Power Engineering Letters

Emulating AC OPF Solvers With Neural Networks

Kyri Baker, *Member, IEEE*

***Abstract*—Using machine learning to obtain solutions to AC optimal power flow has recently been a very active area of research due to the astounding speedups that result from bypassing traditional optimization techniques. However, generally ensuring feasibility of the resulting predictions while maintaining these speedups is a challenging, unsolved problem. In this letter, we train a neural network to emulate an iterative solver in order to cheaply and approximately iterate towards the optimum. Once we are close to convergence, we then solve a power flow to obtain an overall AC-feasible solution. Results shown for networks up to 1,354 buses indicate the proposed method is capable of finding feasible, near-optimal solutions to AC OPF in milliseconds on a laptop computer. In addition, it is shown that the proposed method can find "difficult" AC OPF solutions that cause flat-start or DC-warm started algorithms to diverge.**

***Index Terms*—Machine learning, optimal power flow, computation time.**

I. Introduction

AC OPTIMAL power flow (OPF) is a canonical power systems operation problem that is at the heart of optimizing large-scale power networks. Solving this problem quickly and efficiently has been the subject of decades of research. One particularly interesting development in this area is the use of machine learning (ML), in particular deep learning, to obtain solutions to AC OPF [1]–[3]. Within this area, ensuring feasibility of the resulting solution has been a challenge. In this paper, we propose a neural network model which aims to emulate an AC OPF solver (in particular, the Matpower Interior Point Solver (MIPS), although the framework is not specific to MIPS). The benefit of using an ML model instead of the MIPS solver directly is that no matrix inverses or factorizations are needed, and inference is extremely fast, resulting in an overall faster convergence. While we do not claim that feasibility can be guaranteed for every single output of the learning-based model, empirically, we have observed very positive results on the chosen networks in terms of optimality gap, speed, and convergence success. The model proposed here is comprised of a fully connected three-layer neural network (NN) F_R with feedback, where input x^k is the candidate optimal solution vector at iteration k. Reminiscent of a simple recurrent neural network, the model iteratively uses feedback from the output layer as inputs until convergence ($||\mathrm{x}^{k+1} - \mathrm{x}^k|| \leq \epsilon$). The model thus bypasses any construction of a Jacobian matrix or associated inverse, for example. In order to obtain a final AC feasible solution, a subset of the learned variables are sent to a power flow solver. See Fig. 1 for an overview of the testing phase of the algorithm, where the tilde over the variables indicates the candidate optimal solution produced by the model. Some works have looked at penalizing constraint violations during training, which can help preserve AC feasibility, but cannot guarantee it [2], [3]. In addition, these techniques can result in very cumbersome-to-train loss functions. Other work using ML for AC OPF has recovered AC feasibility by using a post-processing procedure with the AC power flow equations, but requires a restricted training set generated from a modified AC OPF problem, sometimes requires PV/PQ switching, and was only tested on small networks [1]. The approach presented in this paper utilizes a similar concept to the latter technique, but offers advantages over all previous techniques:

- No restriction on the training set is required; previous AC OPF runs can be used to train the ML model.
- The model *emulates* an iterative algorithm, meaning that each model run is a small step towards the optimum, instead of directly predicting the AC OPF solution.
- While only a subset of variables is sent to the power flow solver, the ML model utilizes information about the entire OPF solution, better informing the model as it iterates towards the optimum. Evolutions of variable iterations

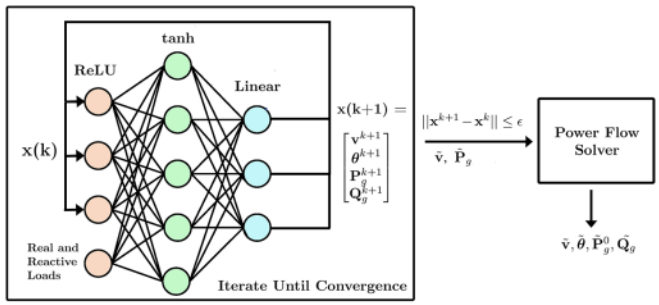

Fig. 1. Using a neural network to approximate fast iterations towards the optimum, then solving a power flow to recover feasibility.

