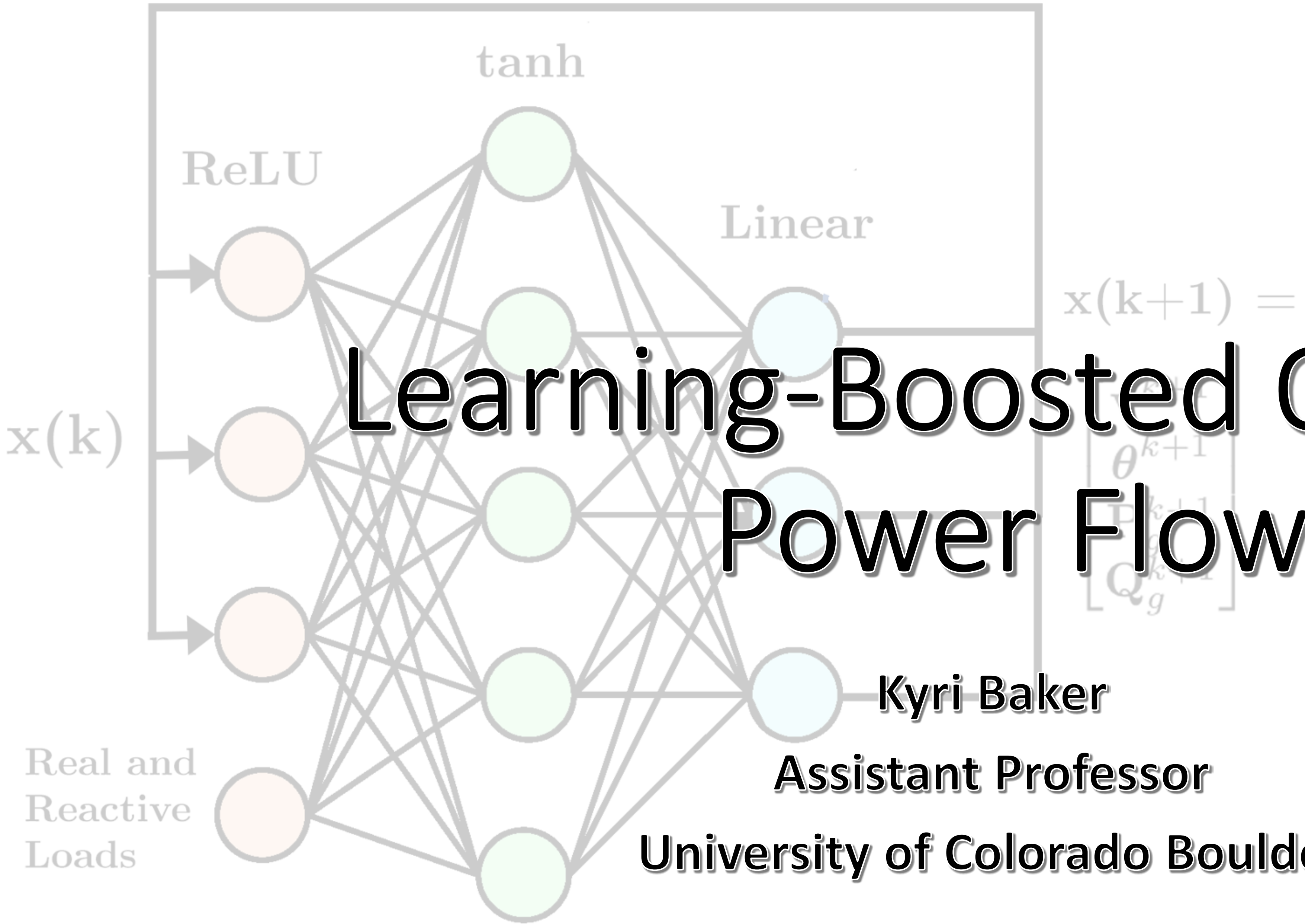


Learning-Boosted Optimal Power Flow

Kyri Baker

Assistant Professor

University of Colorado Boulder

Talk Outline

- Machine learning for optimization: The revolution
- Warm-starting OPF with Machine Learning
- Obtaining approximate OPF solutions extremely quickly
- Obtaining feasible OPF solutions with Machine Learning?
- Future Directions

Main Idea: Optimizing is hard.

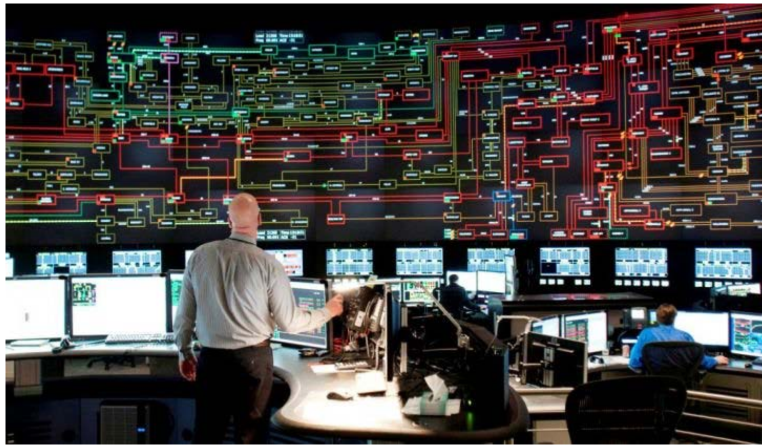
→ Can we obtain the solution to an optimization problem without actually solving one?

Can my smart thermostat solve a complex optimization problem in real time?

Main Idea: Optimizing is hard.

→ Can we obtain the solution to an optimization problem without actually solving one?

Can my smart thermostat solve a complex optimization problem in real time?



Can grid operators pursue optimal generation settings in real time?

→ *Without relying on real-time heuristics like AGC?*

→ *While hedging against suboptimality[0]?*

[0] **Billions** of dollars annually are lost due to OPF suboptimalities. M. Cain, R. P. O'Neill, and A. Castillo, "History of optimal power flow and formulations," *FERC Technical Report*, Aug. 2013.

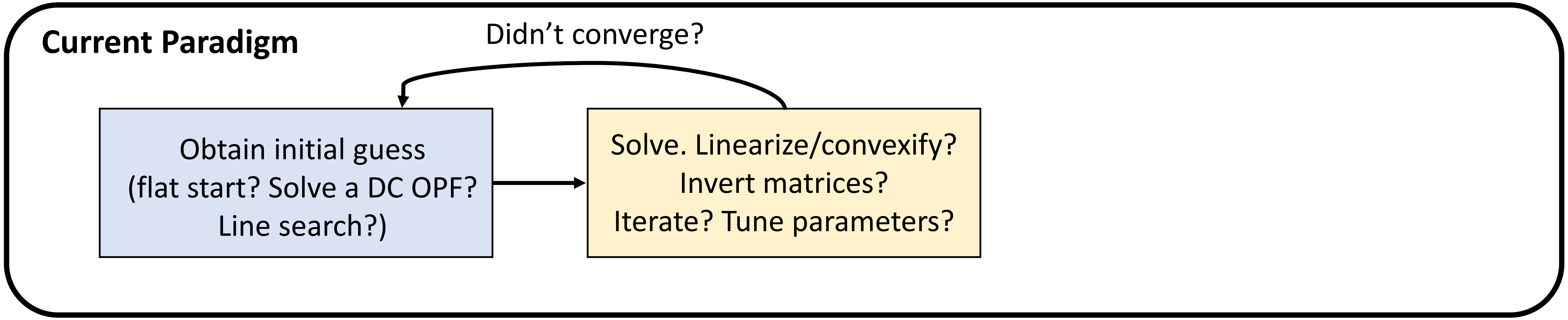
How does it work?

→ Can we obtain the solution to an optimization problem without actually solving one?

Current Paradigm
Obtain initial guess
(flat start? Solve a DC OPF?
Line search?)

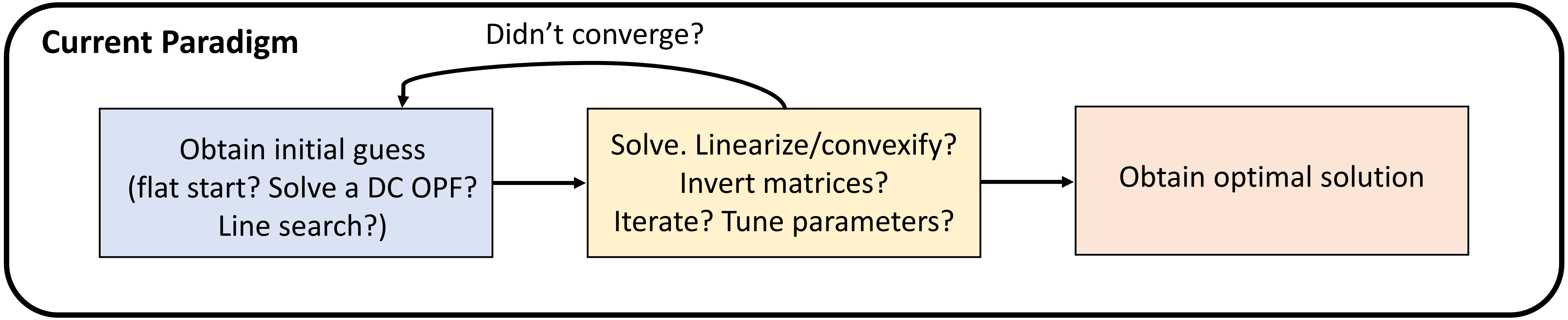
How does it work?

