Fast Security-Constrained Optimal Power Flow with Time-Aware Critical Contingency Prediction

Reza Khalili
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, NY 11791, USA
reza.khalili@stonybrook.edu

Yue Zhao
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, NY 11791, USA
yue.zhao.2@stonybrook.edu

***Abstract*—Accelerating the solution process of preventive Security Constrained Optimal Power Flow (SCOPF) is studied. In an iterative algorithmic framework, knowledge of the relatively sparse critical contingencies can greatly reduce the problem size and hence solution time. Predictors that can predict the critical contingencies that must be included in the SCOPF formulation are trained and integrated into an iterative algorithm. As different types of prediction errors — false negatives and false positives — have markedly different impact on algorithm solution time, a novel time-aware loss function is designed and calibrated for training predictors that directly minimizes algorithm run-time. A multi-objective loss function that incorporates both this time-aware loss and the accuracy promoting binary cross entropy (BCE) loss is then designed and tuned. Comprehensive evaluation of the time-aware predictor-assisted iterative algorithm is conducted based on the IEEE 118-bus system. Effective re-balancing of false negatives and false positives is observed, and significant reduction of algorithm run-time is achieved with the developed time-aware predictors.**

***Index Terms*—SCOPF, Contingency Analysis, Run-time, Multi-Objective Optimization, Learning-Accelerated Optimization**

I. Introduction

As the global expansion of renewable energy sources and the electrification of energy demands continue, reliability has become an even more critical objective for power system operations. To ensure reliable and secure operations of power systems at all times, a key problem that needs to be solved quickly and repeatedly is the Security Constrained Optimal Power Flow (SCOPF) problem. Solving large-scale SCOPF is however a computationally very heavy task. Indeed, significant recent research efforts have been devoted to solving SCOPF efficiently [1]. Moreover, the rapidly advancing computational resources world-wide offer great promises for accelerating SCOPF, both from optimization and machine learning perspectives. In particular, machine learning techniques could reduce the size of the optimization problem of SCOPF by effectively predicting and hence eliminating unnecessary constraints. The high level idea of this work is to incorporate such predictive capability with efficient optimization algorithms so that overall solution run-time can be effectively reduced.

A. Literature Review

For SCOPF, there are two types of security constraints: corrective security and preventive security, both of which involve solving large-scale optimization problems. For corrective security, in [2], the authors highlighted the importance of solving these optimization problems within the required time frame and implemented corrective post-contingency assessments. In [3], the focus was on system re-dispatch and load shedding as corrective measures to maintain system security. The authors proposed a method for recalculating pre-contingency matrices, which significantly reduced computation time. Another study, also centered on corrective solutions, employed an approximate DC-OPF to generate initial results, followed by the application of tighter constraints to improve accuracy [4]. In [5], sparse optimization techniques and decomposition algorithms were used to develop a tractable model.

In the realm of preventive SCOPF, [6] applied decomposition techniques and quadratic convex relaxation to solve SCOPF by dividing the large-scale problem into more manageable subproblems. Notably, leveraging the potential of Artificial Intelligence (AI) can further reduce optimization time. In [7], the authors proposed physics-aware fast learning and prediction of active constraints of OPF that greatly reduced the optimization time. [8] utilized Deep Neural Networks (DNN) to predict the decision variables of the optimization problem, subsequently enforcing feasibility. Moreover, the use of different loss functions can lead to improved results by aligning the model more closely with the required criteria. In [9], a two-stage approach was adopted, where primal-dual learning was employed to obtain near-optimal solutions within milliseconds. Their loss function incorporated both operational cost and penalties for contingency constraint violations. [10] framed the problem as constraint-satisfying training for Deep Reinforcement Learning (DRL), where actor gradients were approximated by solving the KKT conditions of the Lagrangian. DNNs were also used for initialization.

However, AI-based techniques can potentially compromise system security. Addressing this concern, [11] examined the vulnerabilities associated with AI applications in power systems and proposed a more secure DRL-based SCOPF framework. In line with [8], [12] used neural networks to predict the decision variables of the SCOPF problem and validated the feasibility of the output. They also designed a modified loss function based on operational cost and constraint violations, tailored to the specific needs of the problem. In [13], the authors integrated a neural network layer into an

This work is supported by the National Science Foundation under Grant ECCS-2025152.

iterative algorithm to predict active constraints of SCOPF. Notably, however, existing studies have not directly focused on optimization *time* as their design objective.

