An adversarially robust approach to security-constrained optimal power flow

Neeraj Vijay Bedmutha
Carnegie Mellon University
nbedmuth@andrew.cmu.edu

Priya L. Donti
Carnegie Mellon University
pdonti@cs.cmu.edu

J. Zico Kolter
Carnegie Mellon University
Bosch Center for AI
zkolter@cs.cmu.edu

Abstract

Security-constrained optimal power flow (SCOPF) is a critical problem for the operation of power systems, aiming to schedule power generation in a way that is robust to potential equipment failures. However, many SCOPF approaches require constructing large optimization problems that explicitly account for each of these potential system failures, thus suffering from computational complexity issues that limit their use in practice. In this paper, we propose an approach to solving SCOPF inspired by adversarially robust training in neural networks. In particular, we frame SCOPF as a bi-level optimization problem – viewing power generation settings as parameters associated with a neural network defender, and equipment failures as (adversarial) attacks – and solve this problem via gradient-based techniques. We describe the results of initial experiments on a 30-bus test system.

1 Introduction

One of the most crucial problems in power systems engineering involves figuring out ways to schedule electricity generation in an efficient and robust manner. Specifically, power system operators must determine the amount of power each generator on the grid should produce in order to minimize power generation costs, satisfy the physical constraints on the power grid, and ensure robustness to any potential outages of power system equipment (such as power generators or lines). A common approach in this area of *security-constrained optimal power flow* (SCOPF) involves trying to explicitly solve a large optimization problem that enumerates all possible equipment failures (or *contingencies*), as well as accounting for power generation costs. However, these optimization problems tend to be large, requiring massive computing hardware to solve. As a result, power system operators tend to use cheap, inaccurate proxies to this problem instead, leading to large system inefficiencies in practice.

In this paper, we attempt to address this problem via an approach based on adversarial robustness in neural networks. Our approach involves viewing power system failures or contingencies as norm-bounded adversarial attacks, and attempts to "defend" against these attacks by adjusting the amount of power generation at each generator. In particular, for any given setting of power generation, we employ gradient-based techniques to find the "worst-case" failure for that setting. We then adjust our power generation setting by taking a gradient step in a direction that increases robustness with respect to this worst-case failure. By doing so, our method can efficiently learn to be robust to all potential attacks, without ever having to explicitly enumerate them.

We describe our approach in the context of $N-1$ SCOPF, a common setting that assumes at most one contingency at a time will occur on the power grid. (While our framework is also applicable to more general settings, we begin with this setting as an initial demonstration.) We describe the results of our initial SCOPF experiments on a 30-bus test case.

2 Related work

Our work addresses the problem of SCOPF using concepts from adversarial robustness and implicit layers in neural networks. We briefly describe these areas and their relation to the present work.

SCOPF. In power systems analysis, it is extremely important to obtain fast solvers for power system optimization problems and have the ability to simulate the most important contingencies [1]. However, the general SCOPF problem can be extremely computationally intensive for large systems or grid networks consisting of several thousand buses [2]. Approaches to reduce the size of the problem rely on reducing the number of contingencies to be included in the SCOPF analysis; for instance, [2] discusses simplified methods such as Bender's Decomposition Method and Linearization of Post Contingency Constraints, which can lead to a sub-optimal analysis of the violations and typically may need re-solving. Other approaches have tried to speed up SCOPF not by reducing the number of contingencies analyzed, but instead by using deep neural networks, e.g., by training a load-generation mapping and then predicting the generation for the system in question using load inputs [3]. However, such methods often rely on a large amount of existing load and generation data, and may have trouble generalizing to out-of-distribution samples. Our method draws inspiration from deep neural network approaches, but incorporates system knowledge (e.g., the power flow equations and knowledge of the full space of contingencies) to approach the problem of SCOPF.

Adversarial robustness. The area of adversarial robustness in deep learning attempts to address the vulnerability of neural networks to attacks. In particular, adversarially robust training approaches pick neural network parameters using a bi-level optimization approach, in which an inner maximization problem characterizes the worst case attack on a neural network with particular parameters, and an outer minimization problem then attempts to pick parameters that are robust to this attack [4]. In this work, we draw inspiration from the adversarial robustness literature by viewing power system setpoints as akin to neural network parameters, and power system failures as akin to neural network attacks. In particular, we frame SCOPF as a bilevel optimization problem, and solve it (as in the literature on adversarial robustness) using gradient-based techniques.

