

# Physics-Informed Neural Networks for Power Systems

Spyros Chatzivasileiadis  
Associate Professor, DTU

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

# Why even think of applying Machine Learning in Energy Systems?

## 1. Extremely fast

- computation within only a **few milliseconds**  
(100x – 1000x faster than conventional methods)

## 2. Good alternative if we do not have full knowledge of the actual model

- Handle **very complex systems**
- **Infer** from incomplete data

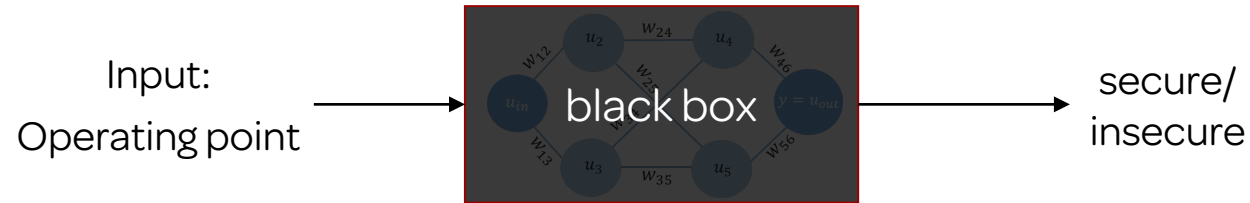
**Energy systems:** extremely large and extremely complex  
Electric power grid: “the **largest machine** humans ever built”

**But:**

1. **Would an Operator ever trust AI in the Control Room?**
2. **Why use ML when engineers and scientists have spent the past 100 years developing good models for power systems?**

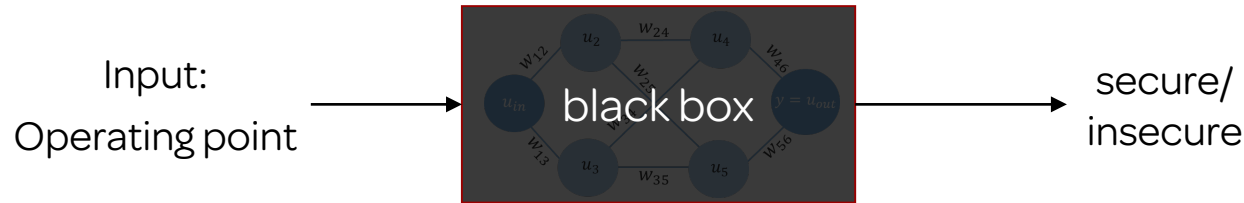


# ML Barriers for Power systems

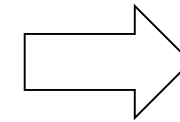


1. Why would we use a “**black box**” to decide about a **safety-critical application**?
2. **Accuracy is a purely statistical** performance metric.  
Who guarantees that the Neural Network can handle well previously unseen operating points?
3. Why would we only depend on **discrete and incomplete data**, when we have developed **detailed physical models** over the past 100 years?

# 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 only on **discrete and incomplete data**, when we have developed **detailed physical models** over the past 100 years?



1. Integrate the power system models in **NN training**  
 → potential to replace conventional solvers?

**2. Integrate the physical models in Neural Network verification :**  
 worst-case guarantees for the NN performance!

## 1. **Physics-Informed Neural Networks for Power System Dynamics**

- Regression neural networks → estimation of numerical values such as rotor angle and frequency
- Work inspired by Raissi et al\* who applied it on physics problems
- There exist a few recent works that use similar principles and apply PINNs on Optimal Power Flow problems (see Kekatos et al, K. Baker et al, Henttenryck et al, among others)

## 2. **Worst-case Guarantees for Neural Networks Learning OPF**

- Introducing physical models to NN verification
- Formulating MILPs that provide worst-case guarantees