→ Can we obtain the solution to an optimization problem without actually solving one?


Current Paradigm
Didn't converge?
Obtain initial guess (flat start? Solve a DC OPF? Line search?)
Solve. Linearize/convexify? Invert matrices? Iterate? Tune parameters?

How does it work?

→ Can we obtain the solution to an optimization problem without actually solving one?



Current Paradigm
Didn't converge?
Obtain initial guess (flat start? Solve a DC OPF? Line search?)
Solve. Linearize/convexify? Invert matrices? Iterate? Tune parameters?
Obtain optimal solution

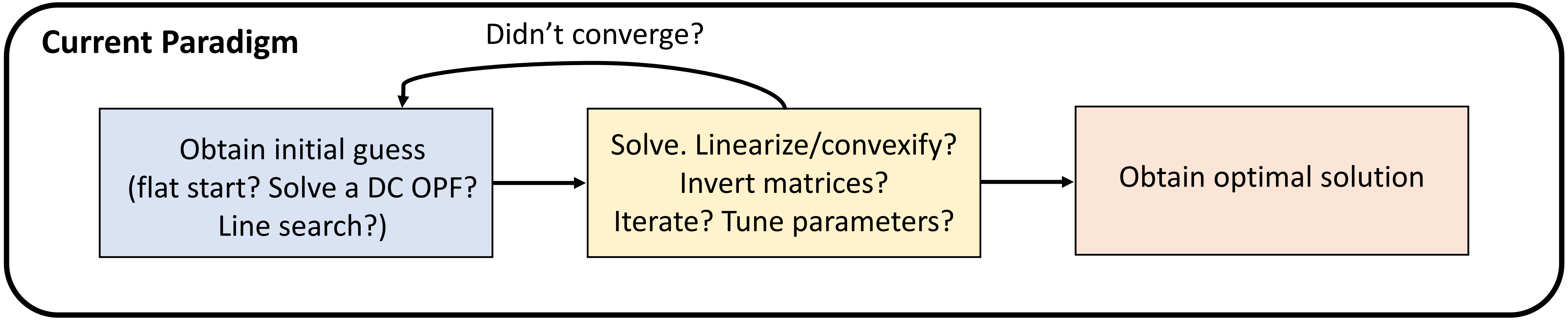
Hmm, this approach is done hundreds of times daily for OPF…can we use this information to help us more quickly find future OPF solutions?

"the volume of data being generated in the power sector has grown tremendously…. the majority of data is either not logged, or they are overwritten very quickly[1]"

[1] H. Akhavan-Hejazi and H. Mohsenian-Rad, "Power systems big data analytics: An assessment of paradigm shift barriers and prospects," *Energy Reports*, 2018.

How does it work?

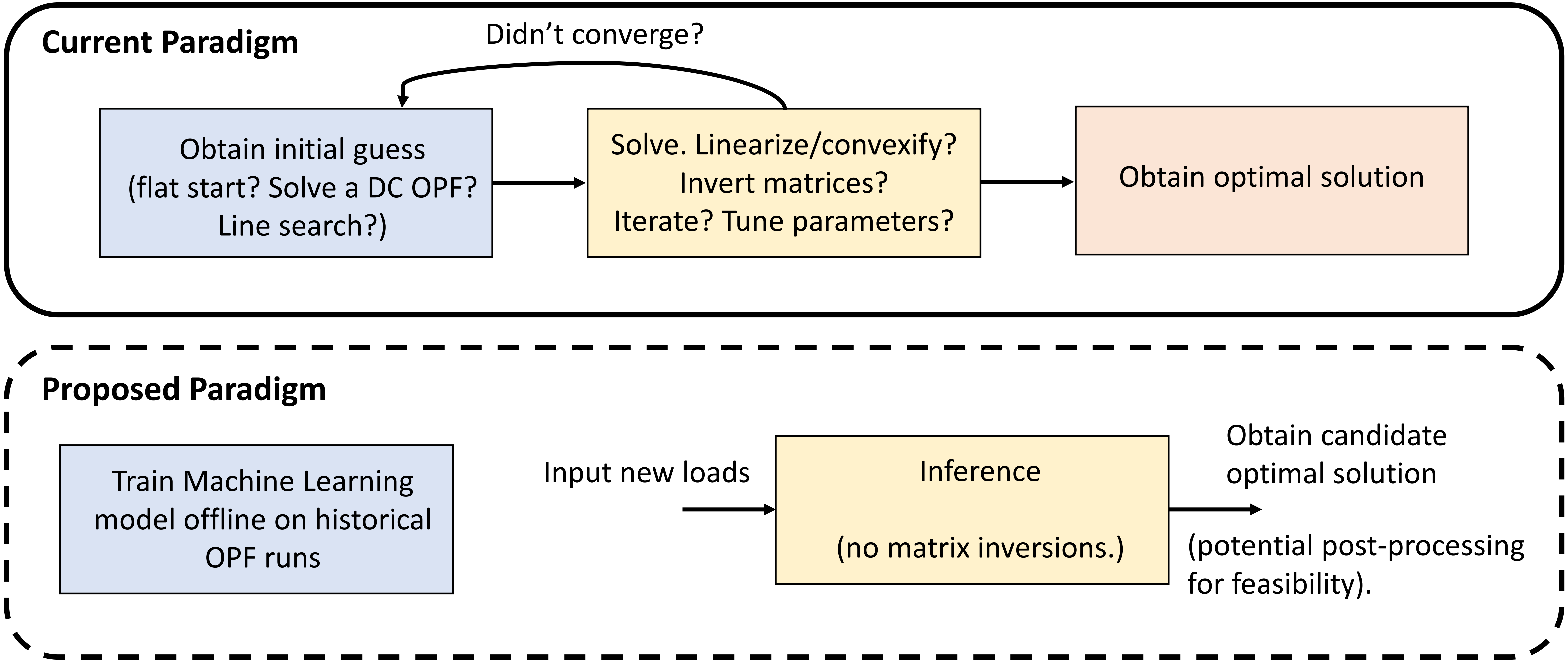
→ Can we obtain the solution to an optimization problem without actually solving one?


Current Paradigm
Didn't converge?
Obtain initial guess (flat start? Solve a DC OPF? Line search?)
Solve. Linearize/convexify? Invert matrices? Iterate? Tune parameters?
Obtain optimal solution

Proposed Paradigm
Train Machine Learning model offline on historical OPF runs

How does it work?

→ Can we obtain the solution to an optimization problem without actually solving one?


Current Paradigm
Didn't converge?
Obtain initial guess (flat start? Solve a DC OPF? Line search?)
Solve. Linearize/convexify? Invert matrices? Iterate? Tune parameters?
Obtain optimal solution
Proposed Paradigm
Train Machine Learning model offline on historical OPF runs
Input new loads
Inference
(no matrix inversions.)
Obtain candidate optimal solution
(potential post-processing for feasibility).

Does this actually work?

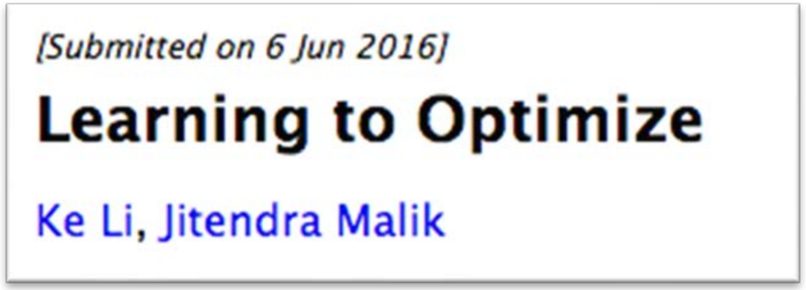
July 2020: A. Velloso and P. Van Hentenryck use deep learning to obtain solutions with near-negligible feasibility and optimality gaps (under 0.1%) for Security Constrained DC OPF on a 1,888 bus network on an average of less than two seconds[2].

Sept. 2019: A. Zamzam and K. Baker use deep learning to obtain **feasible** AC OPF solutions with negligible optimality gaps **6-20x** faster than a state of the art solver[3].