B. Research Gap & Contributions

Given the importance of preventive SCOPF and the need to solve it in a timely manner, iterative algorithms have proved to be an effective approach. Adapting AI tools that are better integrated with such optimization algorithms could greatly reduce optimization time. In this context, a learning objective directly focusing on optimization time could be particularly advantageous, which has however not been thoroughly explored in the literature. The main contributions of this paper are as follows. We start with an iterative algorithm enhanced with a contingency screening procedure for efficiently solving SCOPF with N-1 contingencies. We then integrate a predictor of *critical contingencies* into the iterative algorithm, trained based on a Binary Cross Entropy (BCE) loss function for maximizing prediction accuracy. Next, we design a novel *time-aware loss function* that directly approximates the overall run-time of the predictor assisted iterative algorithm. This loss function is then integrated with BCE to form a multi-objective loss function. Comprehensive performance evaluation is conducted and demonstrates that the developed time-aware predictor-assisted iterative algorithm significantly improves the overall algorithm run-time.

II. Problem Description

A. Security Constrained Optimal Power Flow

We consider the preventive SCOPF problem as follows. The objective function represents the cost of generators' dispatch. Three types of constraints are considered: a) the constraints related to generators' production limits; b) the constraints related to the "baseline" OPF without considering security measures (i.e., contingencies); and c) the constraints related to security measures. In this work, we consider $N-1$ contingencies of the transmission lines, and DC-OPF is employed as the problem model. As such, the mathematical formulation of an SCOPF is as follows:

$$\min_{G,\theta,\{\theta^c\}} \sum_g C_g(G) \tag{1}$$

$$\text{s.t. } 1^T G = 1^T d, \tag{2}$$

$$g_{\min} \le G \le g_{\max}, \tag{3}$$

$$-F_{\max} \le F \le F_{\max}, \tag{4}$$

$$f(G, d, F, \theta) = 0, \tag{5}$$

$$-F'_{\max} \le F^c \le F'_{\max}, \quad \text{for all } c \in C, \tag{6}$$

$$f^c(G, d, F^c, \theta^c) = 0, \quad \text{for all } c \in C. \tag{7}$$

Here, $C_g(G)$ represents the cost function of generator g, and G denotes the power outputs of all the generators. The equality constraint (2) ensures power balance, where d is the demand vector. Generator limits are enforced in (3). Constraints (4) and (6) maintain transmission line security: F_{ij} represents power flow on line ij under normal conditions; F^c_{ij} denotes post-contingency power flows for contingency c in a set of line outage contingencies C; $F_{\max}$ and $F'_{\max}$ are the corresponding thermal limits. (5) and (7) are the power flow equations that relate nodal power injections, line flows, and nodal voltage angles. We let the set of line outage contingencies C include all the single line outages that preserve the connectivity of the entire system. While we employ quadratic generator cost functions in our study, the developed methods apply to general cost functions.

B. Iterative Algorithm for Solving SCOPF

Conventionally, the SCOPF problem can be solved as a one-shot full optimization (Eqs. (1)–(7)), incorporating *all* the contingency constraints from C. Notably, the total number of contingency constraints (6) (7) grows quadratically with the size of the system. This could limit the practical feasibility of this solution process especially under stringent time requirement.

Alternatively, if *which contingency constraints are eventually active* is known beforehand, one can simply include the active constraints in the problem formulation, thereby greatly reducing the size and solution time of the problem. In practice, while one cannot assume the availability of such knowledge, this indeed inspires an *iterative* algorithm that utilizes iterative screenings to include only the critical constraints. Such an algorithm is presented in Fig.1-a. Specifically, the set of contingencies to consider, C, is initialized as an empty set. In the first iteration, the problem is solved considering only the baseline case Eqs. (1)-(5), i.e, without any line outage contingencies. Based on the corresponding dispatch solution, power flow (PF) is computed for all the $N-1$ contingencies: notably, each line outage would result in a modified network topology, leading to a new PF solution. These PF results are then utilized to identify any violated constraints in all the contingencies, and the potentially "critical" contingencies are thereby indicated by the violations and added to the set C for the next iteration of the solution. This process is repeated until no violations are detected any more. As will be demonstrated later in Section IV-B, such iterative algorithms are indeed significantly faster than solving a one-shot SCOPF with all the $N-1$ contingencies included in the constraints.