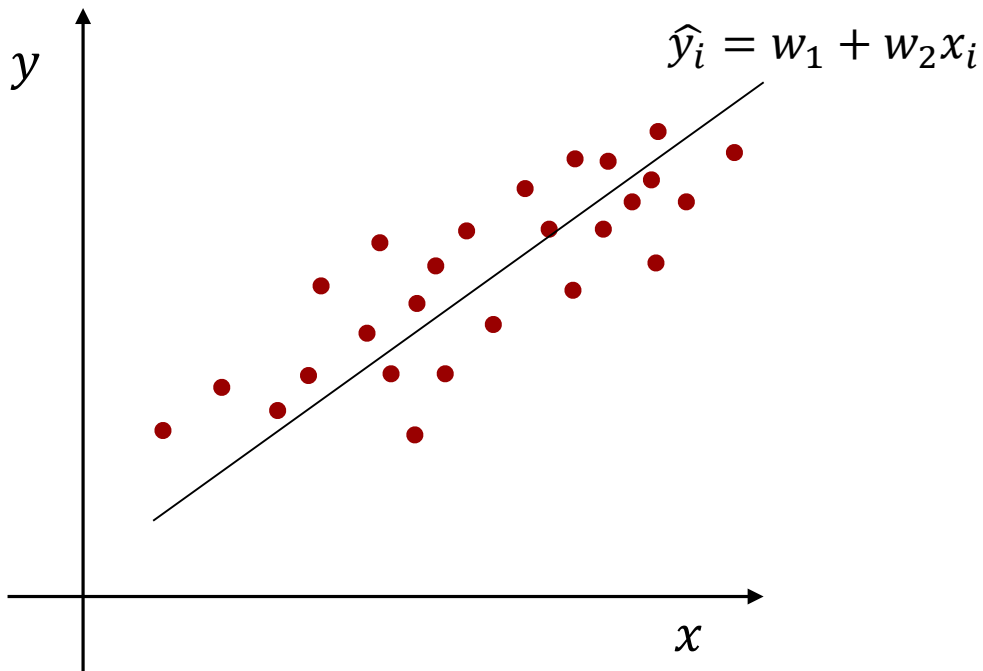
\*M. Raissi, P. Perdikaris, and G. Karniadakis, Physics-Informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations", Journal of Computational Physics, vol.378, pp. 686-707, 2019

# Physics-Informed Neural Networks for Power Systems

# 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**

$$\begin{aligned} & \min_{w_1, w_2} \|y_i - \hat{y}_i\| \\ \text{s.t.