Ensuring DNN Solution Feasibility for Optimization Problems with Linear Constraints

Tianyu Zhao[1,2], Xiang Pan[2], Minghua Chen[3], Steven Low[4]

[1]Lenovo Machine Intelligence Center, Lenovo
[2]Department of Information Engineering, The Chinese University of Hong Kong
[3]School of Data Science, City University of Hong Kong
[4]California Institute of Technology

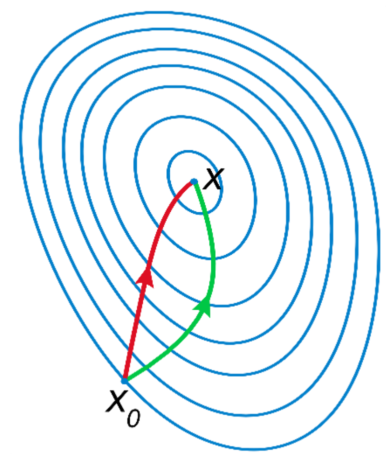
ICLR 2023
Paper ID: 2175

Acknowledgment: Part of the slides are adapted from the ACM SIGMETRICS / IFIP Performance 2022, Tutorial, June

An Input-Solution Mapping Perspective for Constrained Optimization

$$
\begin{aligned}
\min_{x} \quad & f(x, z) \\
s.t. \quad & g_i(x, z) = 0, \;\; i = 1, \dots, n \\
& h_i(x, z) \le e_j, \;\; j = 1, \dots, m
\end{aligned}
$$

z: input parameter vector
x: decision variable vector



Gradient descent (green)
Newton's method (red)

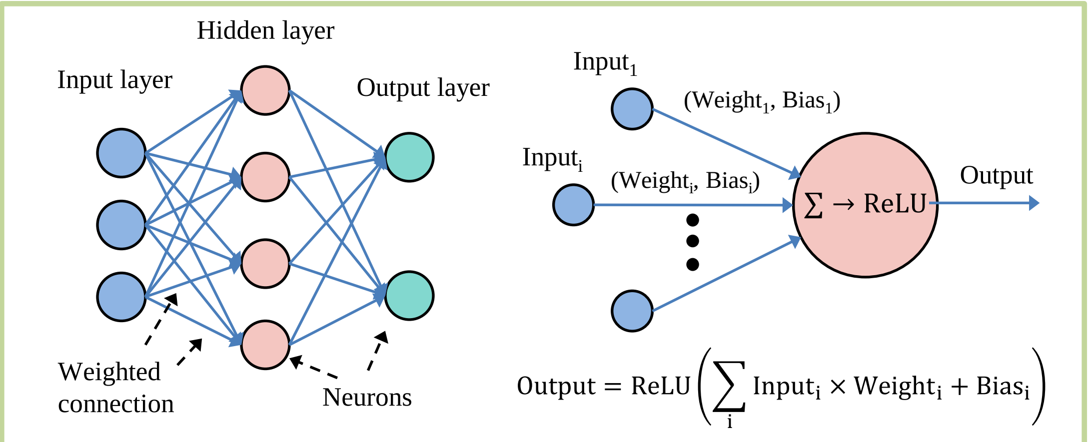
Input → Solution

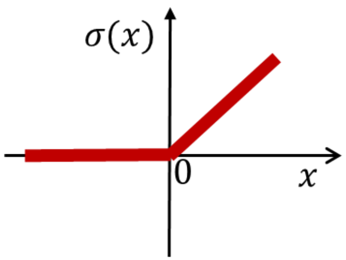
$\dots, z_2, z_1 \rightarrow$ **Solver** $\rightarrow \dots, x_2^*, x_1^*$

- Tremendous applications; many off-the-shelf solvers
- A solver implicitly characterizes an input-solution mapping for a problem

Picture source: wiki

New Machine Learning Viewpoint by DNN





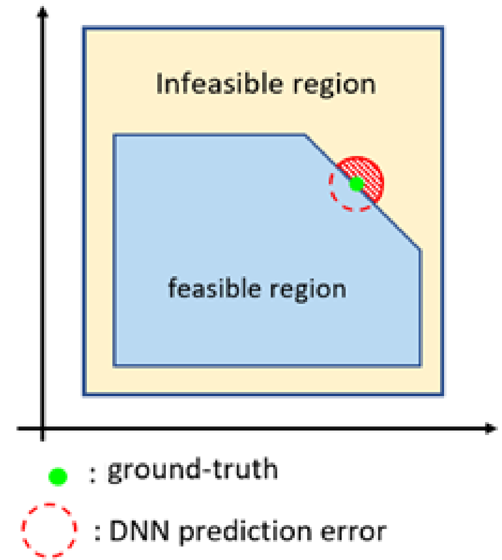
Rectified linear unit (ReLU): $\max(x, 0)$