Implicit layers. There has recently been a growing interest in the creation of neural network layers that characterize implicit functions of their inputs and outputs. Examples include quadratic programming layers [5], layers characterizing ordinary differential equations [6], and power system optimization layers [7], among many others [8–17]. Our approach makes use of concepts from this literature, in particular when embedding the calculation of power flow solutions into neural networks. In particular, while power flow solutions can be calculated explicitly for certain classes of power system approximations, in the more general case, calculating power flow solutions involves employing implicit solution techniques (such as Newton-Raphson). While we use the DC approximation of power flow here (which is amenable to explicit solutions), in future work, we propose to study more general settings, and employ implicit power flow layers [7] in the loop of our proposed method.

3 Background: SC DCOPF

Optimal power flow problems are solved by power system operators in order to determine set points for electricity generators that minimize the cost of supplying power. In particular, the problem of AC optimal power flow (ACOPF) attempts to set the power generation and voltage for each generator to minimize costs subject to equipment limits and a set of (nonlinear, non-convex) power flow equations. As ACOPF is non-convex and expensive to solve, it is common to instead use an approximation called DC optimal power flow (DCOPF) that can be used for fast assessment of power systems.

At a high level, for a system with n buses (i.e., nodes), n_g generators, and some cost function $f_c : \mathbb{R}^{n_g} \to \mathbb{R}$, DCOPF tries to determine the power generation $P_g \in \mathbb{R}^{n_g}$ at each generator by solving the problem

$$\begin{aligned}
\underset{P_g \in \mathbb{R}^{n_g}}{\text{minimize}} \quad & f_c(P_g) && (1)\\
\text{subject to} \quad & P_{\text{min}} \leq P_g \leq P_{\text{max}} && (2)\\
& \tilde{P}_g - P_d = B \times \theta && (3)\\
& -(L_{\text{max}})_{ij} \leq \frac{1}{x_{ij}}(\theta_i - \theta_j) \leq (L_{\text{max}})_{ij} \;\; \forall \text{ lines } (i,j), && (4)
\end{aligned}$$

where $P_d \in \mathbb{R}^n$ describes the electricity demand at each bus, $\tilde{P}_g \in \mathbb{R}^n$ is a vector that is 0 at non-generator buses and equals the total power generation at that bus (based on P_g) at generator buses, $\theta \in \mathbb{R}^n$ is a vector of nodal phase angles, x_{ij} is the *susceptance* of line (i, j), $B \in \mathbb{R}^{n \times n}$ is the *nodal susceptance matrix*, and $P_{\text{min}}, P_{\text{max}}, L_{\text{max}}$ describe generator and line equipment limits.

In this paper, we specifically address the problem of DCOPF with security constraints (SC DCOPF). Security constrained DCOPF aims to both (a) reduce the total cost of generator power output, as in DCOPF, and (b) handle potential system failures (e.g., generator or line breakdowns). Here, we specifically consider the common case of "$N-1$ security constrained DCOPF," i.e., optimal power flow analysis that can account for the failure of any *one* system generator or line at a time. In other words, the goal of $N-1$ SC DCOPF is to obtain a power flow solution that continues to respect the equipment limits of power generators and lines, as well as the DC power flow equations, in the case that any one piece of system equipment should fail.

4 Adversarially robust SCOPF

In this paper, we view SC DCOPF through the lens of adversarial robustness in deep learning. Specifically, SC DCOPF can be written as the bi-level optimization problem

$$\underset{P_g \in \mathbb{R}^{n_g}}{\text{minimize}} \; \underset{\alpha \in \mathbb{R}^{n_c}}{\text{maximize}} \; \ell(P_g, \alpha) + \ell(P_g, 0), \tag{5}$$

where $\ell(\cdot, \cdot)$ is some loss function capturing both power costs and the feasibility of the solution, and $\alpha \in \mathbb{R}^{n_c}$ is a vector of contingencies. (We will describe the details of both of these aspects shortly.) This bi-level optimization formulation captures the fact that we would like our power generation setpoint P_g to be robust both in the case where there are no contingencies, and in the case where we experience a *worst-case* contingency/failure that maximizes the inner maximization problem. In the parlance of adversarially robust neural networks, we view P_g as the parameters of the defender, and α as the actions of an adversarial attacker.