} & \hat{y}_i = w_1 + w_2 x_i \quad \forall i \end{aligned}$$

**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

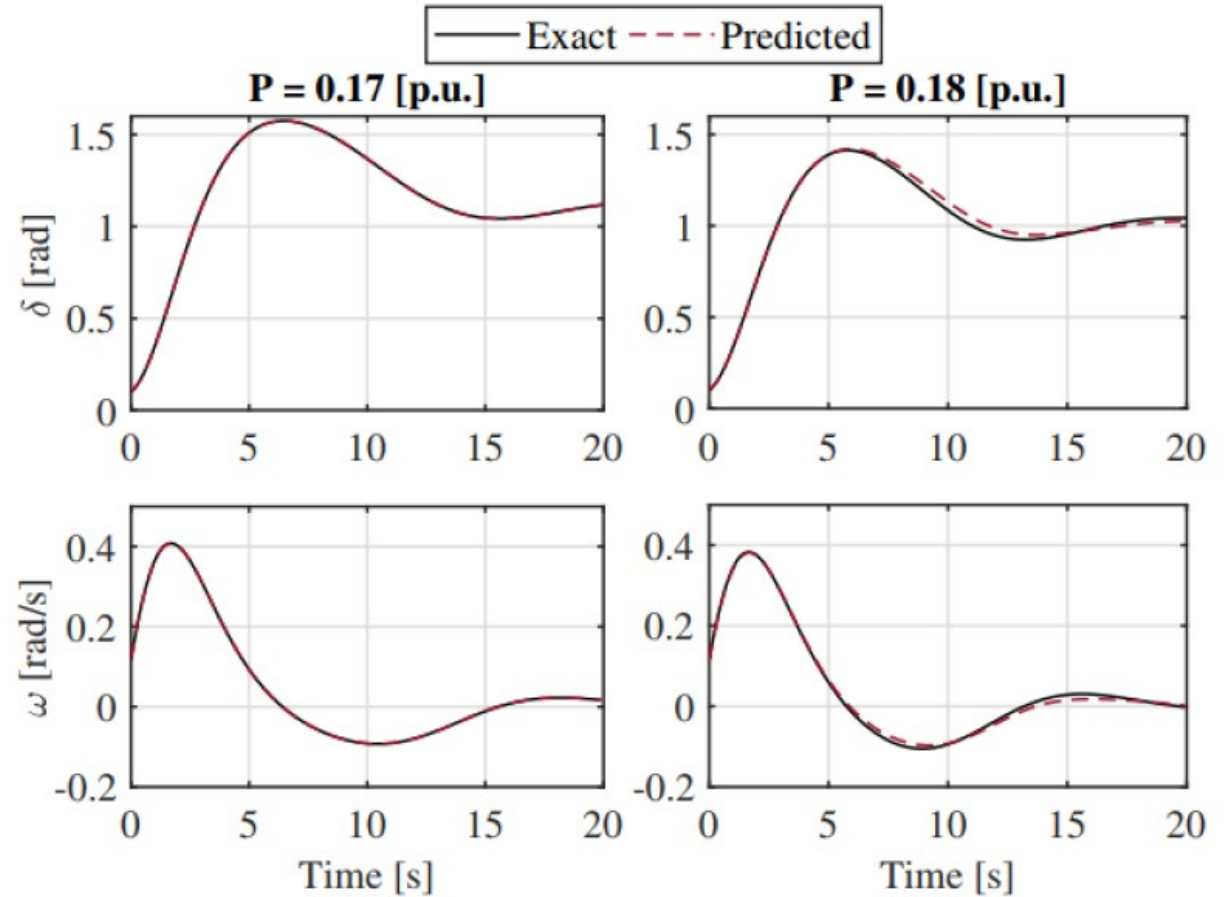
$$\min_{\mathbf{W}, \mathbf{b}} \frac{1}{|N_\delta|} \sum_{i \in N_\delta} |\hat{\delta} - \delta^i|^2 + \frac{1}{|N_f|} \sum_{i \in N_f} |f(\hat{\delta})|^2 \quad (6a)$$