III. Proposed Method

A. Predictor-Assisted Iterative Algorithm

A key idea for accelerating the solution process of the SCOPF problem is to obtain a predictor that can effectively predicts the active contingency constraints of the optimal solution. In this work, we focus on predicting the *critical contingencies* among all the $N-1$ contingencies: For a contingency c, if *any* of the inequality constraints in (6) is active with the optimal solution, it is a critical contingency. Such a predictor can then be employed as the initialization of the contingency set C in the iterative algorithm: intuitively, an accurate predictor can thus further improve the efficiency of the iterative algorithm significantly. This predictor-assisted iterative algorithm is presented in Fig. 1-b.

Notably, if the predictor outputs *completely cover* all the critical contingencies, meaning that there is *no false negative*, the iterative algorithm will end after just one iteration. There

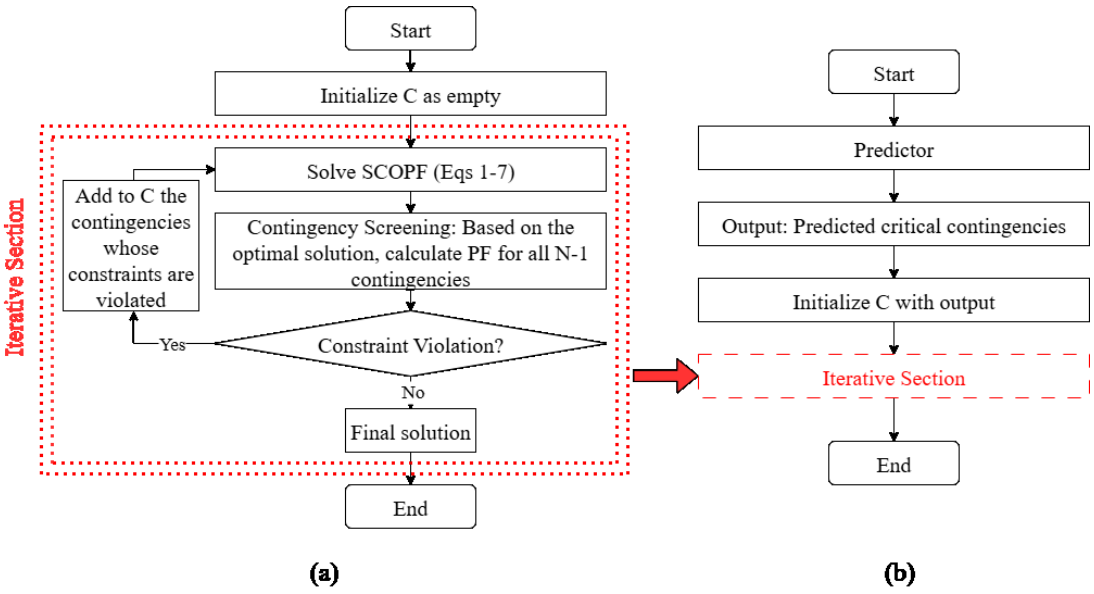

Fig. 1. Flowcharts of: a) iterative algorithm b) predictor-assisted algorithm.

are thus two different mechanisms by which prediction errors could lead to solution time inefficiencies. On the one hand, if any constraint violation is still detected after one iteration due to *false negatives*, the algorithm would need to run at least one *additional iteration* which would lead to significant additional solution time. On the other hand, having *false positives* would also lead to additional solution time due to the (unnecessarily) *increased size of the problem* to solve. For example, one can of course try to cover all the critical contingencies by including a large number of them (the extreme case being including the entire set C), but such an approach would clearly lead to significant greater solution time per iteration.