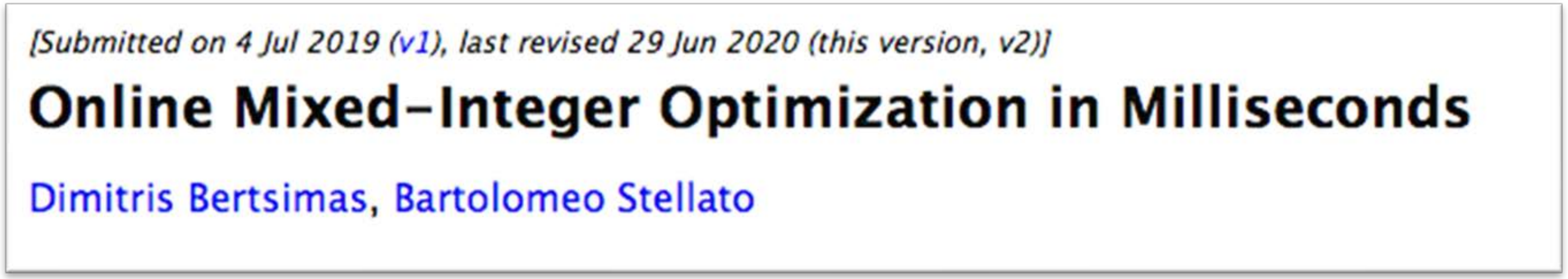
[2] A. Velloso and P. Van Hentenryck, "Combining Deep Learning and Optimization for Security-Constrained Optimal Power Flow," https://arxiv.org/abs/2007.07002, July 2020.

[3] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," https://arxiv.org/abs/2861719, Sept. 2019

Outside of power and energy:


[Submitted on 6 Jun 2016]
Learning to Optimize
Ke Li, Jitendra Malik

→ Uses reinforcement learning to design optimization algorithms for unconstrained problems


[Submitted on 4 Jul 2019 (v1), last revised 29 Jun 2020 (this version, v2)]
Online Mixed-Integer Optimization in Milliseconds
Dimitris Bertsimas, Bartolomeo Stellato

→ Uses neural networks to solve mixed integer quadratic programs in milliseconds (solves a MIQP in **less time than it takes to perform a single matrix factorization**).

Why is it powerful?

→ We can convexify or linearize the hard equations (e.g. the AC power flow equations) to solve these problems quickly, but convexifying generally makes us lose information

→ Neural networks preserve nonconvex, complicated relationships between variables

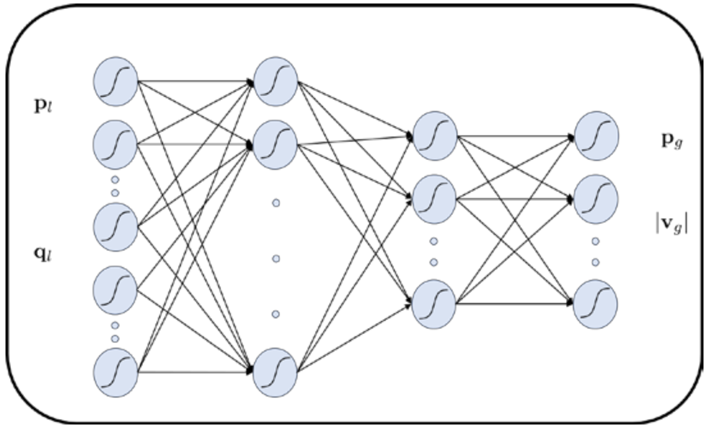
Why is it powerful?

→ We can convexify or linearize the hard equations (e.g. the AC power flow equations) to solve these problems quickly, but convexifying generally makes us lose information

→ Neural networks preserve nonconvex, complicated relationships between variables

→ Inference (the process of making a prediction) mostly involves applying functions, multiplying, and adding

→ It's an approximation, but it can be a damn good one



Did you know?

Many grid operators use a linear approximation every day to solve optimal power flow – the DC approximation. However, solutions from the DC approximation are **never AC feasible.** Meaning, the intersection of the feasible region of DC OPF and the AC OPF is empty[4]!

[4] K. Baker, "Solutions of DC OPF are Never AC Feasible: Learning to Lower the Gap," https://arxiv.org/abs/1912.00319, Apr 2020

Opinions from your presenter

ML for OPF’s benefits decline / do not make sense for most small networks, or most standard DC OPF formulations. These problems can already be solved to optimality with free software in seconds[5].

If used in a real power system, would likely be a combination of learning and post-processing (if a nonconvex problem) or physics-informed/embedded constraints (if a convex problem) to ensure physical constraints hold.

Level of accuracy and speed for the level of optimality/feasibility sacrifice may in general be worth it. Worst-case guarantees have been defined[6]

[5] J. Kardos, D. Kourounis, O. Schenk, and R. Zimmerman, “Complete results for a numerical evaluation of interior point solvers for large-scale optimal power flow problems,” https://arxiv.org/abs/1807.03964v3, July 2018.

[6] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, “Learning Optimal Power Flow: Worst-Case Guarantees for Neural Networks,” https://arxiv.org/abs/2006.11029, Jun 2020.

Talk Outline

- Machine learning for optimization: The revolution
- Warm-starting OPF with Machine Learning
- Obtaining approximate OPF solutions extremely quickly
- Obtaining feasible OPF solutions with Machine Learning?
- Future Directions

Idea: How do we obtain feasibility?

→ Maybe instead of trying to make an ML model directly predict an optimal solution, we use it to predict a **close** solution, then use this as a warm start to an AC OPF solver[7,8,9]?

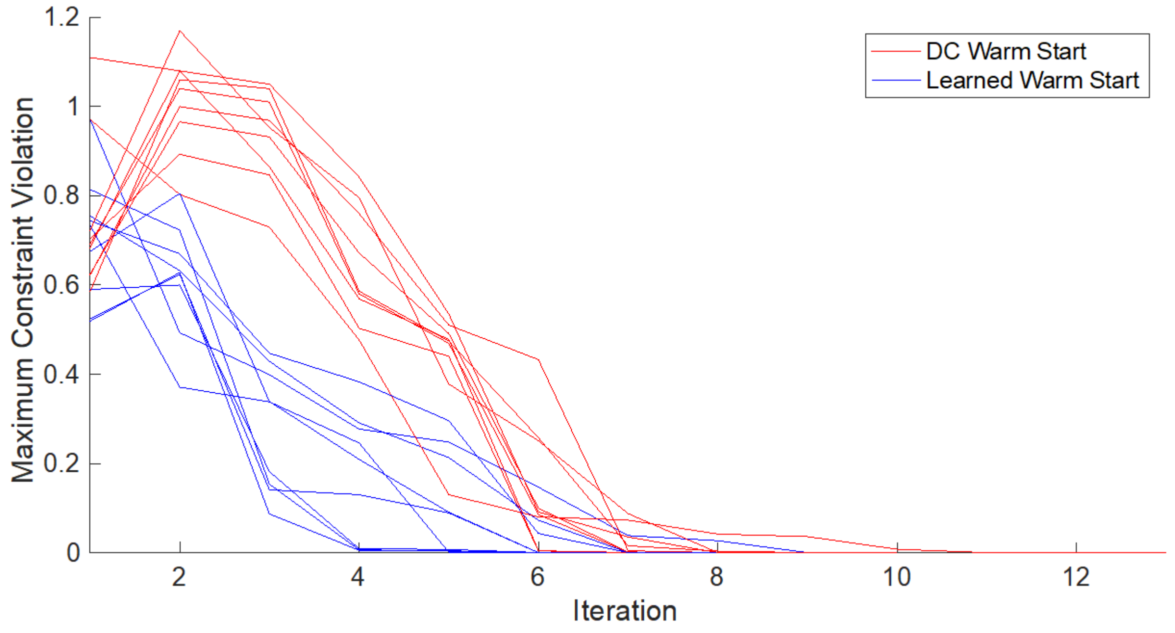
[7] K. Baker, "Learning warm-start points for AC optimal power flow," *IEEE Machine Learning for Signal Proc. Conf.*, Oct. 2019.
[8] F. Diehl, "Warm-starting AC optimal power flow with graph neural networks," *Neural Information Processing Systems (NeurIPS 2019)*, Dec 2019.
[9] L. Chen and J. Tate, "Hot-Starting the Ac Power Flow with Convolutional Neural Networks," https://arxiv.org/abs/2004.09342, Apr. 2020

Idea: How do we obtain feasibility?

→ Maybe instead of trying to make an ML model directly predict an optimal solution, we use it to predict a **close** solution, then use this as a warm start to an AC OPF solver[7,8,9]?

Directly trained a Random Forest on a dataset comprised of AC OPF solutions.