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

$$\dot{\hat{\delta}} = \frac{\partial \hat{\delta}}{\partial t}, \quad \ddot{\hat{\delta}} = \frac{\partial^2 \hat{\delta}}{\partial t^2} \quad (6c)$$

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

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

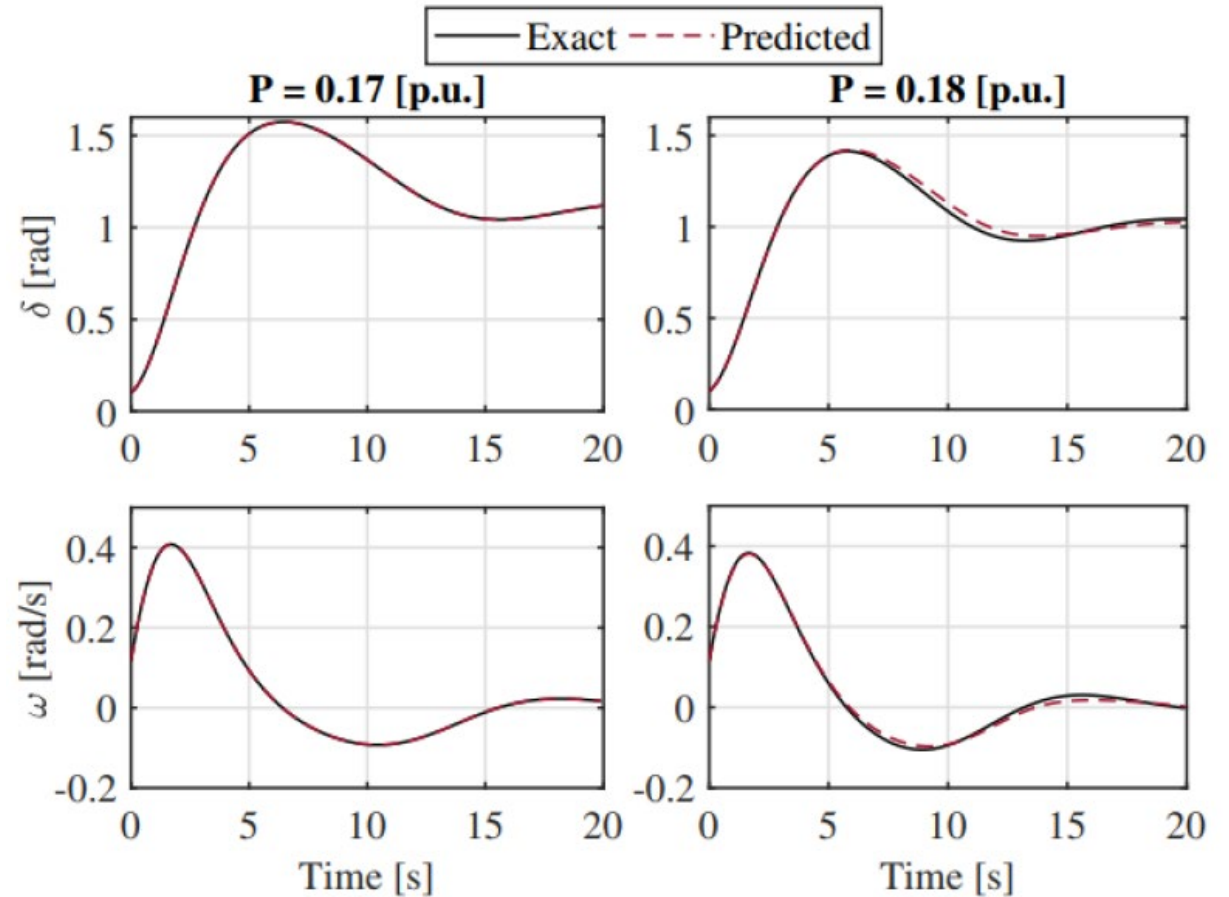
$$\min_{\mathbf{W}, \mathbf{b}} \frac{1}{|N_\delta|} \sum_{i \in N_\delta} |\hat{\delta} - \delta^i|^2 + \frac{1}{|N_f|} \sum_{i \in N_f} |f(\hat{\delta})|^2 \quad (6a)$$