B. Binary Cross Entropy Loss for Maximizing Accuracy

The predictor we aim to train a) takes the system's nodal load profile as the input, and b) outputs a binary vector indicating each contingency as critical or not critical. We begin with training predictors to achieve high *accuracies*. For this purpose, we employ the Binary Cross Entropy (BCE) loss:

$$L_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(P_i) + (1-y_i)\log(1-P_i)] \tag{8}$$

In this loss function, N represents the number of training samples, y_i is the ground truth binary label, and P_i is the predicted probability. As the predictor has multiple binary outputs (each for one contingency), the BCE losses are averaged across all outputs.

C. Time-aware loss for minimizing algorithm run-time

While BCE as the loss function promotes the highest prediction accuracy, it assigns equal weights to false positives and false negatives. As noted above, there are however important nuances on how prediction errors impact solution time which are not captured by BCE. As shown in Fig. 1-a and Fig. 1-b, if a false negative occurs, the contingency screening step — which detects all violated constraints and hence the false negatives — will trigger an additional iteration. In contrast, a false positive would also increase the optimization time although it does not trigger an extra iteration. As such, *we will develop a predictor loss function that aims to approximate the solution time, directly incorporating the different impact of ground truths, false positives and false negatives on the run-time of the predictor-assisted iterative algorithm.*

Specifically, we define the loss function as the difference between the optimization time with ground truths (i.e., perfect predictions) and that with the predictions potentially with errors, modeled as follows:

$$\begin{aligned}\Delta T =& T_p - T_{gt} = [\alpha_1 N_p + \beta_1 - \alpha_1 N_{gt} - \beta_1] + [T_s - T_s] \\ &+ I_{fn}[\alpha_2(N_p + N_{fn}) + \beta_2 + T_s] \qquad (9)\\ =& [\alpha_1(N_p - N_{gt})] \\ &+ I_{fn}[\alpha_2(N_p + N_{fn}) + \beta_2 + T_s] \qquad (10)\end{aligned}$$

Here, T_p denotes the optimization time based on the predictor outputs, and T_{gt} represents the optimization time with ground truths/perfect prediction. N_p is the number of predicted critical contingencies; N_{gt} is the ground truth's number of critical contingencies; N_{fn} is the number of false negatives in the current iteration (to be included in the next iteration). The solution time of each iteration's SCOPF is modeled as a linear function of the total number of positives (true and false) with parameters α and β. T_s denotes the time of the contingency screening step. I_{fn} is a binary variable which equals to 1 if there is *any* false negative from the prediction, triggering another iteration. It is also convenient to express this time difference in terms of false positives and false negatives:

$$\begin{aligned}\Delta T =& [\alpha_1(N_{fp} - N_{fn})] \\ &+ I_{fn}[\alpha_2(N_p + N_{fn}) + \beta_2 + T_s]\end{aligned} \tag{11}$$

In practice, however, using (11) as the training loss function introduces numerical challenges due to the lack of differentiability. Specifically, the numbers of positives, false positives, and false negatives are counted integers. Moreover, I_{fn} is a binary indicator. To address the numerical issue, we introduce the following relaxations. First,

$$N_p \approx \sum_{i \in C_p} P_i \tag{12}$$

$$N_{fp} \approx \sum_{j \in C_{fp}} P_j \tag{13}$$

$$N_{fn} \approx \sum_{k \in C_{fn}} P_k \tag{14}$$

where C_p, C_{fp} and C_{fn} are the sets of indices for positives, false positives, and false negatives of critical contingencies. In other words, we approximate counting binary outputs by summing probabilities. Next, to approximate the binary indicator function I_{fn}, let y_{gt} be the binary vector indicating the ground truth critical contingencies, and y the binary vector of predictor outputs. The binary indicator of existence of any false negative can then be written as

$$I_{fn} = \max(V_{fn}) \tag{15}$$

where

$$V_{fn} = y_{gt} \cdot (1 - y) \tag{16}$$

To approximate I_{fn} with a differentiable function, first, we let y' be the vector of *probabilities* from the predictor output as opposed to the binary vector y, and let

$$P_{fn} = y_{gt} \cdot (1 - y') \tag{17}$$

Next, we approximate $\max$ by a log-sum-exp function, resulting in the approximate run-time loss function L_T:

$$\begin{aligned} \Delta T \approx L_T = [\alpha_1 (\sum_{j \in C_{fp}} P_j - \sum_{k \in C_{fn}} P_k)] \\ + (1/\gamma) log(\sum_i e^{\gamma P_{fn,i}}) [\alpha_2 (\sum_{i \in C_p} P_i + \sum_{k \in C_{fn}} P_k) + \beta_2 + T_s] \end{aligned} \tag{18}$$

Notably, (18) is differentiable, allowing numerically efficient training of the predictors.