Following the literature in adversarially robust training in neural networks, our approach is to then optimize this problem via gradient-based techniques. At a high-level, this approach is as follows:

- Initialize some $P_g = P_g^{\text{init}}$.
- Until convergence
  - Solve the inner maximization problem to find an argmax $\alpha^\star$ (i.e., the worst-case failure) for the current setting of P_g. (This can be done either in closed-form or via gradient-based techniques.)
  - Calculate the loss $\ell(P_g, \alpha^\star) + \ell(P_g, 0)$.
  - Update $P_g = P_g - \beta \nabla_{P_g}(\ell(P_g, \alpha^\star) + \ell(P_g, 0))$, for some learning rate β.

We dive into the details of the algorithm and lay out various intricacies specific to this method.

4.1 Defining contingencies and outages

We start by defining the attack space (contingency space) used in our setting. In standard SC DCOPF analysis, equipment failure is viewed as a discrete event, i.e., a piece of equipment either fails or does not fail. In order to optimize Equation (5) via gradient-based techniques, however, we instead define a continuous contingency space over which to optimize α. Specifically, we define $\mathrm{A} \subseteq \mathbb{R}^{n_c}$ as a relaxed contingency space, where for any vector of n_c contingencies $\alpha \in \mathrm{A}$,

$$\alpha_i = \begin{cases} 0 & \text{iff contingency } i \text{ is not active,} \\ 1 & \text{iff contingency } i \text{ is fully active,} \\ f \in (0,1) & \text{iff contingency } i \text{ is partially active (see description below).} \end{cases} \tag{6}$$

The first two notions presented in Equation (6) as to whether a contingency is active or not are as standard in power systems: the generator/branch described by contingency i functions as normal if the contingency is not active, and suffers a complete outage if the contingency is fully active. We newly define the notion of a fractional contingency as one in which a generator or branch may suffer a partial outage. Specifically, a contingency of strength $f \in (0, 1)$ denotes a reduction in the capacity of the relevant component by a fraction $(1 - f)$.

Since α is a vector of contingencies over both the generators and the lines, for notational convenience, we define it as

$$\alpha = \begin{bmatrix} \Phi \\ \Psi \end{bmatrix}, \tag{7}$$

where Φ is the vector of generator contingencies and Ψ is the vector of line contingencies.

In this setting, we further distinguish between the notion of a *contingency* and an *outage*. In particular, we qualify any contingency α_i as an outage if and only if it reduces the capacity of a generator or line below the amount of existing power generation or line flow, respectively, associated with that line – in other words, if the contingency makes the operation of that component infeasible. We define a power system component as being in an infeasible region if it experiences an outage. (In contrast, we say a component is in the feasible region if the contingency on it does not translate to an outage.) We now describe our model for outages in more detail.

4.1.1 Generator Outage

In the presence of contingencies, the system might incur generator outages. As discussed before, outages occur when a contingency reduces the capacity of a generator below its current power generation. We consider the setting of automatic generation control (AGC), where working power generators adjust their power generation to compensate for power outages on the grid.

Specifically, for some set of outages, we note that the total reduction in generation after outages on some generators must be met by all of the other (non-outage) generators. In other words, if n_{out} generators undergo an outage, then we let

$$P_{\text{outage}} = \sum_{\mathscr{o}=1}^{n_{\text{out}}} (\Delta P_{\mathscr{o}}), \tag{8}$$

where $\Delta P_{\mathscr{o}}$ is the power outage for a particular generator and P_{outage} is the total lost power generation due to the partial power outage contingencies in generators. In particular, for a generator $\mathscr{o}$ experiencing an outage, we refer to its original generation as $(P_g)_{\mathscr{o}}$ and the generator capacity as $(P_{\max})_{\mathscr{o}}$. Then, in case of a contingency $\Phi_{\mathscr{o}}$ on this generator, the change in power for this generator is given by:

$$\Delta P_{\mathscr{o}} = -\max\big((P_g)_{\mathscr{o}} - (P_{\max})_{\mathscr{o}} \times (1 - \Phi_{\mathscr{o}}), 0\big). \tag{9}$$

The pickup ΔP_w by any working generator w of the remaining n_g - n_{out} working generators is given to us by a participation factor PF_w. Specifically, for a working generator w, we let

$$\Delta P_w = P_{\text{outage}} \times PF_w, \text{ where } PF_w = \frac{(P_{\max})_w}{\sum_w (P_{\max})_w}. \tag{10}$$

This formulation ensures that all of the power lost (P_{outage}) is picked up by the $n_g - n_{\text{out}}$ working generators put together. The new power output at each generator j is then given by $(P_g)_j + \Delta P_j$, where ΔP_j is given by Equation (9) for outage generators, and by Equation (10) for working generators. In vectorized form, we will refer to the adjusted power generations at each generator by $P_g + \Delta P$.