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

$$\dot{\hat{\delta}} = \frac{\partial \hat{\delta}}{\partial t}, \quad \ddot{\hat{\delta}} = \frac{\partial^2 \hat{\delta}}{\partial t^2} \quad (6c)$$

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

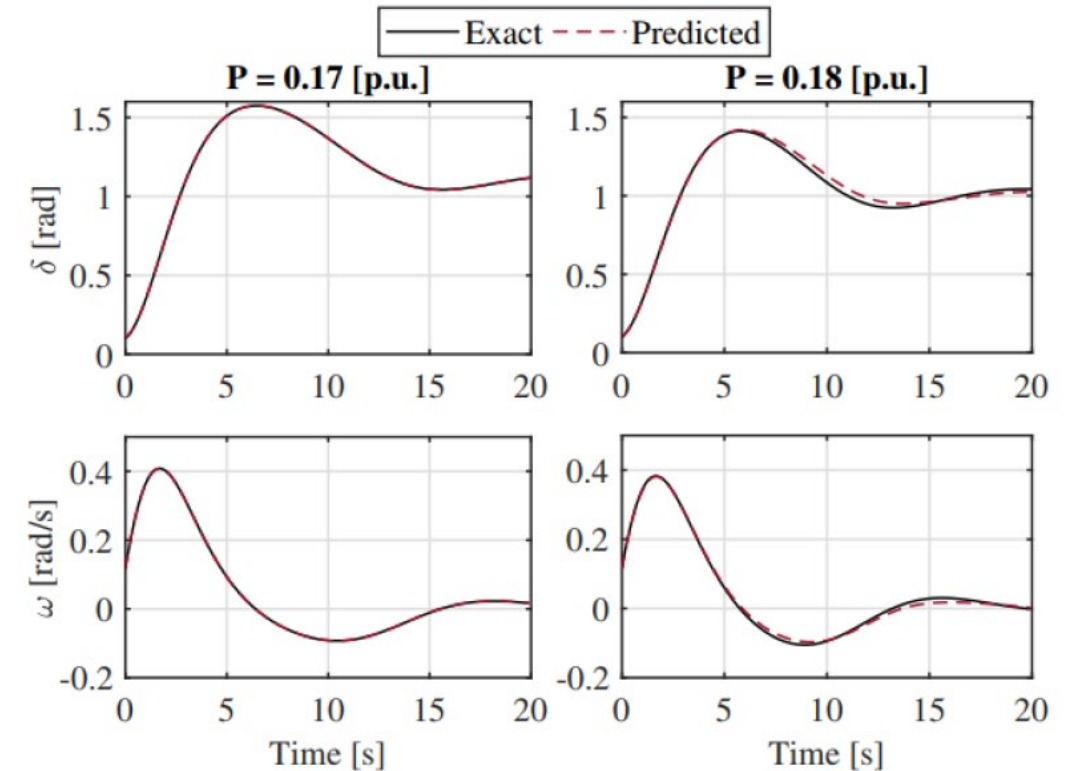
Swing equation



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>

# 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

# 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>

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

$$\min_{\mathbf{p}_g, \boldsymbol{\theta}} \mathbf{c}^T \mathbf{p}_g$$

Minimizes generation cost

$$\text{s.t. } \mathbf{M}_g \mathbf{p}_g - \mathbf{M}_d \mathbf{p}_d = \mathbf{B}_{\text{bus}} \boldsymbol{\theta}$$

Nodal power balance

$$-\mathbf{p}_{\text{line}}^{\max} \leq \mathbf{B}_{\text{line}} \boldsymbol{\theta} \leq \mathbf{p}_{\text{line}}^{\max}$$

Transmission line limits

$$\mathbf{p}_g^{\min} \leq \mathbf{p}_g \leq \mathbf{p}_g^{\max}$$

Generator limits

# 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**

$$\min_{\mathbf{p}_g, \boldsymbol{\theta}} \mathbf{c}^T \mathbf{p}_g$$

Minimizes generation cost

$$\text{s.t. } \mathbf{M}_g \mathbf{p}_g - \mathbf{M}_d \mathbf{p}_d = \mathbf{B}_{\text{bus}} \boldsymbol{\theta}$$

Nodal power balance

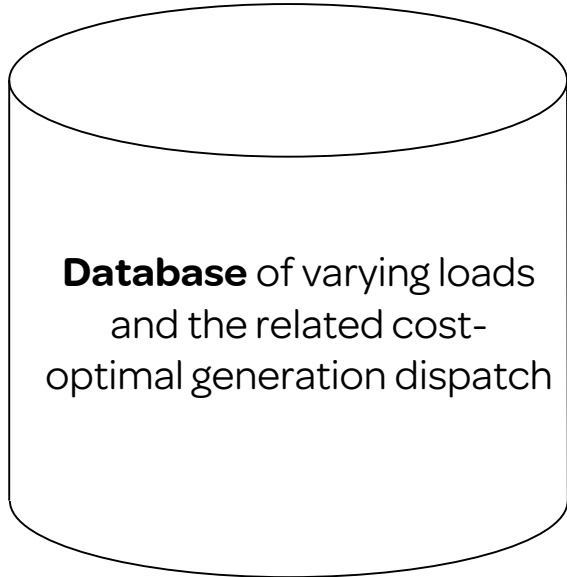
$$-\mathbf{p}_{\text{line}}^{\max} \leq \mathbf{B}_{\text{line}} \boldsymbol{\theta} \leq \mathbf{p}_{\text{line}}^{\max}$$