Input: System loads (real and reactive)
Output: Optimal generation and voltages



[7] K. Baker, "Learning warm-start points for AC optimal power flow," *IEEE Machine Learning for Signal Proc. Conf.*, Oct. 2019.
[8] F. Diehl, "Warm-starting AC optimal power flow with graph neural networks," *Neural Information Processing Systems (NeurIPS 2019)*, Dec 2019.
[9] L. Chen and J. Tate, "Hot-Starting the Ac Power Flow with Convolutional Neural Networks," https://arxiv.org/abs/2004.09342, Apr. 2020

How well does it do over a DC warm start?

Used MIPS MATPOWER solver and changed the initial point from flat start to DC OPF start to Learned-start

Ran 400 AC OPFs with randomly generated loading profiles

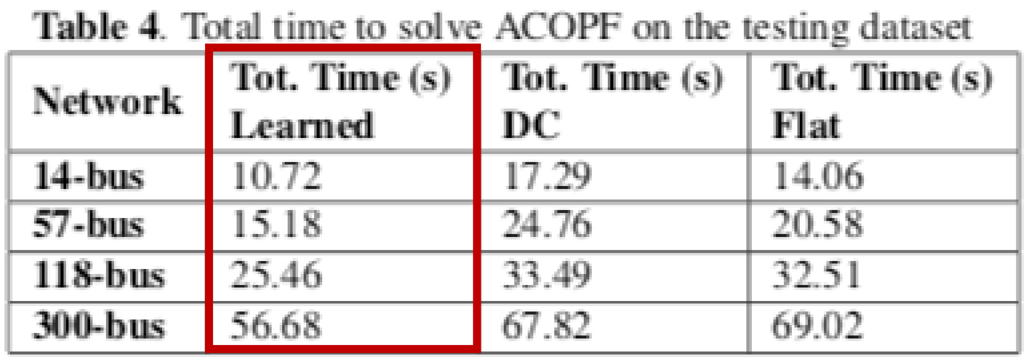


Table 4. Total time to solve ACOPF on the testing dataset

Network	Tot. Time (s) Learned	Tot. Time (s) DC	Tot. Time (s) Flat
14-bus	10.72	17.29	14.06
57-bus	15.18	24.76	20.58
118-bus	25.46	33.49	32.51
300-bus	56.68	67.82	69.02

How well does it do over a DC warm start?

Used MIPS MATPOWER solver and changed the initial point from flat start to DC OPF start to Learned-start

Ran 400 AC OPFs with randomly generated loading profiles

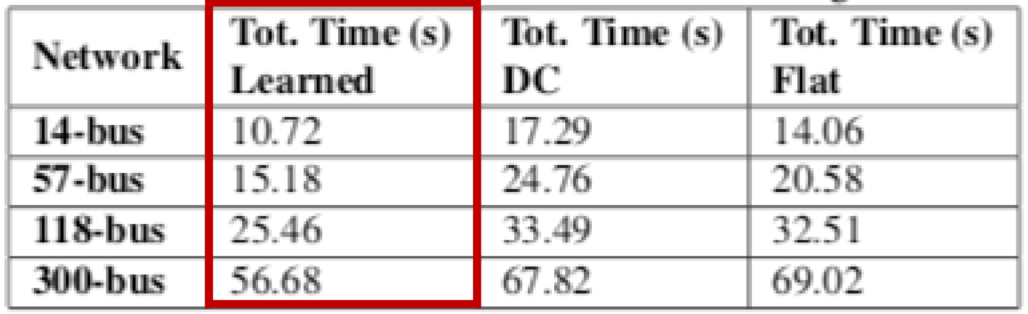


Table 4. Total time to solve ACOPF on the testing dataset

Network	Tot. Time (s) Learned	Tot. Time (s) DC	Tot. Time (s) Flat
14-bus	10.72	17.29	14.06
57-bus	15.18	24.76	20.58
118-bus	25.46	33.49	32.51
300-bus	56.68	67.82	69.02

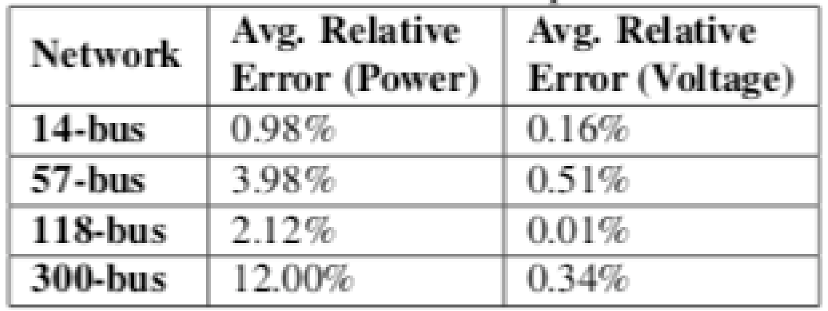


Table 2. Relative errors of predictions

Network	Avg. Relative Error (Power)	Avg. Relative Error (Voltage)
14-bus	0.98%	0.16%
57-bus	3.98%	0.51%
118-bus	2.12%	0.01%
300-bus	12.00%	0.34%

Random forest actually didn’t do that bad in predicting the optimal solution, making it a good starting point for an iterative solver. But neural networks can do even better…

Talk Outline

- Machine learning for optimization: The revolution
- Warm-starting OPF with Machine Learning
- Obtaining approximate OPF solutions extremely quickly
- Obtaining feasible OPF solutions with Machine Learning?
- Future Directions

Consider one of the most popular algorithms to solve AC OPF: Newton-Raphson

AC OPF

$$\begin{aligned}\min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t} : \quad & g_i(\mathbf{x}) = 0, \quad i = 1, \ldots M \\ & h_j(\mathbf{x}) \leq 0, \quad j = 1, \ldots, P\end{aligned}$$

Typically:

Active/reactive power generation
Voltage magnitudes/angles

Newton step k+1

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha J^{-1}(\mathbf{x}^k) d(\mathbf{x}^k),$$

Jacobian of KKT conditions
Evaluated at $\mathbf{x}^k$

Vector of KKT conditions
evaluated at $\mathbf{x}^k$

Newton-Raphson. What makes it slow?

Using traditional Newton's method to solve AC OPF.

Using learning-boosted Newton's method to solve AC OPF.

Forming, and inverting, the Jacobian/Hessian

→ Introduce **Quasi-Newton** methods that either:

- → Approximate the Jacobian
- → Approximate the inverse of the Jacobian

Chord method:

Only periodically calculate J^{-1} and just deal with inexact search directions in between

Approximate Newton Directions[10]:

Assume block-diagonal Jacobian and distribute the computations

… many more

[10] A. Conejo, F. Nogales, and F. Prieto, "A decomposition procedure based on approximate Newton directions," *Mathematical Programming*, 2002.

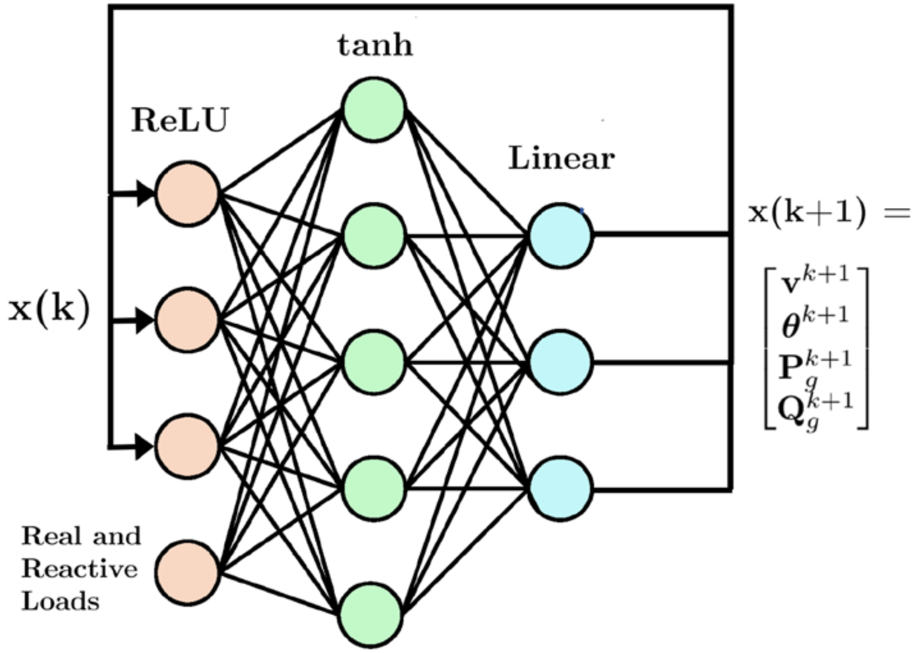
Can we…skip the whole Jacobian thing?