D. Multi-objective Loss Function

Finally, we employ a multi-objective approach by combining both the run-time-based loss function L_T and BCE, essentially using BCE as a strong regularization term. As such, we aim to a) promote high prediction accuracy via BCE and b) seek an effective balance between false positives and false negatives via L_T. To begin with, a straightforward combination using a weighting factor $\lambda \in [0, 1]$ is as follows:

$$L_{total} = \lambda L_T + (1 - \lambda) L_{BCE} \tag{19}$$

However, we recognize the following challenge: as training progresses, the scales of L_T and L_{BCE} may evolve differently so that there is no single λ that can balance the two well at all times. We thus employ the following normalization approach to address this issue. Specifically, a moving average is used to compute and update the average values of the two loss functions over iterations. These averages serve as normalization factors. Using these averages brings both loss objectives to approximately 1, which is on the same scale.

$$\bar{L}_T^{(k+1)} = \mu \cdot \bar{L}_T^{(k)} + (1 - \mu) \cdot L_T^{(k+1)} \tag{20}$$

$$\bar{L}_{BCE}^{(k+1)} = \mu \cdot \bar{L}_{BCE}^{(k)} + (1 - \mu) \cdot L_{BCE}^{(k+1)} \tag{21}$$

In the above, k represents the iteration index, or more specifically, the epoch index in the training process. $\bar{L}$ denotes the average values of the losses. μ is the momentum for incorporating new inputs into the average value. By dividing L by their respective average values, we have the following:

$$\tilde{L}_T = \frac{L_T}{\bar{L}_T} \tag{22}$$

$$\tilde{L}_{BCE} = \frac{L_{BCE}}{\bar{L}_{BCE}} \tag{23}$$

The final objective function is thus:

$$L_{total} = \lambda \tilde{L}_T + (1 - \lambda) \tilde{L}_{BCE} \tag{24}$$

By varying λ, effective weighting of the two objectives can then be found.

IV. Performance Evaluation

We evaluate the performance of the proposed method based on the IEEE 118-bus test system [14]. The system comprises 99 load buses and 54 generators that supply the demand. A total of 186 branches are considered for the N-1 contingencies. Four different solution processes of SCOPF are evaluated for comparison: 1) full optimization of SCOPF, 2) iterative algorithm, 3) vanilla predictor-assisted algorithm with BCE loss, and 4) time-aware predicted-assisted algorithm (Eq(24)).

A. Data generation and predictor architecture

To generate training samples for the predictor, 100,000 load profiles across all buses are created. Uniform distributions are employed: Each demand varies independently between 50% below and 100% above its nominal value. Fig. 2 illustrates the demand samples generated in all the scenarios for both predictor training and testing.

Each load profile leads to a different solution for SCOPF. The iterative algorithm as in Fig. 1-a is used to compute these solutions. The dataset generation is performed based on two AMD EPYC 7643 CPUs with 96 cores, of which 85 cores are used with parallel processing. Among the 100,000 generated scenarios, 63,520 of them resulted in feasible solutions, while the remaining ones are in a sense too extreme and not even feasible. Among these feasible scenarios, 43 lines among all the 186 lines (i.e., 23% of them) appear at least once as critical contingencies.