Transmission line limits

$$\mathbf{p}_g^{\min} \leq \mathbf{p}_g \leq \mathbf{p}_g^{\max}$$

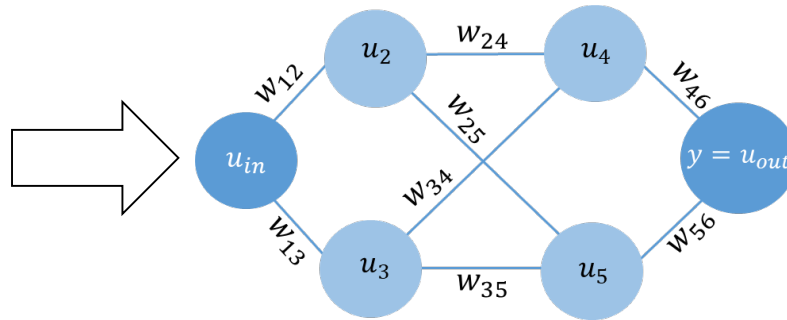
Generator limits

# Guiding Application for Regression NN: Learning OPF



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

## Approaches proposed up to now

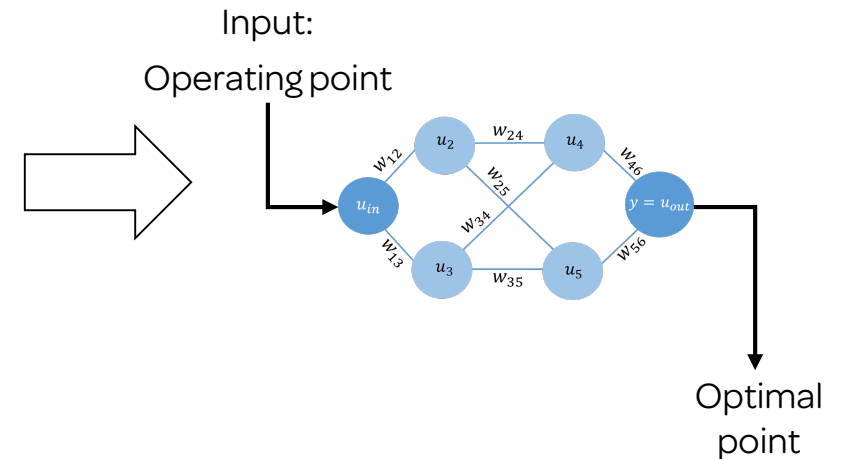


2. Train a neural network

3. Test the neural network

4. Is accuracy high enough?

## 5. Use the NN



## NN Output:

Optimal generation dispatch

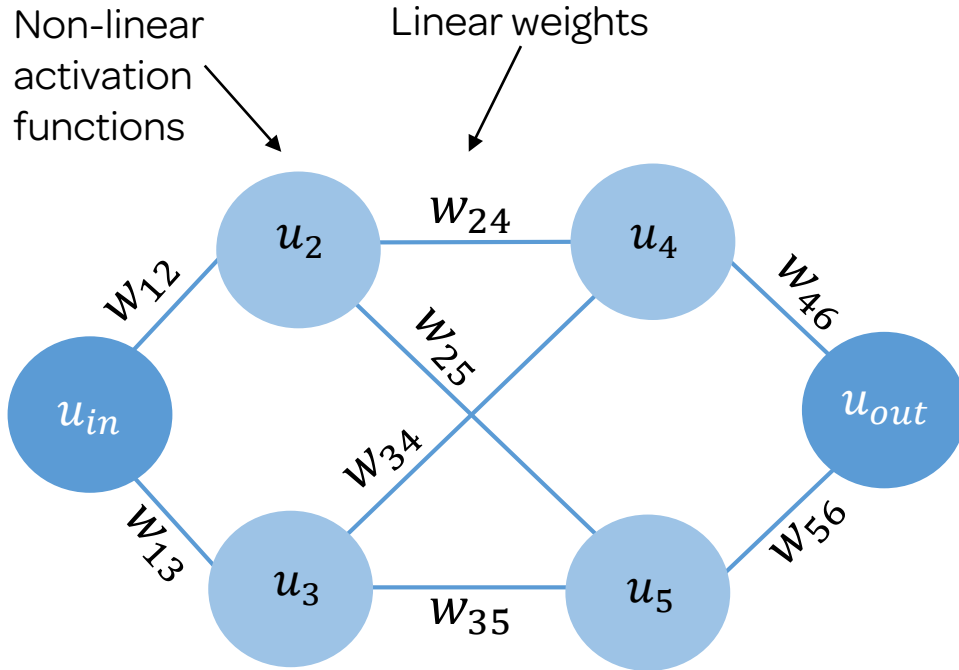
**Extremely fast:**  
up to 100x faster



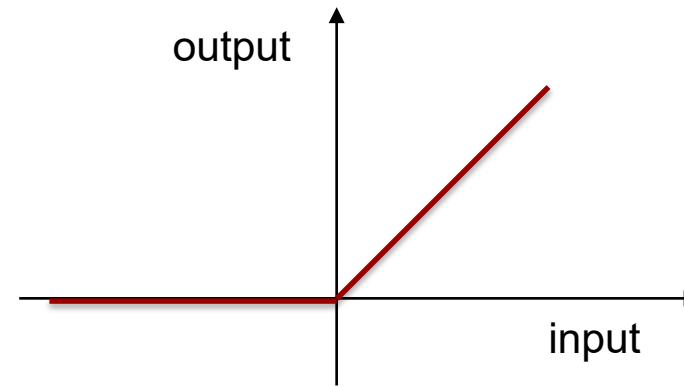
# Neural Network Verification for Regression NNs: HOW?

