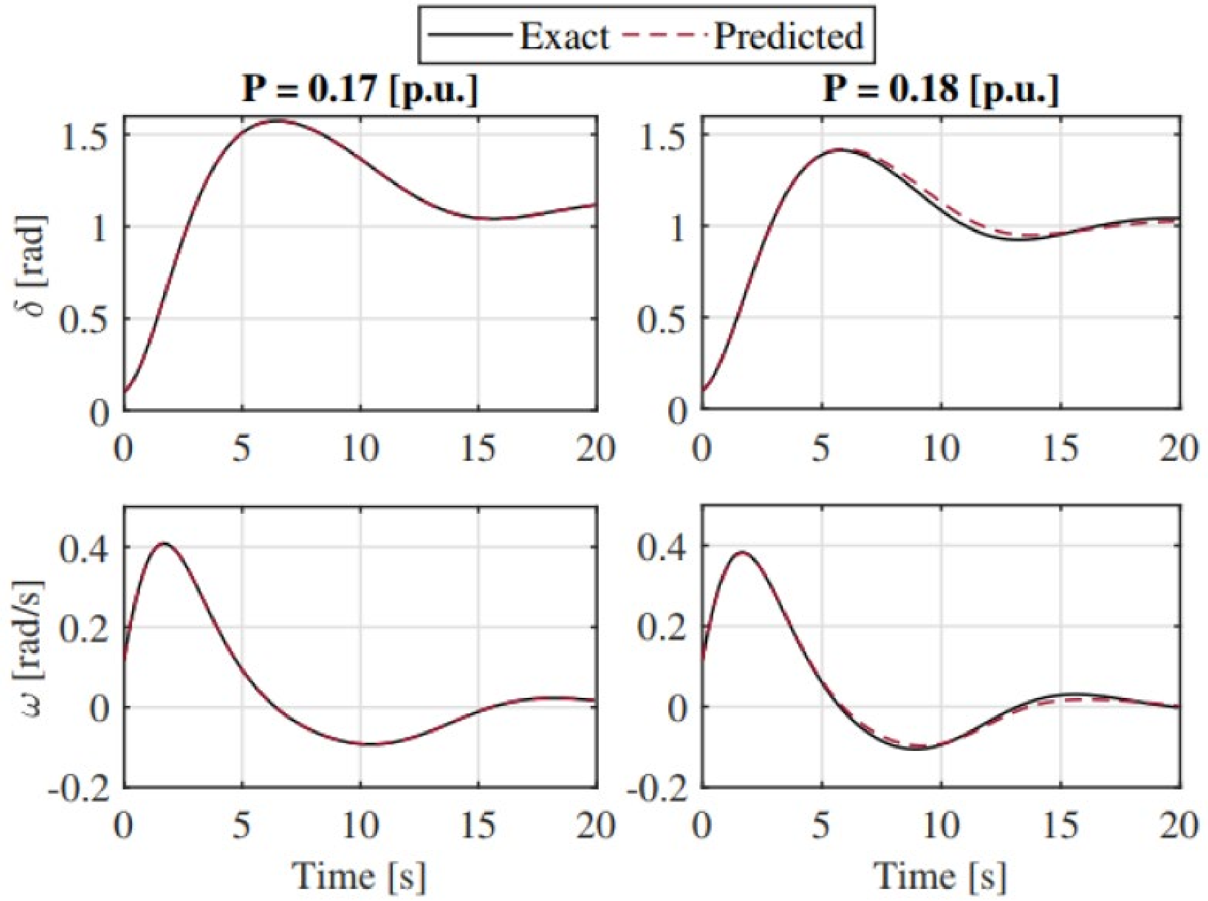
$$\boxed{\min_{\mathbf{W},\mathbf{b}} \quad \frac{1}{|N_\delta|}\sum_{i\in N_\delta}|\hat{\delta}-\delta^i|^2} + \frac{1}{|N_f|}\sum_{i\in N_f}|f(\hat{\delta})|^2 \tag{6a}$$

$$s.t. \qquad \hat{\delta} = NN(t, P_m, \mathbf{W}, \mathbf{b}) \tag{6b}$$

$$\dot{\hat{\delta}} = \frac{\partial\hat{\delta}}{\partial t}, \qquad \ddot{\hat{\delta}} = \frac{\partial\dot{\hat{\delta}}}{\partial t} \tag{6c}$$

$$f(\hat{\delta}) = M\ddot{\hat{\delta}} + D\dot{\hat{\delta}} + A\sin\hat{\delta} - P_m \tag{6d}$$