Manuscript received 17 December 2020; revised 20 July 2021, 19 December 2021, and 6 June 2022; accepted 19 July 2022. Date of publication 29 July 2022; date of current version 20 October 2022. This work was supported by the National Science Foundation under CAREER Award 2041835. Paper no. PESL-00369-2020.

The author is with the University of Colorado Boulder, Boulder, CO 80309 USA (e-mail: kyri.baker@colorado.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TPWRS.2022.3195097.

Digital Object Identifier 10.1109/TPWRS.2022.3195097

0885-8950 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: UNIVERSITY OF COLORADO. Downloaded on January 01,2023 at 18:41:44 UTC from IEEE Xplore. Restrictions apply.

towards the optimum can also be observed while using the model, unlike completely black-box approaches.
- A deep network is not required for this framework; in this paper, a shallow neural network (only one hidden layer) is used, which facilitates very fast training (under 12 hours on a laptop) and simpler parameter tuning.

These works [1]–[3] additionally only perturb the given loads by 10% or 20% in both training and testing, which results in a much smaller set of optimal solutions, making it easier for the model to map these conditions onto optimal values. Here, we generate data than spans a larger region of the feasible space, including solutions which represent the state of the system near voltage collapse. These situations pose challenging situations for traditional solvers. Results are shown for 30, 500, and 1,354-bus networks suggesting that the learning-based model may provide a desirable tradeoff between speed and optimality gap while maintaining feasibility.

II. Iterating Using Inference

A nonconvex optimization problem with n-dimensional variable vector x, cost function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$, M equality constraints $g_i(x) = 0, g_i(\cdot) : \mathbb{R}^n \to \mathbb{R}$, and P inequality constraints $h_j(x) \leq 0$, $h_j(\cdot) : \mathbb{R}^n \to \mathbb{R}$ can be written as

$$\min_{\mathbf{x}} \ f(\mathbf{x}) \tag{1a}$$

$$\text{s.t} : g_i(\mathbf{x}) = 0, \ i = 1, \ldots M \tag{1b}$$

$$h_j(\mathbf{x}) \leq 0, \ j = 1, \ldots, P \tag{1c}$$

Many iterative optimization solvers use Hessians of the Lagrangian function to iterate towards the optimal solution of constrained nonconvex problems, including the Matpower Interior Point Solver (MIPS) [4], which leverages a primal-dual interior point algorithm to update candidate solution $\mathbf{x}^k$ at iteration k. Instead of using Lagrangian functions or forming Hessian matrices, the learning-based method uses a deep learning model $F_R(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ that takes in $\mathbf{x}^k$ as an input and provides $\mathbf{x}^{k+1}$ as an output; e.g. $\mathbf{x}^{k+1} = F_R(\mathbf{x}^k)$.

The fully connected three-layer NN has inputs/outputs $\mathbf{x}^k = [\mathbf{v}^k, \boldsymbol{\theta}^k, \mathbf{P}_g^k, \mathbf{Q}_g^k]^T$, where the iteration index is k, v contains the complex voltage magnitudes at each bus, $\boldsymbol{\theta}$ contains the complex voltage angles, and $\mathbf{P}_g$ and $\mathbf{Q}_g$ are the real and reactive power outputs at each generator, respectively.

A. Network Architecture

A rectified linear unit (ReLU) was used for the input layer; a hyperbolic tangent (tanh) activation function was used in the hidden layer; and a linear function was used on the output. Bounds on generation and voltages were enforced with a threshold on the output layer of the NN. Normalization of the inputs was also performed such that all inputs were in the range [0,1]. In addition to $\mathbf{x}^k$, the network loading was given as an input to the NN. The data generation, training and testing of the network, and simulations were performed on a 2017 MacBook Pro laptop with 16 GB of memory. Keras with the Tensorflow backend was used to train the neural network using the Adam optimizer.