Newton step k+1

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha J^{-1}(\mathbf{x}^k)d(\mathbf{x}^k),$$

$$\mathbf{x}^{k+1} = F(\mathbf{x}^k)$$

Learning-Boosted Quasi-Newton[11]


tanh
ReLU
Linear
x(k)
$x(k+1) = \begin{bmatrix} \mathbf{v}^{k+1} \\ \boldsymbol{\theta}^{k+1} \\ \mathbf{P}_g^{k+1} \\ \mathbf{Q}_g^{k+1} \end{bmatrix}$
Real and Reactive Loads

Can we **learn** the next Newton step based on our current step?

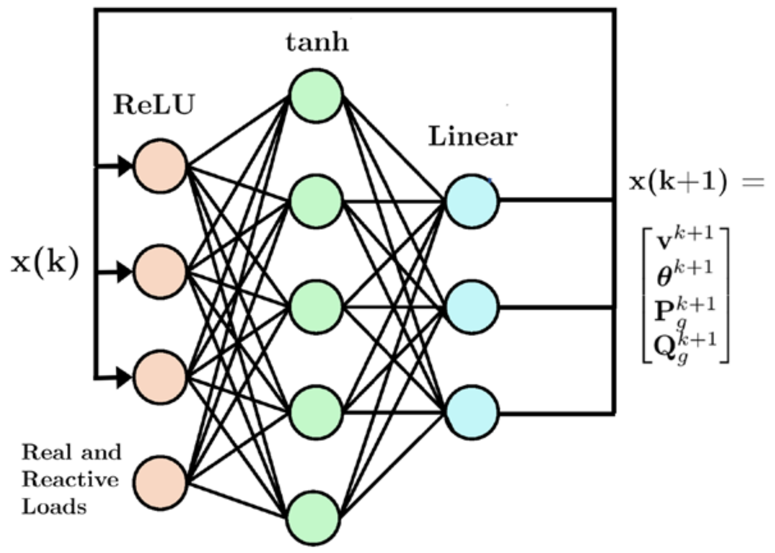
[11] K. Baker, "A Learning-boosted Quasi-Newton Method for AC Optimal Power Flow," https://arxiv.org/abs/2007.06074, Jul 2020.

Idea: Use a recurrent neural network to iterate

Both Newton-Raphson and Learning-Boosted Newton Raphson are fixed point iterations:

$$\mathbf{x}^{k+1} = F(\mathbf{x}^k)$$

But the learning-boosted method has an easier to evaluate *F()*


tanh
ReLU
Linear
x(k)
Real and Reactive Loads
$\mathrm{x(k+1)} = \begin{bmatrix} \mathbf{v}^{k+1} \\ \boldsymbol{\theta}^{k+1} \\ \mathbf{P}_g^{k+1} \\ \mathbf{Q}_g^{k+1} \end{bmatrix}$

Key Idea:

It may be easier to learn what direction to move in than to directly learn an optimal solution from an initial point

K. Baker, "A Learning-boosted Quasi-Newton Method for AC Optimal Power Flow," https://arxiv.org/abs/2007.06074, Jul 2020.

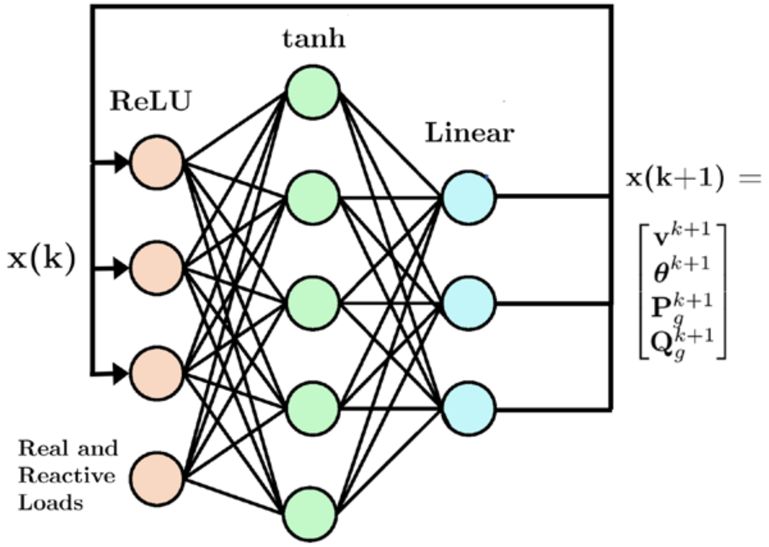
Guaranteed convergence : *F* is a Contraction

$$\mathbf{x}^{k+1} = F(\mathbf{x}^k)$$

Given restrictions on activation functions and bounds on weight magnitudes in the neural network, convergence of the learning-boosted model is guaranteed[12]!

(Note this just guarantees convergence, not convergence to the optimal.)

The network can be represented as a contraction mapping and shown to converge to a unique fixed point.

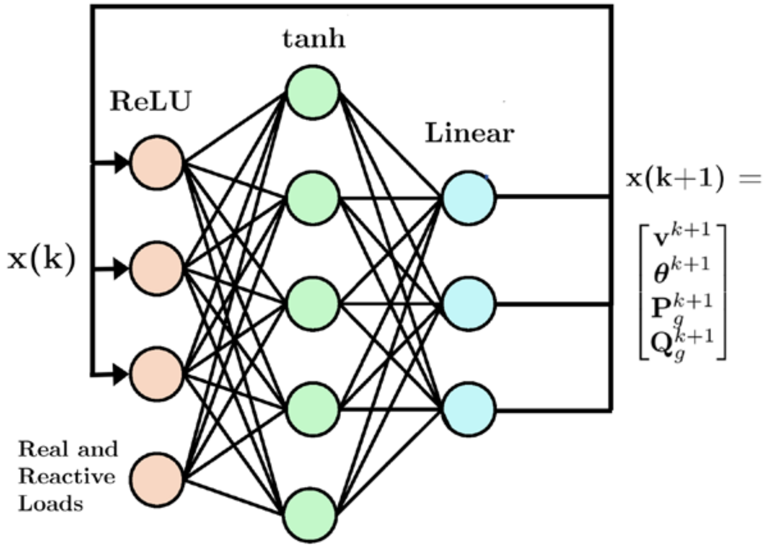

tanh
ReLU
Linear
$\mathbf{x(k)}$
$\mathbf{x(k+1)} = \begin{bmatrix} \mathbf{v}^{k+1} \\ \boldsymbol{\theta}^{k+1} \\ \mathbf{P}_g^{k+1} \\ \mathbf{Q}_g^{k+1} \end{bmatrix}$
Real and Reactive Loads

[12] J. E. Steck, “Convergence of recurrent networks as contraction map- pings,” in *Intl. Joint Conference on Neural Networks (IJCNN)*, Jun. 1992.

Other tricks and implementation

Thresholds on output layer can ensure upper/lower variable limits

Used a convergence criteria of 10^{-9} for generating training data to generate ”basins of attraction” for the model



tanh
ReLU
Linear
x(k)
Real and Reactive Loads
$\mathbf{x}(\mathbf{k+1}) = \begin{bmatrix} \mathbf{v}^{k+1} \\ \boldsymbol{\theta}^{k+1} \\ \mathbf{P}_g^{k+1} \\ \mathbf{Q}_g^{k+1} \end{bmatrix}$

Implementation details

Each training data point was a pair [x(k), x(k+1)] generated from iterations in the MATPOWER MIPS solver

A heuristic for number of nodes was used and then tuned. Predictors (inputs) normalized

Keras + Tensorflow were used with the Adam optimizer to train the network locally on my 3-year old laptop

30, 300, 500, 1,354-bus networks were tested

Convergence

Takes more iterations to converge because each iteration is using an approximate direction, but each iteration is faster