Swing equation



G. S. Misyris, A. Venzke, S. Chatzivasileiadis, Physics-Informed Neural Networks for Power Systems. Presented at the Best Paper Session of IEEE PES GM 2020. https://arxiv.org/pdf/1911.03737.pdf

Physics-Informed Neural Networks for Power Systems

“Original” Loss function

“Physics-Informed” term

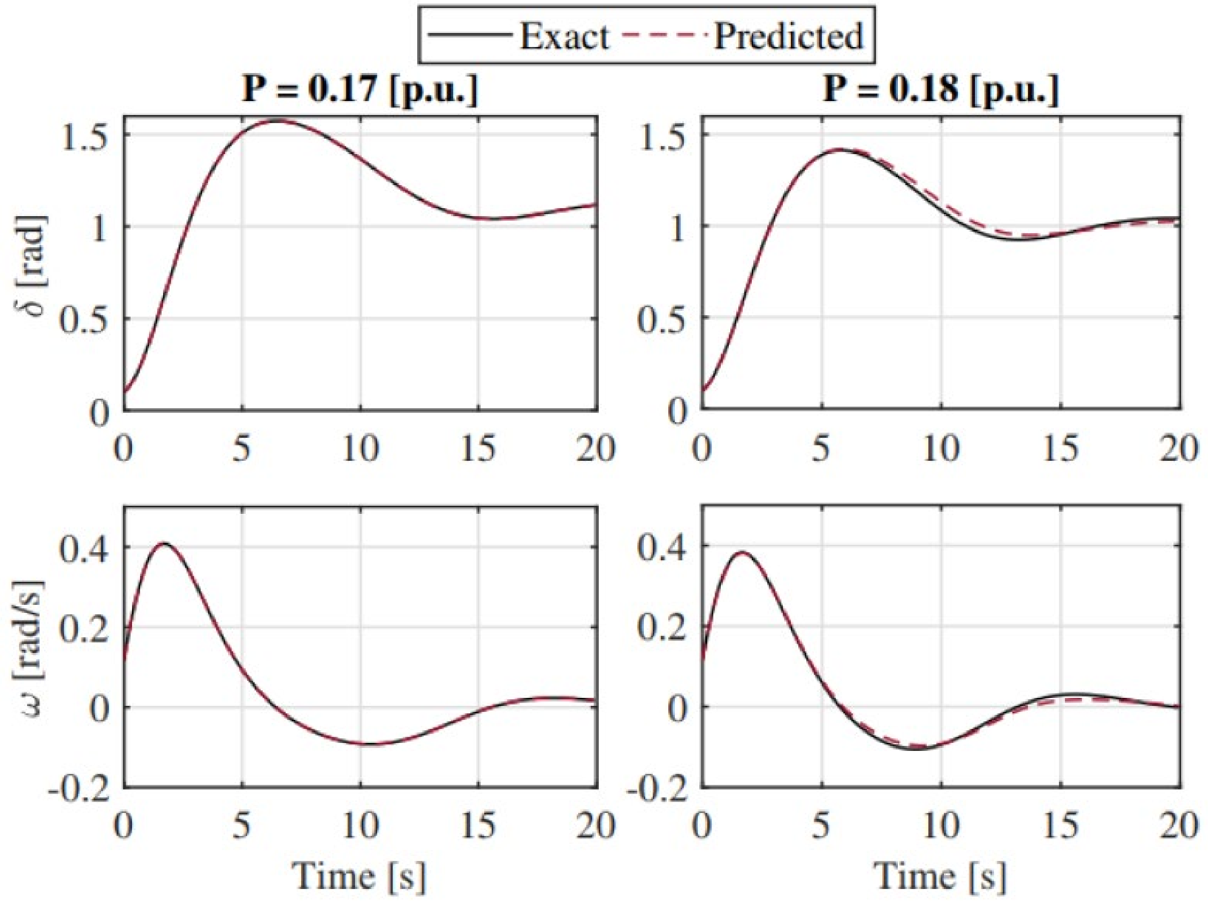
$$\min_{\mathbf{W},\mathbf{b}} \quad \frac{1}{|N_\delta|}\sum_{i\in N_\delta}|\hat{\delta}-\delta^i|^2 + \boxed{\frac{1}{|N_f|}\sum_{i\in N_f}|f(\hat{\delta})|^2} \tag{6a}$$