To understand how often these 43 lines appear as critical contingencies, in Fig. 3, we plot the cumulative percentage of the lines vs. rate of being critical. Specifically, a point on this curve shows what percentage of all these 43 lines are critical for no more than a particular rate. For example, among these

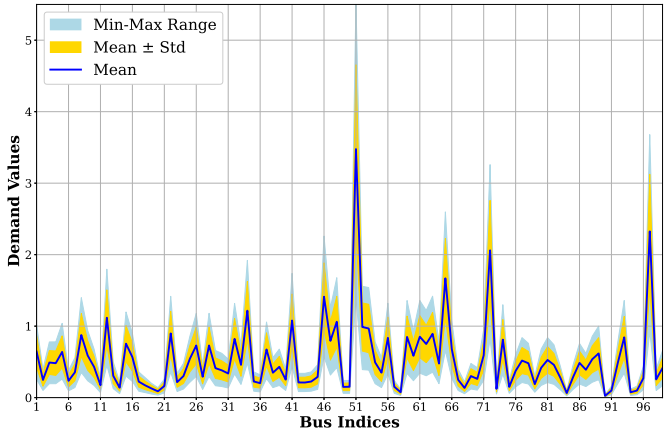

Fig. 2. Load profile scenarios across all the load buses.

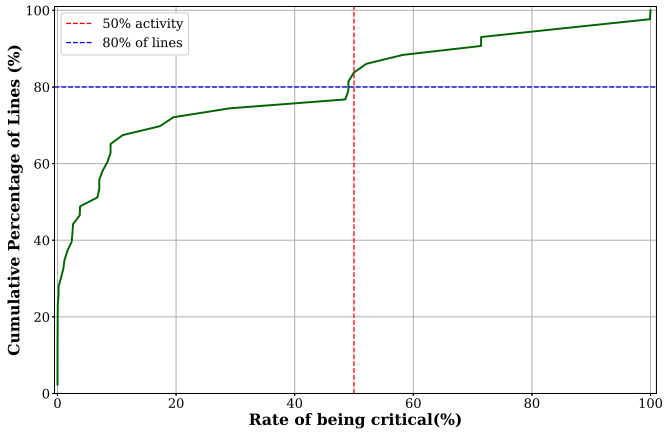

Fig. 3. Cumulative percentage of the lines vs. rate of being critical in all feasible scenarios.

43 lines, about 80% of them are critical for only no more than 50% of the feasible scenarios. Importantly, we observe that critical contingencies are relatively sparse occurrences, offering great potentials for predictor-assisted reduction of problem size and hence solution time.

The predictor used for predicting critical contingencies is a fully-connected neural network with one hidden layer. The input layer transforms input to 512 neurons, the hidden layer transforms 512 to 256 neurons, and the output layer transforms 256 to output. Two dropout layers with 30% dropout rates are implemented. There are 43 binary outputs: Each output takes the value 1 if the corresponding line is predicted to be a critical contingency, and 0 otherwise. 80% of the dataset is used for training and 20% for testing. The Adam optimizer is adopted with a learning rate of 0.001. 70 epochs were used to train the model with a batch size of 256.

B. Time function estimation

To estimate the parameters (α and β) of the solution run-time model, the iterative algorithm is used to evaluate the optimization times for 500 scenarios: the exact times consumed by the screening and optimization steps are measured. The averages over these values are employed for estimating how much time each step of the solution process generally takes. The linear model for the second iteration time is distinguished from the first iteration and is estimated separately. We note that, in the vast majority of the cases, the algorithm ends after no more than two iterations; only 2.6% of scenarios require a third iteration. The estimated time model parameters are shown in Table I.

TABLE I
TIME MODEL PARAMETERS

Parameter	α_1	α_2	β_2	T_s
Estimate	0.008582	0.01106587	0.110013	0.219247

C. Tuning λ for the multi-objective loss

Using BCE as the loss function for training the predictor results in the highest predictor accuracy. In this approach, the predictor treats false positives and false negatives on an equal footing as both decrease accuracy. In practice (cf. Fig. 1-b,) however, a false negative tends to have a much more significant negative impact than a false positive on optimization time due to the necessity of a second iteration. On the other hand, a false positive adds unnecessary constraints to the problem, and hence increases optimization time. As our ultimate objective is to minimize overall optimization time, a carefully balanced trade-off between false negatives and false positives is needed. While our model of solution run-time captures the time-based objective, it is however only approximate. As such, in the multi-objective loss (24), it is not advisable to choose either $\lambda = 1$ (i.e., fully relying on the approximate run-time objective) or $\lambda = 0$ (i.e., fully relying on accuracy as the objective). As such, λ is selected between 0 and 1. In particular, $\lambda = 0.25$ is selected based on validation, and the corresponding progression of training and testing losses are illustrated in Fig. 4. We observe that the training and testing losses decreases over epochs without overfitting, demonstrating effective learning.