- Learn the input-solution mapping for a given problem
- Pass input through the mapping for solution
  - Low run-time complexity for real-time scenarios
  - Learning complexity is amortized if the problem is solved repeatedly
- **Q**: can we learn such a mapping?

Challenges and Motivations

$$\min_{x} \quad f(x, z)$$
$$s.t. \quad g_i(x, z) = 0, \; i = 1, \ldots, n$$
$$h_i(x, z) \leq e_j, \;\; j = 1, \ldots, m$$

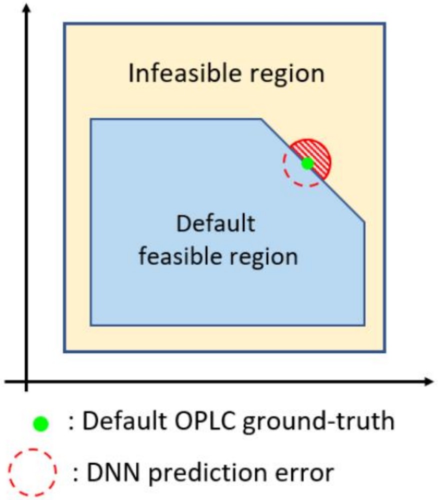
z: input parameter vector
x: decision variable vector

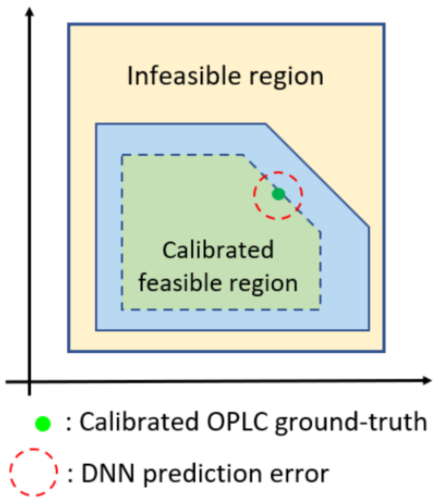


- □ The learned solution from ML models should be as close as the optimum x^* : optimality requirement
- □ The solution should respect the constraints: feasibility requirement
- □ **Q**: can we achieve such a goal?

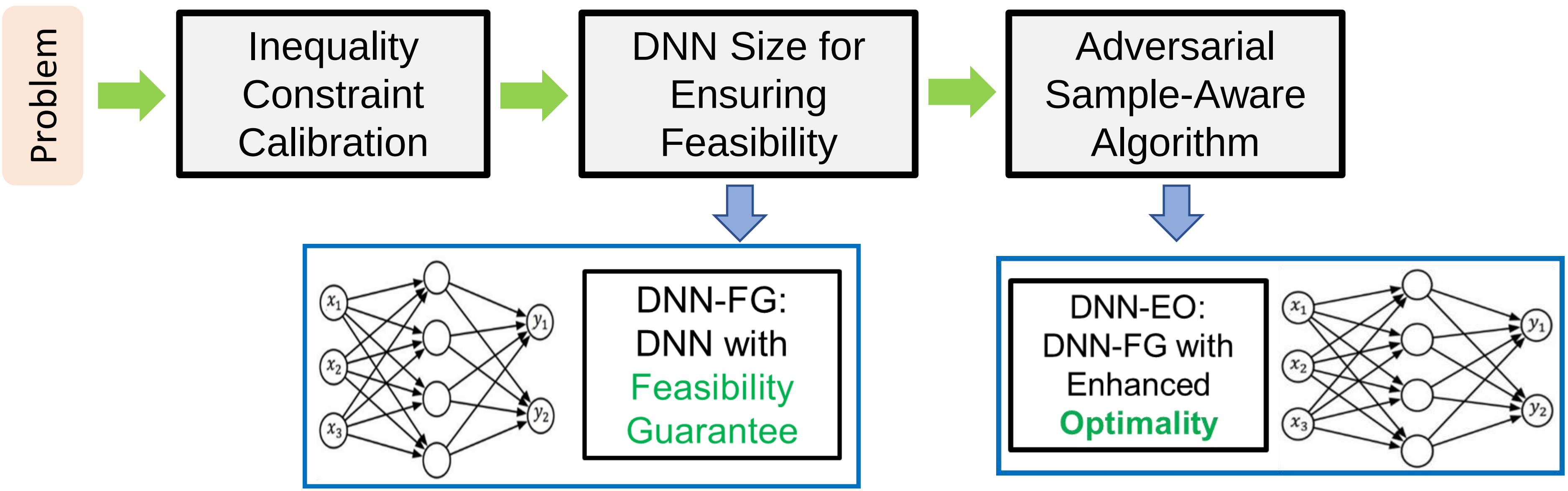
Preventive Learning (PL)