$$s.t. \qquad \hat{\delta} = NN(t, P_m, \mathbf{W}, \mathbf{b}) \tag{6b}$$

$$\dot{\hat{\delta}} = \frac{\partial\hat{\delta}}{\partial t}, \qquad \ddot{\hat{\delta}} = \frac{\partial\dot{\hat{\delta}}}{\partial t} \tag{6c}$$

$$\boxed{f(\hat{\delta}) = M\ddot{\hat{\delta}} + D\dot{\hat{\delta}} + A\sin\hat{\delta} - P_m} \tag{6d}$$

Swing equation



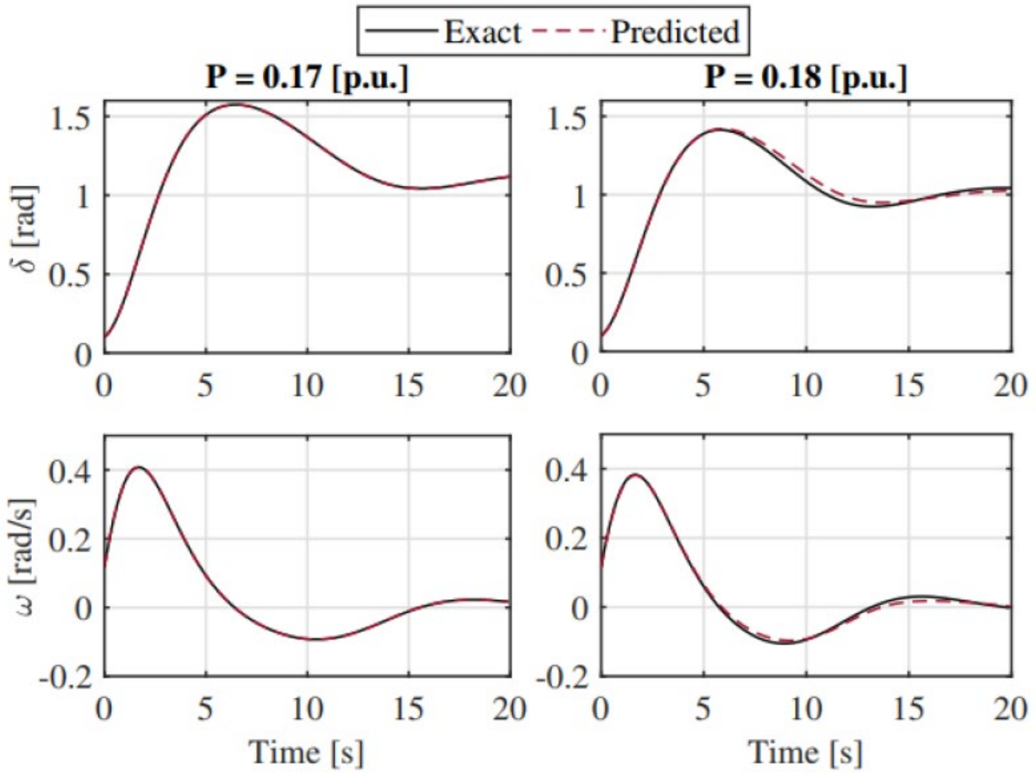
G. S. Misyris, A. Venzke, S. Chatzivasileiadis, Physics-Informed Neural Networks for Power Systems. Presented at the Best Paper Session of IEEE PES GM 2020. https://arxiv.org/pdf/1911.03737.pdf