TABLE I
THE ML SOLUTION FINDS FEASIBLE AC OPF SOLUTIONS FASTER THAN TRADITIONAL METHODS AND SOLVES IN A COMPARABLE TIME TO DC OPF

Network	OPF Type	Average Solve Time (s)	Percent Solve Success (%)
30-bus	AC OPF w/Flat Start	0.069 s	100%
	AC OPF w/DC Start	0.079 s	100%
	AC OPF w/PF Start	0.068 s	100%
	NN with PF	0.050 s	100%
	DC OPF	0.010 s	100%
	DC OPF w/PF	0.018 s	100%
500-bus	AC OPF w/Flat Start	3.26 s	100%
	AC OPF w/DC Start	3.34 s	100%
	AC OPF w/PF Start	3.15 s	100%
	NN with PF	0.12 s	100%
	DC OPF	0.12 s	100%
	DC OPF w/PF	0.13 s	100%
1354-bus	AC OPF w/Flat Start	20.12 s	59.4%
	AC OPF w/DC Start	11.63 s	59.4%
	AC OPF w/PF Start	10.64 s	59.8%
	NN with PF	0.25 s	98.00%
	DC OPF	0.33 s	100%
	DC OPF w/PF	0.38 s	99.20%

B. Data Generation

The MATPOWER Interior Point Solver (MIPS) [4] was used to generate the data and was used as the baseline for comparison with the NN model. A single training sample consists of the pair $[\mathbf{x}^k, \mathbf{x}^{k+1}]$ obtained from the solver. The termination tolerance of the MIPS solver was set to 10^{-9} for data generation and 10^{-4} for testing. The tolerance of the learning-based solver was set to 10^{-4}, where convergence is reached when $||\mathbf{x}^{k+1} - \mathbf{x}^k|| \leq \epsilon$. A smaller tolerance was used for data generation to promote smoother convergence and "basins of attraction" within the ML model. For a fair comparison, the same convergence criteria was used for the NN model and MIPS during testing. 500 different loading scenarios were randomly generated at each load bus from a uniform distribution of $\pm 40\%$ around the given base loading scenario in MATPOWER. Each generated set of loads was used to solve a standard power flow, and if the power flow could not find a solution, the data was not included.

III. Simulation Results

The IEEE 30 bus, IEEE 500 bus, and 1,354-bus PEGASE networks [5] were considered. All training/testing was performed locally on a MacBook Pro with 16 GB of RAM. To be consistent with the constraint set across networks, because some of the considered networks do not contain line flow limits, these were neglected in all networks. However, it is not expected that the inclusion of these constraints would dramatically change the results shown here. In Table I, the learning-based method ("NN") was compared with 4 other cases across the 500-sample training set: AC OPF flat-started, AC OPF warm-started with both a DC

Authorized licensed use limited to: UNIVERSITY OF COLORADO. Downloaded on January 01,2023 at 18:41:44 UTC from IEEE Xplore. Restrictions apply.

OPF solution and a power flow solution, a DC OPF, and a DC OPF with an AC PF in post-processing to pursue AC feasibility. While the DC OPF never produces an AC feasible solution, it is often used as an approximation for AC OPF and used in many LMP-based markets to calculate prices and thus provides an interesting comparison. Note that no "extra" feasibility steps were performed; the output of the NN was directly sent to a power flow solver.

A. Speedups

As Table I shows, especially for the 500 and 1,354-bus networks, the learning-based solution can provide solutions with speedups of over twenty times faster than using a traditional solver, even one that has been warm-started. A DC OPF, in comparison, takes about the same amount of time to solve a much simpler, convex problem, and, considering it entails a linear set of constraints, has no issues finding a solution. For smaller networks, the benefit of using a ML-based method is negligible; sub-second solution times are already achieved by standard AC OPF methods. Another interesting idea is to use a ML model to warm-start the AC OPF as in [6] or the AC PF as in [7], but we only compare with traditional warm-start methods here. Lastly, it should be mentioned that the NN-based solver can be trained and tested using data generated from commercial solvers such as CPLEX or Gurobi which may be able to improve the performance of all of the test cases here. As the training data for the NN is simply the variable vector at each iteration, any iterative algorithm can be used for data generation. We expect the *relative* speedups to remain relatively consistent regardless of the solver, however.