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. Include the physical equations
3. Formulate an **optimization** problem (MILP) and solve it → certificate for NN behavior

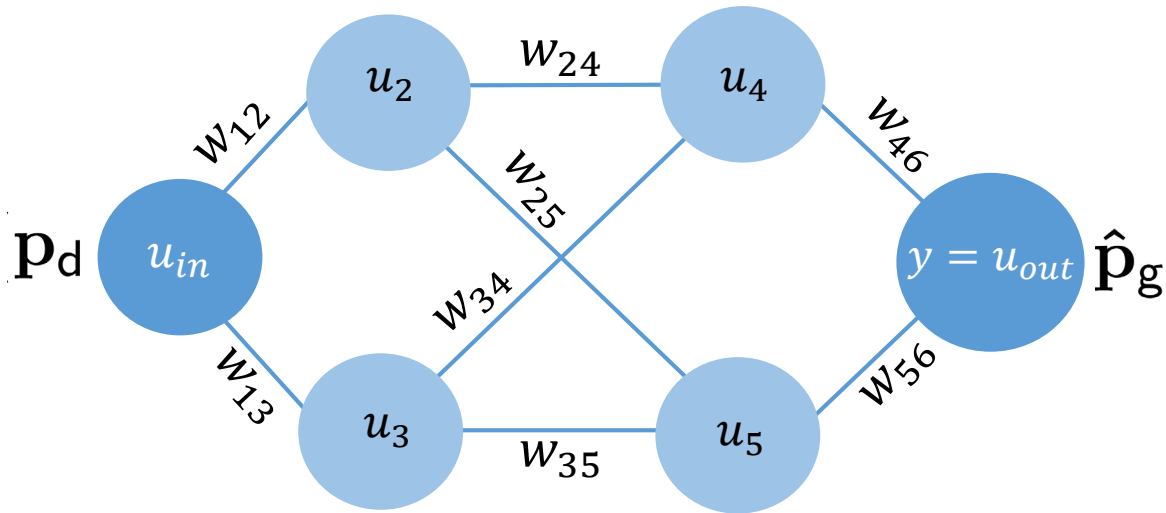
# From Neural Networks to Mixed-Integer Linear Programming



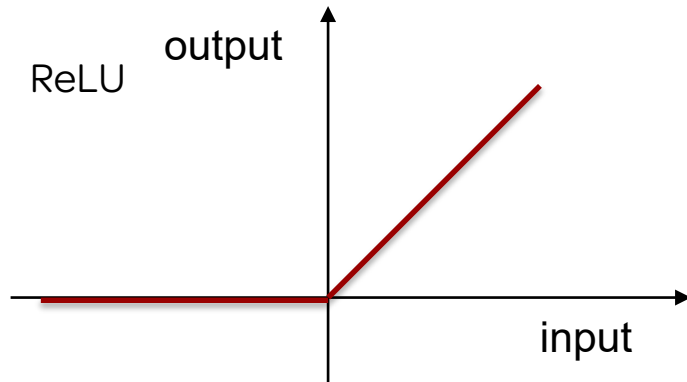
- Most usual activation function: ReLU
- ReLU: Rectifier Linear Unit



# From Neural Networks to Mixed-Integer Linear Programming



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



$$\text{MILP} \rightarrow \hat{\mathbf{p}}_g = NN(\mathbf{p}_d)$$

# Part I: Maximum limit-violations

1. Maximum violation of generator limits

$$\nu_g = \max(\hat{\mathbf{p}}_g - \mathbf{p}_g^{\max}, \mathbf{p}_g^{\min} - \hat{\mathbf{p}}_g, \mathbf{0})$$

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

Example:

$$0.6 \mathbf{p}_d^{\max} \leq \mathbf{p}_d \leq 1.0 \mathbf{p}_d^{\max}$$

# Part I: Maximum limit-violations

1. Maximum violation of generator limits

$$\nu_g = \max(\hat{\mathbf{p}}_g - \mathbf{p}_g^{\max}, \mathbf{p}_g^{\min} - \hat{\mathbf{p}}_g, \mathbf{0})$$

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

Example:

$$0.6 \mathbf{p}_d^{\max} \leq \mathbf{p}_d \leq 1.0 \mathbf{p}_d^{\max}$$

2. Maximum violation of line limits

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

Line flow equations for DC-OPF based on PTDFs

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

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	$\nu_g$ (MW)	$\nu_{line}$ (MW)	$\nu_g$ (MW)	$\nu_{line}$ (MW)
<i>case9</i>				
<i>case30</i>				
<i>case39</i>				
<i>case57</i>				
<i>case118</i>				
<i>case162</i>				
<i>case300</i>				