- Train DNN with a calibrated feasible set
  - Still supporting the full input region
- With prediction error, DNN solutions are feasible w.r.t. the original constraints
- Inevitable optimality loss if the optimal solution is at the boundary
  - Use larger DNN to reduce optimality loss





PL for Optimization with Linear Constraints



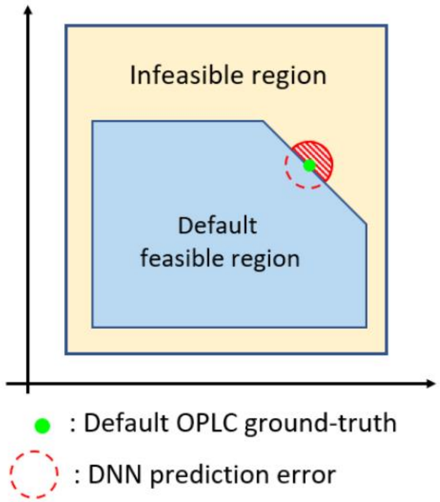
1. Determine the maximum allowed calibration rate
2. Determine the DNN size needed to ensure feasibility
   – Without training, output a DNN-FG with provably guaranteed feasibility
3. Adversarial-sample aware training to pursue strong optimality performance while maintaining feasibility

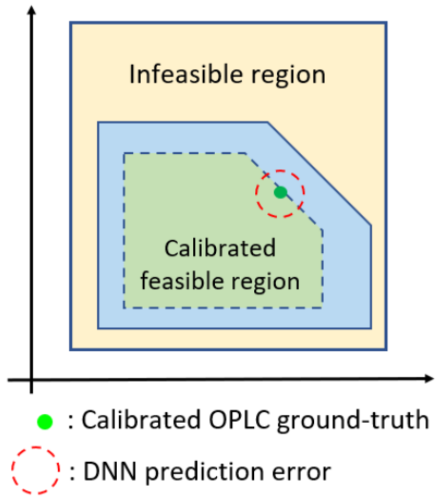
Step 1: Calibrating Inequality Constraints

- Rewrite the OPLC with only inequality constraints by using variable reduction techniques

- Calibrating inequality constraints
  - $h_j \leq \begin{cases} e_j(1-\eta_j), e_j \geq 0 \\ e_j(1+\eta_j), e_j < 0 \end{cases}$
  - $\eta_j \in [0, \infty]$: Calibration rate

- Solve a min-max problem to find η_{max} that supports all possible input
  - $\eta_{max} = 0$ means no feasibility guarantee
  - A lower bound on η_{max} can be found in polynomial time, denoted by Δ