B. Challenging OPF Scenarios

For system loadings that challenge conventional solvers, continuation methods can be used to robustly solve the AC OPF by solving a series of simpler OPF problems [8]. While robust, these methods can be very time consuming and may not be suitable for real-time operation. The learning-based method may provide an alternative to these methods, as issues with singular Jacobian matrices and solutions close to voltage instability do not affect inference as much. These solutions may still prove challenging for the post-processing power flow step, however, which is perhaps why a few failures were still encountered when using the learning-based method. In Table I, the last column refers to the percentage of trial runs in which the solver converged to a feasible solution. For the AC OPF cases, this refers to successful convergence of the MIPS AC OPF solver. For the NN and DC OPF w/PF case, this refers to the MIPS AC PF solver converging to a point which satisfies the AC power flow constraints. Lastly, for the DC OPF case, this refers to the percentage of runs in which the MIPS solver found the optimal solution to the DC OPF problem. Considering the DC OPF problem is convex, if the initial loading point was feasible, it is reasonable that the percent solve success is high.

TABLE II
PERCENT ERROR FOR THE PROPOSED METHOD AND FOR DC OPF ACROSS THE 500-SAMPLE TESTING SETS

Network	Average % Error (NN)	Average % Error (DC)	Worst % Error (NN)	Worst % Error (DC)
30-bus	0.20%	1.46%	0.31%	1.82%
500-bus	2.70%	5.30%	10.12%	16.22%
1354-bus	0.09%	1.40%	0.20%	1.73%

TABLE III
PERCENT ERROR FOR THE PROPOSED METHOD AND FOR DC OPF WITH AN AC PF ACROSS THE 500-SAMPLE TESTING SETS

Network	Average % Error (NN)	Average % Error (DC w/PF)	Worst % Error (NN)	Worst % Error (DC w/PF)
30-bus	0.20%	0.26%	0.31%	0.37%
500-bus	2.70%	29.13%	10.12%	35.34%
1354-bus	0.09%	1.87%	0.20%	2.22%

C. Error of the Learning-Based Solution Vs. MIPS

Tables II and III show the difference in cost for the learning-based solution ("NN") and DC OPF without and with an AC power flow (respectively), calculated as follows:

$$\%\,\text{Error} = \left|\frac{f(\mathbf{x}^*) - f(\tilde{\mathbf{x}})}{f(\mathbf{x}^*)}\right| \cdot 100\%. \tag{2}$$

Where $f(\cdot)$ is the AC OPF objective function, and $\mathbf{x}^*$ is the solution from MIPS, and $\tilde{\mathbf{x}}$ is the solution from the method being compared (i.e., the learning based or the DC solution). The equation in (2) is akin to the definition of "optimality gap"; however, we avoid using this specific term here because there is no guarantee that $f(\mathbf{x}^*)$ is the globally optimal solution. Other strategies, such as comparing the learning-based method with relaxations such as the second-order cone programming (SOCP) relaxation, could also provide indications of possible optimality gap.

In the results, the DC OPF solution with and without AC PF post processing is compared against the AC OPF solution provided by MIPS. This is a key comparison because in many markets, DC OPF is used to calculate prices; thus, it is currently deemed an acceptable way of approximating the cost of network operation. As the table shows, however, the learning-based method produces even lower operating costs than DC OPF, on average and in the worst case throughout the testing set. Surprisingly, the solution obtained with the AC PF post-processing step, while now AC feasible, were found to overall increase the cost difference with the AC OPF solution. This may be due to the fact that when a power flow is run, all generator values except the slack bus generator are fixed to the given values. This means that the slack generator must account for losses to obtain AC feasibility, and this is often not the most optimal solution.