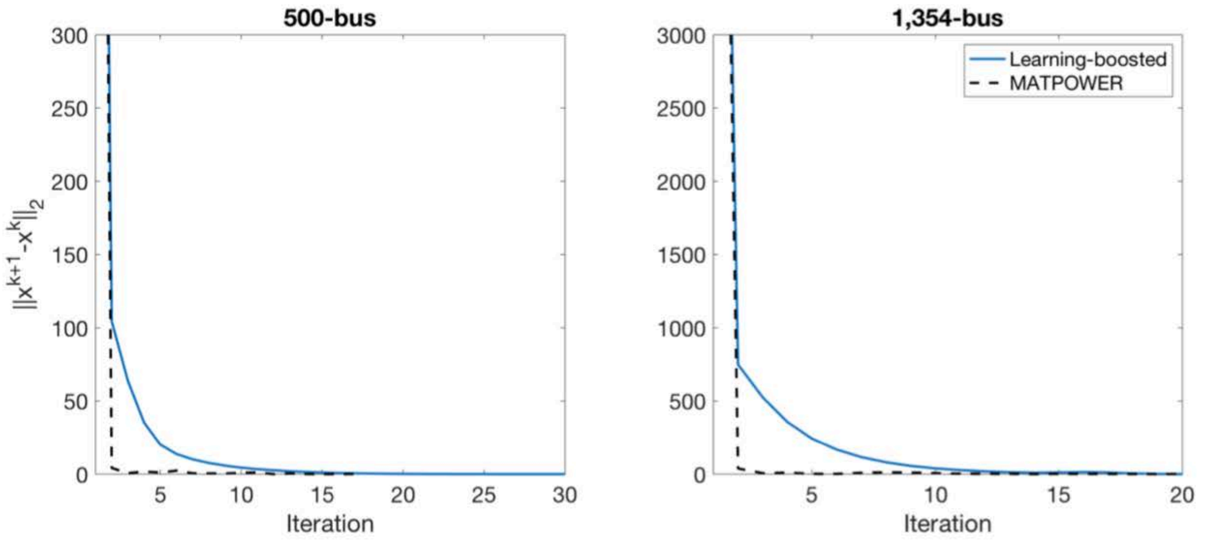
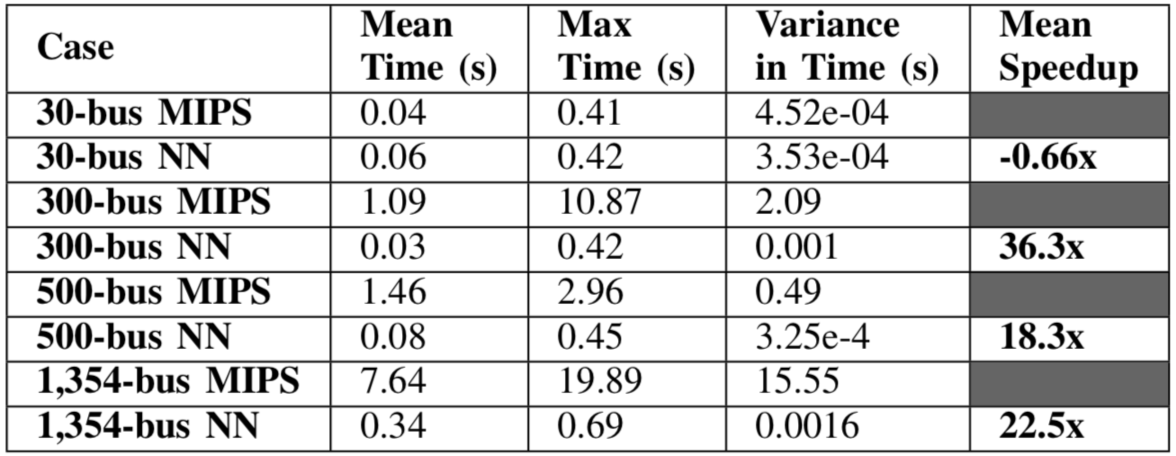


Fig. 3. Norm of $\mathbf{x}^{k+1} - \mathbf{x}^k$ for two scenarios in the 500-bus (left) and 1,354-bus (right) systems.

TABLE IV
TIME TO CONVERGENCE FOR EACH NETWORK.

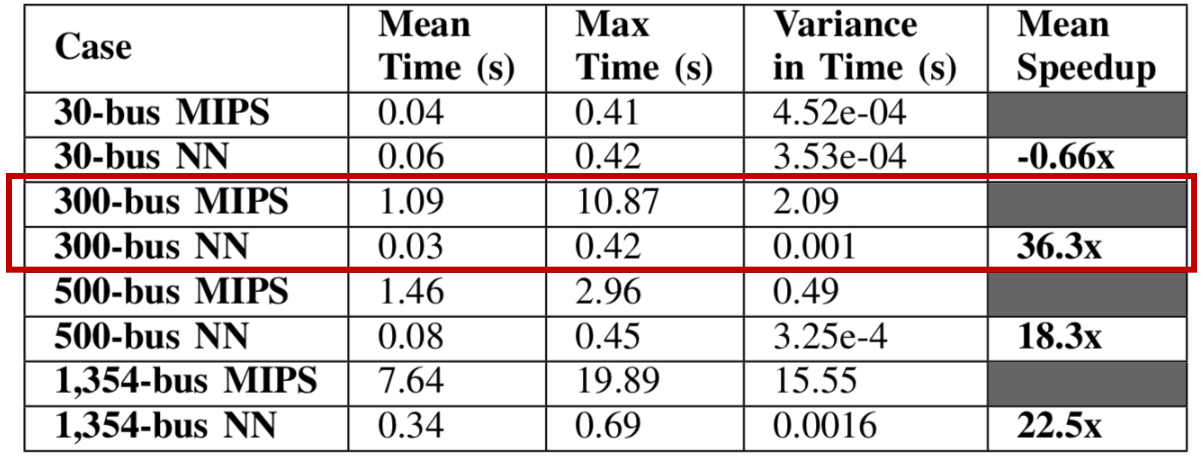


Case	Mean Time (s)	Max Time (s)	Variance in Time (s)	Mean Speedup
30-bus MIPS	0.04	0.41	4.52e-04	
30-bus NN	0.06	0.42	3.53e-04	**-0.66x**
300-bus MIPS	1.09	10.87	2.09	
300-bus NN	0.03	0.42	0.001	**36.3x**
500-bus MIPS	1.46	2.96	0.49	
500-bus NN	0.08	0.45	3.25e-4	**18.3x**
1,354-bus MIPS	7.64	19.89	15.55	
1,354-bus NN	0.34	0.69	0.0016	**22.5x**

Added benefit: No singular Jacobians!

The Jacobian can be singular near or even **at** the optimal solution in AC OPF, making convergence time slow and unpredictable.

TABLE IV
TIME TO CONVERGENCE FOR EACH NETWORK.



Case	**Mean Time (s)**	**Max Time (s)**	**Variance in Time (s)**	**Mean Speedup**
30-bus MIPS	0.04	0.41	4.52e-04	▨
30-bus NN	0.06	0.42	3.53e-04	**-0.66x**
300-bus MIPS	1.09	10.87	2.09	▨
300-bus NN	0.03	0.42	0.001	**36.3x**
500-bus MIPS	1.46	2.96	0.49	▨
500-bus NN	0.08	0.45	3.25e-4	**18.3x**
1,354-bus MIPS	7.64	19.89	15.55	▨
1,354-bus NN	0.34	0.69	0.0016	**22.5x**

No singular matrix inversions have to be dealt with in the learning boosted approach, making the time to convergence predictable and variance in convergence time low.

Optimality and Feasibility gaps

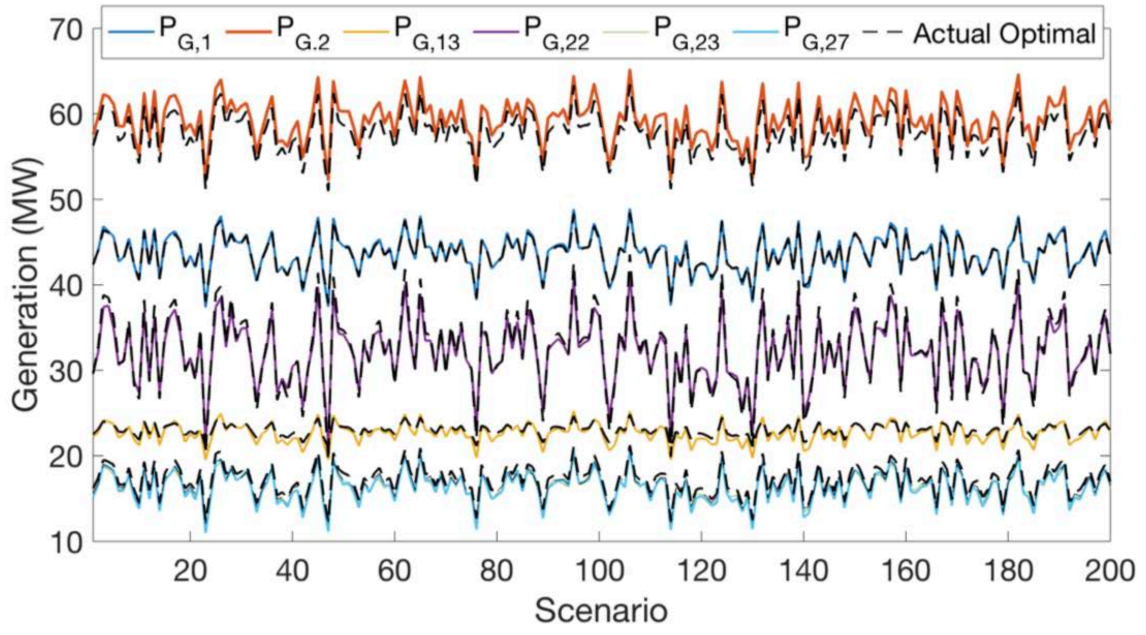


Fig. 2. Predicted generation values for 200 test scenarios (colors) and actual optimal values (black dashed line) for the IEEE 30-bus system.