Step 2: Determining Sufficient DNN Size

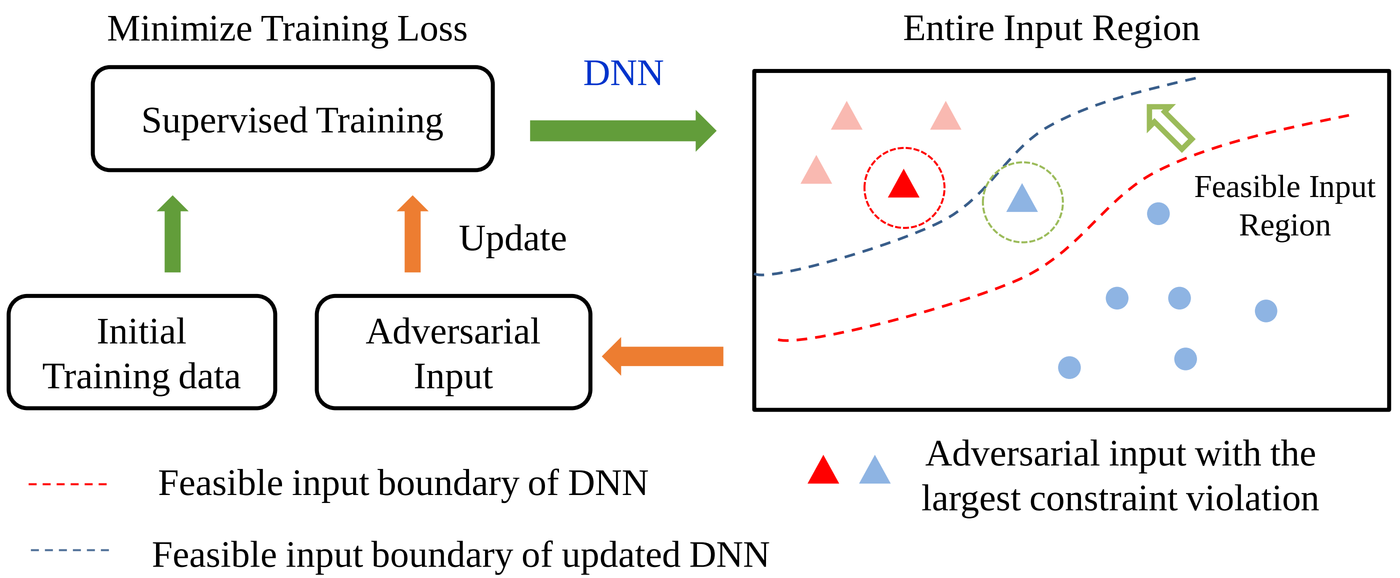
- Given a ReLU DNN with n hidden layers and m neurons per layer, we optimize parameters (W, b) to minimize the worst-case constraint violation by an ILP approach [1,2]
  - An upper bound of the best worst violation can be found in polynomial time, denoted as ρ

- We double the DNN width, m, until $\rho < \Delta$

- **Feasibility guarantee**: We obtain a DNN with feasibility guarantee! (named DNN-FG)

[1] V. Tjeng, K. Xiao, and R. Tedrake. Evaluating robustness of neural networks with mixed integer programming. In International Conference on Learning Representations, 2019.
[2] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis. Learning optimal power flow: Worst-case guarantees for neural networks. IEEE SmartGridComm, 2020.

Step 3: Adversarial Sample Aware Training



- Loss function captures both the prediction errors and constraint violation

$$\mathcal{L}^t = w_1 \frac{1}{N} \sum_{i=1:N} (\hat{x} - x^*)^2 + w_2 \frac{1}{M} \sum \text{Violation of each calibrated inequality constraints}$$

Experiments

- DC-OPF problem

$$
\begin{aligned}
\min \quad & \sum_{i\in\mathcal{N}} \left(\lambda_{i,2} p_{gi}^2 + \lambda_{i,1} p_{gi} + \lambda_{i,0}\right) \\
\text{s.t.} \quad & \mathbf{B}\boldsymbol{\theta} = \boldsymbol{p}_G - \boldsymbol{p}_D, \\
& b_{ij}\left(\theta_i - \theta_j\right) \le S_{ij}^{\max}, \quad \forall (i,j) \in \mathcal{E}, \\
& \boldsymbol{P}_G^{\min} \le \boldsymbol{p}_G \le \boldsymbol{P}_G^{\max} \\
\text{var.} \quad & p_{Gi}, \theta_i, \forall i \in \mathcal{N}.
\end{aligned}
$$

Quadratic generation cost

Linearized power balance equation

Branch flow and power generation limits

- Non-convex optimization[1]

$$
\min_{y \in \mathcal{R}^n} \quad \frac{1}{2} y^T Q y + p^T \sin(y), \text{s.t.} \quad Ay = x, -h \le Gy \le h,
$$

[1] P. L. Donti, D. Rolnick and J. Z. Kolter, "DC3: a learning method for optimization with hard constraints", in Proceedings of 9th International Conference on Learning Representations (ICLR), virtual conference, May 3 – 7, 2021.