4.1.2 Line Outage

The next step in the algorithm is to evaluate potential line violations. The new power generations $P_g + \Delta P$ are used in a DC power flow (which solves the power flow equations (3)) to solve for the power flows along each line. Let L denote this set of line flows. We note that the line flows may very well be infeasible depending on the line contingencies Ψ, as reflected in our loss function.

4.2 Loss Function

The loss function in the bi-level optimization problem (5) is formulated to (a) evaluate the power generation costs of the power system, (b) penalize outages or infeasibilities, At a high level, the loss function can be expressed in terms of its components as:

$$\ell(P_g, \alpha) = f_c(P_g, \alpha) + f_{\text{infeas}}(P_g, \alpha), \tag{11}$$

where again P_g is the vector of current power generation, α is the contingency vector, $f_c(P_g, \alpha)$ is a function representing the power generation costs, and $f_{\text{infeas}}(P_g, \alpha)$ represents infeasibilities in power generation or line flows.

More specifically, we define f_{infeas} so as to represent both (a) true infeasibilities (i.e., when generator power outputs or line flows go beyond their respective generation or line flow limits), and (b) how close a particular generator or line is to its limit in the case that it is feasible. In particular, incorporating the latter enables us to design a continuous loss function that is amenable to gradient descent-based techniques, as follows:

$$f_{\text{infeas}}(P_g, \alpha) = \left((P_g + \Delta P) - \frac{P_{\min} + P_{\max} \circ (1 - \Phi)}{2} \right)^2 + \left(|L| - \frac{L_{\max} \circ (1 - \Psi)}{2} \right)^2. \quad (12)$$

We note that this loss function includes L, which is the solution to power flow on the adjusted power generation values $P_g + \Delta P$. In the case of DC power flow, this simply entails solving a set of linear equations, and therefore is easy to differentiate through. In the case of AC power flow, which is required for SC ACOPF (which we are exploring in concurrent work), the power flow equations are nonlinear and non-convex, and therefore must be solved via implicit methods such as Newton's method. In this case, differentiating through these equations requires implicit differentiation techniques, e.g., as described in [7].

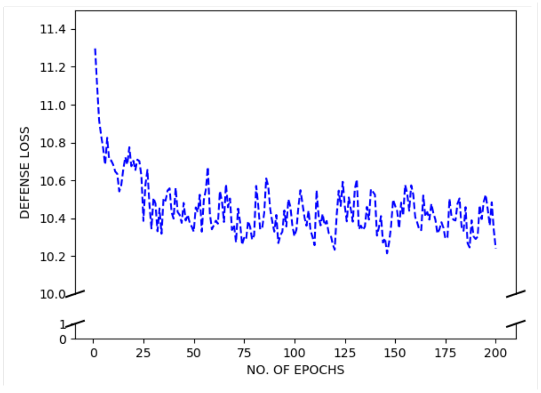
5 Preliminary results

We train the proposed adversarial neural network on a 30-bus test case file for 200 "defense epochs" (i.e., 200 iterations of the outer minimization in Equation (5)) to obtain the preliminary results. The expected goal is to reduce the loss (i.e., power costs and infeasibilities) over time by adjusting the control variable P_g in every outer iteration, after having found the worst-case vector of contingencies α via the inner maximization. Figure 1 represents the plotted results describing (a) the power costs f_c, (b) the penalty contributions f_{infeas} of generator and line infeasibilities, and (c) sum of these two components over training epochs.

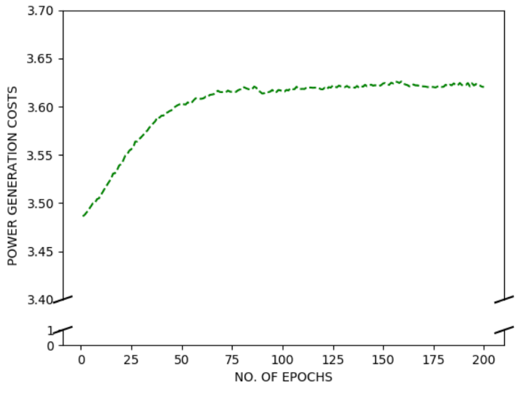
Figure 1a shows the total loss over epochs, which (as expected) our algorithm reduces. Decomposing this into its components, Figure 1b depicts the power generation cost for the 30 bus test case system, whereas Figure 1c shows the infeasibility contributions to the loss function. The results from these plots show that the power generation costs increase over the epochs, whereas infeasibilities go down. In particular, these results demonstrate that there is a trade-off between power costs and the robustness of the obtained solutions.