Physics-Informed Neural Networks for Power Systems

- Physics-Informed Neural Networks (PINN) **can potentially replace** solvers for systems of differential-algebraic equations

- In our example: PINN 87 times faster than ODE solver

- Can **directly estimate** the rotor angle at **any** time instant



Code is available on GitHub: https://github.com/gmisy/Physics-Informed-Neural-Networks-for-Power-Systems/

G. S. Misyris, A. Venzke, S. Chatzivasileiadis, Physics-Informed Neural Networks for Power Systems. Presented at the Best Paper Session of IEEE PES GM 2020. https://arxiv.org/pdf/1911.03737.pdf

Wrap-up

- Neural network verification can remove barriers for neural network applications in power systems

- Physics Informed Neural Networks can take advantage of the rich information about existing power system models

- This talk:
  1. Verification of Classification Neural Networks for Power Systems
  2. Provable worst-case guarantees for Regression Neural Networks
     – Application in OPF (and probably any linear program)
  3. Physics-Informed Neural Networks for Power Systems

Ongoing work

Exploring a wide range of research directions

1. Contracting Neural-Newton Solver
   - Derive convergence guarantees for Neural Networks that can replace conventional Newton solvers [www.chatziva.com/publications/Chevalier_etal_CoNNS_ArXiv.pdf]

2. Physics-Informed Neural Networks for Fast Dynamic Security Assessment
   [soon on ArXiv, and more user-friendly code on PINNs on github!]

3. Physics-Informed Neural Networks for OPF with worst-case guarantees [soon on ArXiV]

4. Using neural networks to capture previously intractable optimization constraints
   [https://arxiv.org/pdf/2103.17004.pdf]

5. Accelerating MILPs: using Decision Trees to estimate the active set and drastically reduce the number of binary variable [https://arxiv.org/pdf/2010.06344.pdf]

and others…

Open Research Challenges

- **Tractability** for large neural networks
  - Up to now, we have verified NNs with 4 layers and 100 nodes at each layer (NN used for the 162-bus system)
  - We require weight sparsification, bound tightening, and ReLU pruning (remove binary variables) to maintain tractability
- **Retraining** is necessary to avoid adversarial examples
  - The **quality of the training database is crucial** for good performance!
- **Connect verification with ground truth assessment**
- **PINNs:** tractability for larger power systems

Thank you!

Spyros Chatzivasileiadis
Associate Professor, PhD
www.chatziva.com
spchatz@elektro.dtu.dk

A. Venzke, S. Chatzivasileiadis. Verification of Neural Network Behaviour: Formal Guarantees for Power System Applications. Accepted at IEEE Trans. on Smartgrid. 2020.
https://arxiv.org/pdf/1910.01624.pdf

A. Venzke, G. Qu, S. Low, S. Chatzivasileiadis, Learning Optimal Power Flow: Worst-case Guarantees for Neural Networks. **Best Student Paper Award** at IEEE SmartGridComm 2020.[.pdf | slides | video]

G. S. Misyris, A. Venzke, S. Chatzivasileiadis, Physics-Informed Neural Networks for Power Systems. Presented at the **Best Paper Session** of IEEE PES GM 2020. https://arxiv.org/pdf/1911.03737.pdf

J. Stiasny, G. S. Misyris, S. Chatzivasileiadis, Physics-Informed Neural Networks for Non-linear System Identification applied to Power System Dynamics. IEEE Powertech 2021.
https://arxiv.org/pdf/2004.04026.pdf

Some code available at:
www.chatziva.com/downloads.html

Current Work and Open Challenges

- PINNs for System Identification

- PINNs for quick simulation/estimation of converter dynamics
  - Can provide a quick estimate to a multi time-scale or co-simulation platform

- Tractability is an issue
  - Currently capturing all system dynamics for up to an 11-bus system with a single PINN
  - PINNs can also be applied for larger systems, but the computing effort (e.g. for training) will increase as well
  - Need to develop good approaches to maintain a lower computing effort while we scale up to bigger systems