Case Study for Solving DC-OPF Problems

- We design DeepOPF+ by the preventive learning framework
- Test cases and maximum calibration rates
  - Input load region [100%, 115%] and [115%, 130%]

	IEEE Case30	IEEE Case118	IEEE Case300
Maximum calibration rate	7.0%	16.7%	21.6%

- DNN size for ensuring solution feasibility (3 hidden layers)

	IEEE Case30	IEEE Case118	IEEE Case300
DNN size	32/16/8	128/64/32	256/128/64

Consistent Speedup and Optimality

- DeepOPF+ with 3% and 7% calibration rates achieves consistent speedup and minor optimality loss

Case	Scheme	Average speedups		Feasibility rate (%)		Optimality loss (%)		Worst-case violation (%)	
		light-load	heavy-load	light-load	heavy-load	light-load	heavy-load	light-load	heavy-load
Case30	DNN-P	×85	×86	100	88.12	0.02	0.03	0	5.43
	DNN-D	×85	×84	100	93.36	0.02	0.03	0	11.19
	DNN-W	×0.90	×0.86	100	100	0	0	0	0
	DNN-G	×24	×26	100	100	0.13	0.04	0	0
	DeepOPF+-3	×86	×92	100	100	0.03	0.04	0	0
	DeepOPF+-7	×86	×93	100	100	0.03	0.09	0	0
Case118	DNN-P	×137	×125	68.84	54.92	0.17	0.21	19.5	44.8
	DNN-D	×138	×124	73.42	55.37	0.20	0.24	16.69	43.1
	DNN-W	×2.08	×2.26	100	100	0	0	0	0
	DNN-G	×26	×16	100	100	1.29	0.39	0	0
	DeepOPF+-3	×201	×226	100	100	0.18	0.19	0	0
	DeepOPF+-7	×202	×228	100	100	0.37	0.41	0	0
Case300	DNN-P	×115	×98	91.29	78.42	0.06	0.08	261.1	443.0
	DNN-D	×115	×102	91.99	82.92	0.07	0.07	231.6	348.1
	DNN-W	×1.04	×1.08	100	100	0	0	0	0
	DNN-G	×2.44	×2.65	100	100	0.32	0.06	0	0
	DeepOPF+-3	×129	×136	100	100	0.03	0.03	0	0
	DeepOPF+-7	×130	×138	100	100	0.10	0.06	0	0

[1] Xiang Pan, Tianyu Zhao, Minghua Chen, and Shengyu Zhang. Deepopf: A deep neural network approach for security-constrained DC optimal power flow. IEEE Transactions on Power Systems, 36(3):1725–1735, 2020.
[2] Priya L Donti, David Rolnick, and J Zico Kolter. DC3: A learning method for optimization with hard constraints. arXiv preprint arXiv:2104.12225, 2021.
[3] Wenqian Dong, Zhen Xie, Gokcen Kestor, and Dong Li. Smart-pgsim: using neural network to accelerate AC-OPF power grid simulation. In SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, 2020.
[4] Meiyi Li, Soheil Kolouri, and Javad Mohammadi. Learning to solve optimization problems with hard linear constraints. arXiv preprint arXiv:2208.10611, 2022.

Consistent Speedup and Optimality

- DNN scheme with 5% and 10% calibration rates achieves consistent speedup and minor optimality loss

Scheme	Average objective			Average running time (ms)			Feasibility rate (%)	Worst-case violation (%)
	Scheme	Ref.	Loss (%)	Scheme	Ref.	Speedup		
DNN-P	-5.44		0.40	1.36		85.7	39.8	68.3
DNN-D	-5.44		0.42	0.79		117.0	39.8	41.5
DNN-W	-5.47	-5.47	0	86.6	86.6	1.02	100	0
DNN-G	53.69		1076.0	1.00		87.0	100	0
Pre-DNN-5	-5.45		0.34	0.60		144.9	100	0
Pre-DNN-10	-5.43		0.67	0.60		145.3	100	0

Conclusion and Future work

- Conclusion
  - Design Preventive Learning as the first framework to guarantee DNN solution feasibility
  - Simulations show the higher speedup and minor optimality loss of our design

- Future work
  - Solution feasibility for non-convex constrained optimization
  - Application to larger problem size and DNN size
  - Setting up the DNNs more efficiently and accelerate the corresponding steps

Thanks.