6 Conclusion

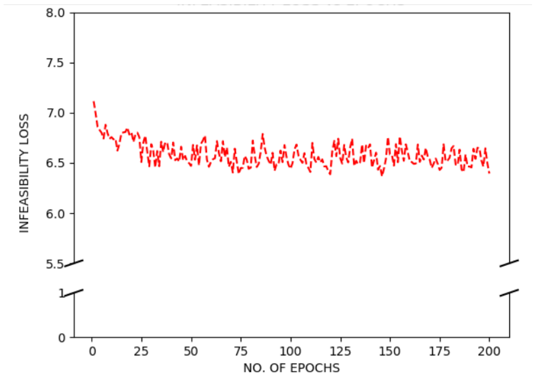
In this work, we propose a framework for SCOPF based on adversarially robust training in neural networks. We investigate this approach in an $N-1$ SC DCOPF setting, via experiments on a 30-bus test case. Future work includes further tests in DC settings, as well as extending our approach to the SC ACOPF setting (which will involve drawing on the literature in implicit differentiation in neural networks).



(a) Total loss vs. defense epochs



(b) Cost of power generation vs. defense epochs



(c) Infeasibility loss vs. defense epochs

Figure 1: Preliminary results for the 30-bus test case. We observe that the infeasibility losses decrease as we train the network, whereas power generation costs rise. Basically, we are trading off some sense between "optimality" in the power costs and a sense of robustness to attacks.

References

[1] V. H. Hinojosa and F. Gonzalez-Longatt, "Preventive security-constrained dcopf formulation using power transmission distribution factors and line outage distribution factors," *Energies*, vol. 11, no. 6, p. 1497, 2018.

[2] F. Capitanescu, J. M. Ramos, P. Panciatici, D. Kirschen, A. M. Marcolini, L. Platbrood, and L. Wehenkel, "State-of-the-art, challenges, and future trends in security constrained optimal power flow," *Electric Power Systems Research*, vol. 81, no. 8, pp. 1731–1741, 2011.

[3] X. Pan, T. Zhao, and M. Chen, "Deepopf: A deep neural network approach for security-constrained dc optimal power flow," *arXiv preprint arXiv:1910.14448*, 2019.

[4] Z. Kolter and A. Madry, "Tutorial: Adversarial robustness - theory and practice."

[5] B. Amos and J. Z. Kolter, "OptNet: Differentiable Optimization as a Layer in Neural Networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 136–145, JMLR. org, 2017.

[6] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[7] P. L. Donti, I. L. Azevedo, and J. Z. Kolter, "Inverse optimal power flow: Assessing the vulnerability of power grid data,"

[8] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[9] S. Gould, R. Hartley, and D. Campbell, "Deep declarative networks: A new hope," *Preprint arXiv:1909.04866*, 2019.

[10] P. L. Donti, B. Amos, and J. Z. Kolter, “Task-based end-to-end model learning in stochastic optimization,” *Preprint arXiv:1703.04529*, 2017.

[11] J. Djolonga and A. Krause, “Differentiable learning of submodular models,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[12] S. Tschiatschek, A. Sahin, and A. Krause, “Differentiable submodular maximization,” *Preprint arXiv:1803.01785*, 2018.

[13] B. Wilder, B. Dilkina, and M. Tambe, “Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization,” in *AAAI Conference on Artificial Intelligence*, 2018.

[14] P.-W. Wang, P. L. Donti, B. Wilder, and Z. Kolter, “SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver,” in *International Conference on Machine Learning (ICML)*, 2019.

[15] C. K. Ling, F. Fang, and J. Z. Kolter, “What game are we playing? End-to-end learning in normal and extensive form games,” *Preprint arXiv:1805.02777*, 2018.

[16] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, “End-to-end differentiable physics for learning and control,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[17] S. Bai, J. Z. Kolter, and V. Koltun, “Deep equilibrium models,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.