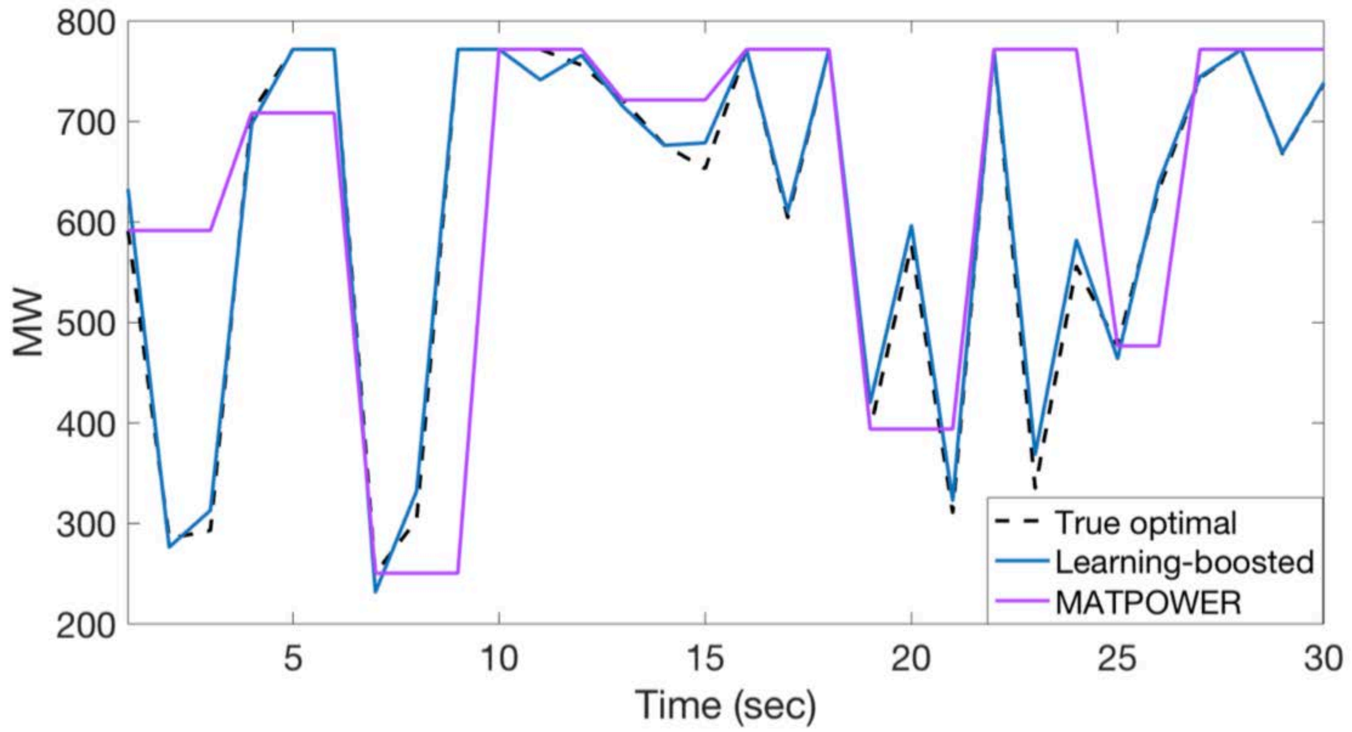
CONSIDERED NETWORK PARAMETERS

Case	# of Loads	# of Lines	# of Gens	Total Real Gen. Capacity
30-bus	20	41	6	335 MW
300-bus	191	411	69	32.68 GW
500-bus	200	597	56	12.19 GW
1,354-bus	621	1991	260	128.74 GW

MEAN ABSOLUTE ERROR ACROSS TESTING DATASET

Case	MAE: Voltage Magnitude (pu)	MAE: Active Power (MW)	MAPE: Cost (%)
30-bus	0.004 pu	0.64 MW	0.29%
300-bus	0.009 pu	10.47 MW	0.65%
500-bus	0.099 pu	0.62 MW	0.66%
1,354-bus	0.019 pu	7.55 MW	1.16%

Real-time optimal solution tracking



Slack-bus generator tracking optimal solutions (black dashed line) for the IEEE-500 bus system.

MATPOWER MIPS solver takes a couple seconds to solve, meaning its optimal generation setpoints are outdated in between solutions.

The learning-boosted method provides approximate optimal solutions in less than a second, making it more appropriate for real-time optimization.

Downsides to ML for OPF or learning-boosted OPF

Traditional optimization still has its upsides!

→ If a single constraint (e.g., a component) needs to be added or removed, it's likely you'll need an entirely separate ML model (you can do it since they're trained offline, but it would be annoying).

→ No guarantee on solution feasibility (there are ways you can do it for convex problems[13]); but a post-processing method can help with feasibility (more on this in a minute)

→ Improperly tuned models, not using enough training data, class imbalance, etc. can be disastrous! (designing a good NN architecture is a complicated problem)

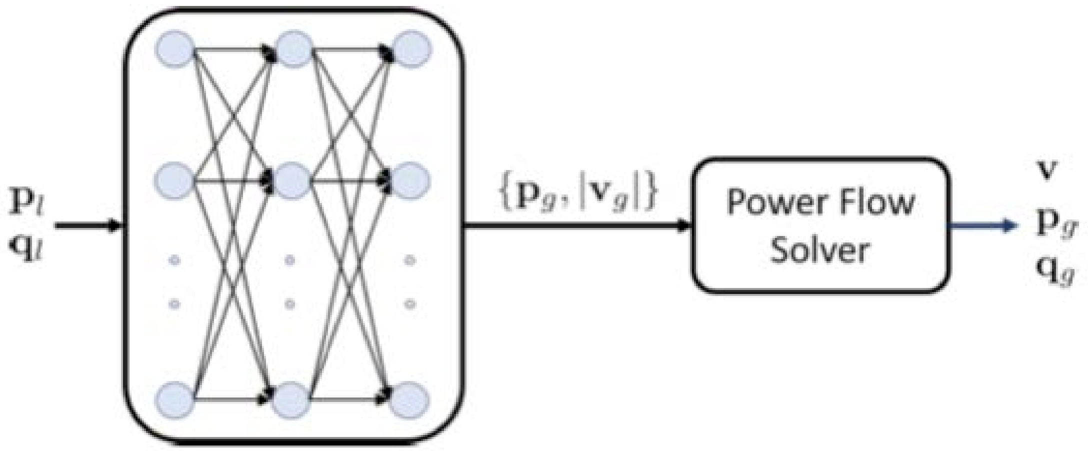
[13]Y. Chen, Y. Shi, B. Zhang, "Input Convex Neural Networks for Optimal Voltage Regulation," https://arxiv.org/abs/2002.08684, Feb. 2020

Talk Outline

- Machine learning for optimization: The revolution
- Warm-starting OPF with Machine Learning
- Obtaining approximate OPF solutions extremely quickly
- Obtaining feasible OPF solutions with Machine Learning?
- Future Directions

ML for OPF with feasibility recovery step

Based on work in collaboration with Ahmed Zamzam at NREL[14]



Overall idea: Use neural network to obtain subset of OPF variables sufficient to find the rest of the solution. Use nonlinear equation solver to obtain full OPF solution

Tricks:

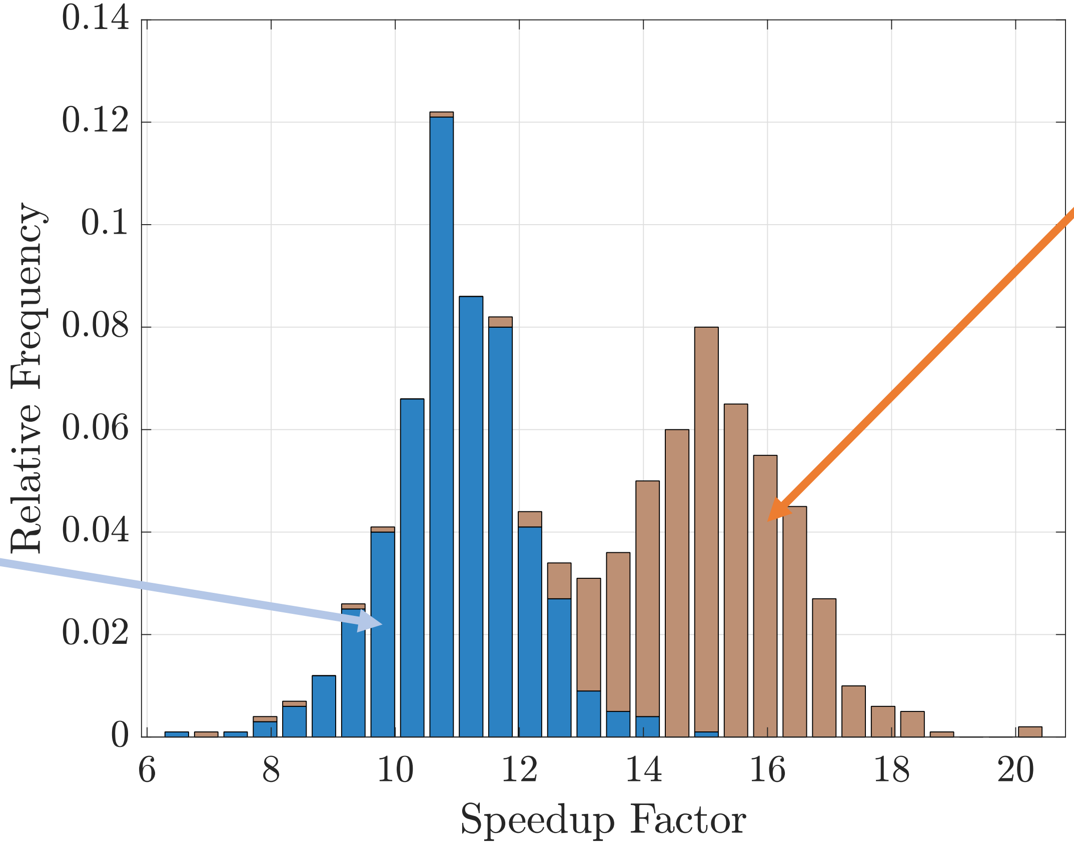
- Generate training samples with slightly restricted feasible regions to push samples away from constraint boundaries
- Parameterization of variables / output layer ensures variable upper/lower limits satisfied

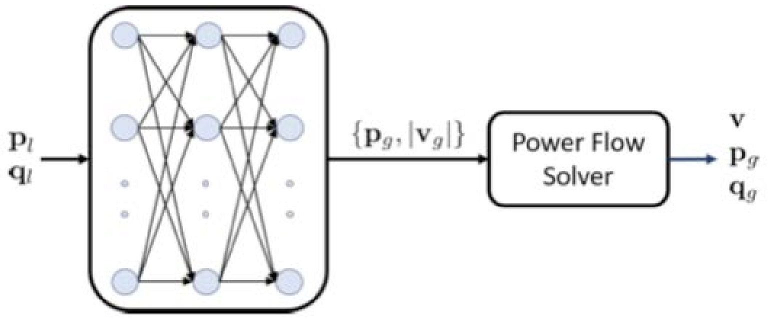
[14] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," https://arxiv.org/abs/2861719, Sept. 2019

Speedup = depends how often are we infeasible

Predicted solution **was** feasible = bigger speedups

Predicted solution **wasn't** feasible = smaller speedups





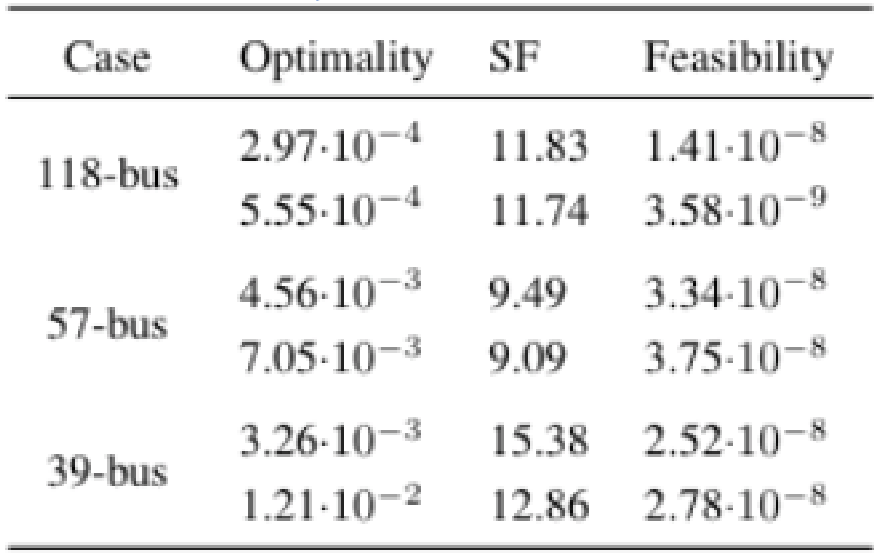
[14] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," https://arxiv.org/abs/2861719, Sept. 2019

Feasibility and Optimality gaps?

Average gap between optimal and predicted generation

Average speedup

Maximum power flow constraint violation



Case	Optimality	SF	Feasibility
118-bus	$2.97 \cdot 10^{-4}$	11.83	$1.41 \cdot 10^{-8}$
	$5.55 \cdot 10^{-4}$	11.74	$3.58 \cdot 10^{-9}$
57-bus	$4.56 \cdot 10^{-3}$	9.49	$3.34 \cdot 10^{-8}$
	$7.05 \cdot 10^{-3}$	9.09	$3.75 \cdot 10^{-8}$
39-bus	$3.26 \cdot 10^{-3}$	15.38	$2.52 \cdot 10^{-8}$
	$1.21 \cdot 10^{-2}$	12.86	$2.78 \cdot 10^{-8}$

[14] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," https://arxiv.org/abs/2861719, Sept. 2019

Talk Outline

- Machine learning for optimization: The revolution
- Warm-starting OPF with Machine Learning
- Obtaining approximate OPF solutions extremely quickly
- Obtaining feasible OPF solutions with Machine Learning?
- Future Directions

So…real-time AC OPF is a solved problem?

So…real-time AC OPF is a solved problem?

Absolutely not! These are just some promising first steps

Embedding constraints into the NN is an area of active research
→ Can penalize constraint violations in the loss function[15]

Optimal design of NN architectures

Using ML for both classification + regression problems (Unit Commitment, etc.)

[15]M. Chatzos, F. Fioretto, T. Mak, P. Van Hentenryck, "High-Fidelity Machine Learning Approximations of Large-Scale Optimal Power Flow," https://arxiv.org/abs/2006.16356, Jun 2020.

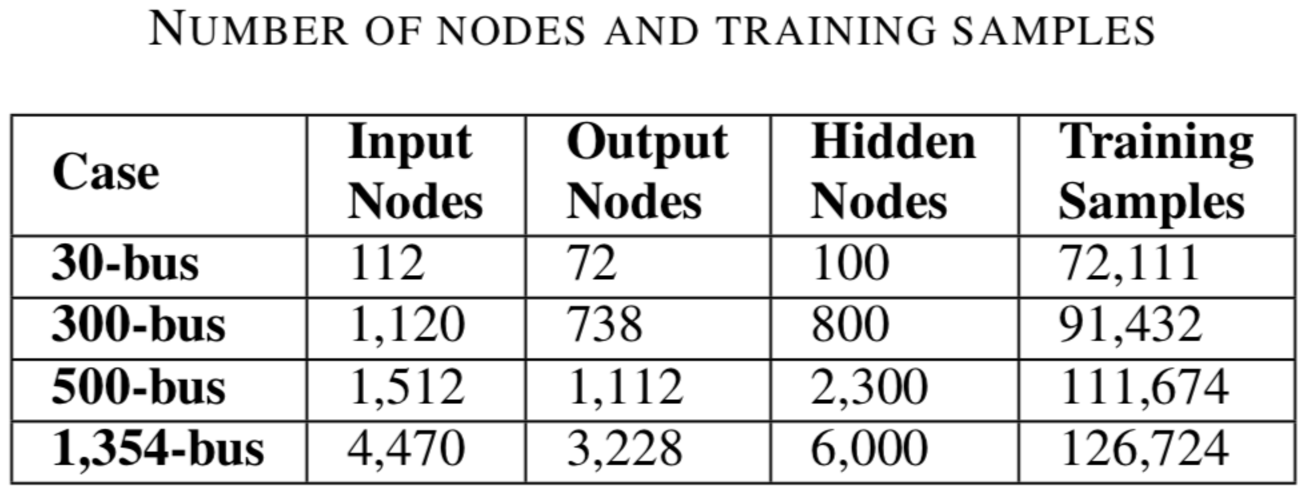
Thank you! Questions?

Kyri Baker
Kyri.baker@colorado.edu

GRid-Interactive Frameworks For Intelligent iNfrastructure (GRIFFIN) Lab

Baker Research Group at the University of Colorado Boulder

Extra Slide: NN parameters



NUMBER OF NODES AND TRAINING SAMPLES

Case	Input Nodes	Output Nodes	Hidden Nodes	Training Samples
30-bus	112	72	100	72,111
300-bus	1,120	738	800	91,432
500-bus	1,512	1,112	2,300	111,674
1,354-bus	4,470	3,228	6,000	126,724