$\nu_g$  Maximum violation of generator limits

$\nu_{line}$  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**

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

$\nu_g$  Maximum violation of generator limits

$\nu_{line}$  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		Exact worst-case guarantee	
	$\nu_g$ (MW)	$\nu_{line}$ (MW)	$\nu_g$ (MW)	$\nu_{line}$ (MW)
<i>case9</i>	2.5	1.8	2.8	1.9
<i>case30</i>	1.7	0.6	3.6	3.1
<i>case39</i>	51.9	37.2	270.6	120.0
<i>case57</i>	4.2	0.0	23.7	0.0
<i>case118</i>	149.4	15.6	997.8	510.8
<i>case162</i>	228.0	180.0	1563.3	974.1
<i>case300</i>	474.5	692.7	3658.5	3449.3

$\nu_g$  Maximum violation of generator limits

$\nu_{line}$  Maximum violation of line limits

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



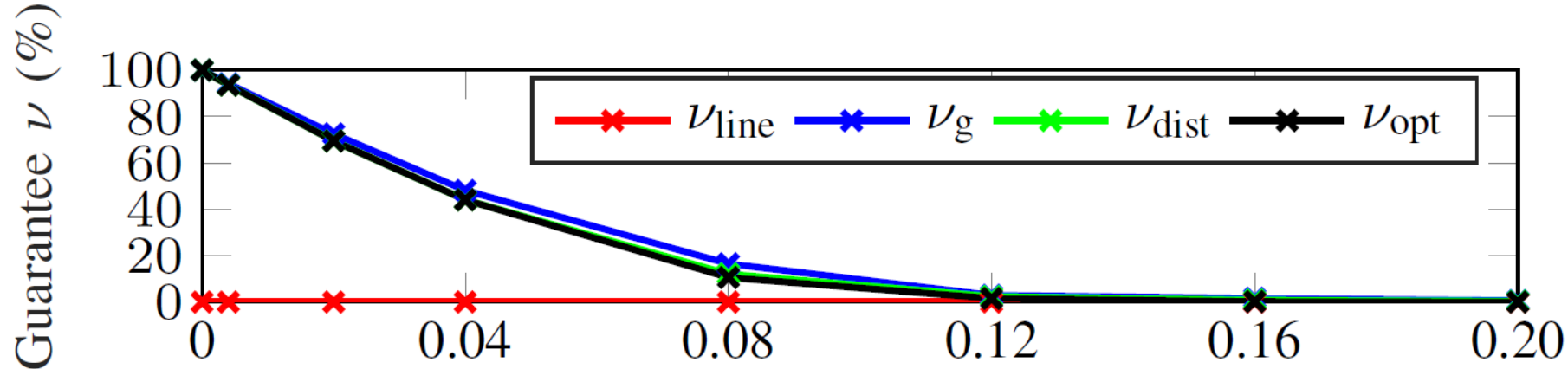


## How can we reduce the worst-case violations?

- 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  $\delta$  (-)

- Input domain used for training

$$0.6 \mathbf{p}_d^{\max} \leq \mathbf{p}_d \leq 1.0 \mathbf{p}_d^{\max}$$

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

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

Example: If  $\delta = 0.1$  then  $0.7 \mathbf{p}_d^{\max} \leq \mathbf{p}_d \leq 0.9 \mathbf{p}_d^{\max}$

- **Physics-Informed Neural Networks for Power System Dynamics**
  - Reduce dependence on discrete or incomplete data
  - Drastically reduce computing time
  - Potential to replace conventional differential-algebraic solvers?  
(maybe for use cases where computing time is more important than high accuracy?)
  
- **Provable worst-case guarantees for Regression Neural Networks**
  - Combination of Neural Network verification with underlying physical models
  - Application in OPF (and probably any linear program)

# Thank you!



Spyros Chatzivasileiadis  
Associate Professor, PhD  
[www.chatziva.com](http://www.chatziva.com)  
spchatz@elektro.dtu.dk

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>

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](#)]

Some code available at:

[www.chatziva.com/downloads.html](http://www.chatziva.com/downloads.html)