IV. COMPARISON WITH OTHER POWER FLOW MODELS

Even though DC OPF is used in many existing network operations, it is useful to compare the proposed method against other convexifications. In this section, the learning-based method is

Authorized licensed use limited to: UNIVERSITY OF COLORADO. Downloaded on January 01,2023 at 18:41:44 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
PERCENT ERROR FOR THE PROPOSED METHOD AND FOR SOCP ACROSS THE 500-SAMPLE TESTING SETS

Network	Average Solve Time (NN)	Average Solve Time (SOCP)	Average Error (NN)	Average Error (SOCP)
30-bus	0.05 s	0.15 s	0.20%	2.86%
500-bus	0.12 s	3.48 s	2.70%	26.85%
1354-bus	0.25 s	33.72 s	0.09%	0.28%

compared against the popular SOCP relaxation in terms of speed and accuracy. The particular implementation that will be used for the comparison is that found in the `PowerModels` package [9] for SOCP [10]. Table IV shows a side-by-side comparison of solving SOCP using IPOPT in Julia compared to the learning-based method, across the same testing set. While the learning based method appears to be faster and more accurate, it is important to note that unlike SOCP, it does not provide a theoretical lower bound for the true optimum. Additionally, an AC power flow could be used as a post-processing step for SOCP, akin to the previous analysis of running a DC OPF with an AC PF post-processing step. Note that other relaxations could be considered for comparison in future work; for example, the semidefinite programming relaxation (SDP). As [11] mentions, the SDP relaxation is generally significantly slower than SOCP (in our simulations, the given computational platform was unable to solve SDP for the largest considered network), but provides a tighter relaxation. Tradeoffs are dependent on the application - the application here focuses on real time AC OPF.

V. CONCLUSION AND DISCUSSION

This paper provided a learning-based approximation for solving AC optimal power flow. Promising initial results indicate that the method can achieve fast convergence speeds and high accuracy. However, there are multiple drawbacks that should be discussed here; although these results show an effective speed gain, traditional optimization has multiple upsides. First, the model is trained on one static configuration of the network. That means that for any transformer tap-changing, any line switching, or switchable shunt control for example, a different model would have to be trained. Second, the performance of the model is limited by the dataset from which it was trained on. Lastly, neural networks do not offer grid operators as much insight or confidence in the resulting decision-making as traditional optimization does. This heuristic also does not provide a theoretical lower bound.

Data Availability: The datasets and code used in this paper may be obtained by emailing the author.

REFERENCES

[1] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," in *Proc. IEEE SmartGridComm*, Dec. 2020, pp. 1–6.
[2] M. Chatzos, F. Fioretto, T. W. K. Mak, and P. V. Hentenryck, "High-fidelity machine learning approximations of large-scale optimal power flow," 2020, *arXiv:2006.16356*.
[3] X. Pan, T. Zhao, and M. Chen, "DeepOPF: Deep Neural Network for DC Optimal Power Flow," in *Proc. 10th IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (IEEE SmartGridComm 2019)*, Oct. 2019.
[4] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.
[5] S. Fliscounakis, P. Panciatici, F. Capitanescu, and L. Wehenkel, "Contingency ranking with respect to overloads in very large power systems taking into account uncertainty, preventive, and corrective actions," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4909–4917, Nov. 2013.
[6] K. Baker, "Learning warm-start points for AC optimal power flow," in *Proc. IEEE Mach. Learn. Signal Proc. Conf.*, 2019, pp. 1–6.
[7] L. Chen and J. E. Tate, "Hot-starting the AC power flow with convolutional neural networks," 2020, *arXiv:2004.09342*.
[8] S. Cvijic, P. Feldmann, and M. Ilic, "Applications of homotopy for solving AC power flow and AC optimal power flow," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, 2012, pp. 1–8.
[9] C. Coffrin, R. Bent, K. Sundar, Y. Ng, and M. Lubin, "PowerModels. JL: An open-source framework for exploring power flow formulations," in *Proc. Power Syst. Computation Conf.*, 2018, pp. 1–8.
[10] R. A. Jabr, "Radial distribution load flow using conic programming," *IEEE Trans. Power Syst.*, vol. 21, no. 3, pp. 1458–1459, Aug. 2006.
[11] C. Coffrin, H. L. Hijazi, and P. Van Hentenryck, "The QC relaxation: A theoretical and computational study on optimal power flow," *IEEE Trans. Power Syst.*, vol. 31, no. 4, pp. 3008–3018, Jul. 2016.

Authorized licensed use limited to: UNIVERSITY OF COLORADO. Downloaded on January 01,2023 at 18:41:44 UTC from IEEE Xplore. Restrictions apply.