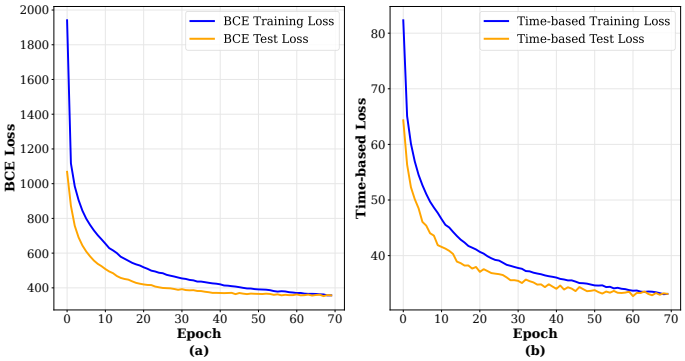

Fig. 4. The loss vs epoch curves: a) BCE loss function, b) Time-based loss function.

D. Accuracy comparison

To illustrate the impact of combining BCE and time-based loss functions, the confusion matrices with $\lambda = 0$ (i.e.,

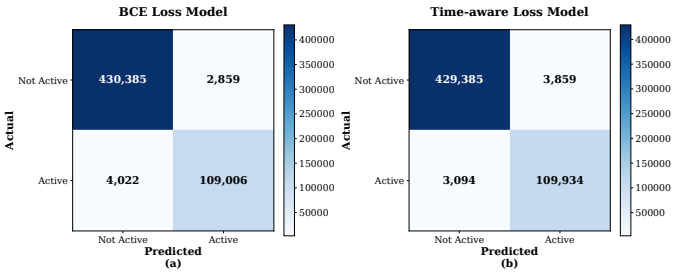

Fig. 5. The confusion matrices: a) BCE loss, b) Multi-objective with time-aware loss incorporated.

BCE loss only) and $\lambda = 0.25$ (i.e., multi-objective loss) are presented in Fig. 5:

With the time-aware loss function incorporated, the number of false negatives decreased, whereas the number of false positives increased, demonstrating the heavier impact of false negatives on optimization run-time that led to this trade-off. Interestingly, the *total* number of false positives and false negatives with the time-aware loss incorporated is 6,953, only slightly higher than that with the BCE loss only (6,881). This implies very similar prediction accuracies: 98.76% with the BCE loss vs. 98.75% with the multi-objective loss. The significant re-balancing of false negatives and false positives thus demonstrates the effectiveness of incorporating time-aware loss which strictly improves the predictor outcomes *with almost no sacrifice of accuracy*.

E. Testing run-time comparison

Finally, 1,000 randomly selected scenarios are tested for run-time evaluation of various solution algorithms. The average run-times per scenario are summarized in Table II.

TABLE II
AVERAGE RUN-TIME OF DIFFERENT METHODS

Method	Full opt.	Iterative algorithm	Predictor -assisted algorithm with BCE	Time-aware predictor -assisted algorithm	Knowing ground-truth critical contingencies
Run-time (s)	1.790 (base)	0.927 (48.21%↓)	0.391 (78.15%↓)	0.356 (80.11%↓)	0.355 (80.17%↓)

We observe that, compared with the full optimization of SCOPF, the average run-time is almost halved by the iterative algorithm. With predictor-assisted algorithm with BCE, the run-time is further reduced by 58%. Moreover, incorporating time-aware loss yet again reduced the run-time by an additional 9%, indicating the importance of re-balancing false negatives and false positives for run-time improvement. Finally, we evaluate the average run-time with *perfect* prediction, i.e., knowing the ground-truth critical contingencies. We observe that the time-aware predictor-assisted algorithm achieves a run-time that is in fact very close to this performance bound.

V. CONCLUSION

We studied accelerating the solution process of preventive SCOPF problems. Predictor-assisted iterative algorithms are developed so that predictions of critical contingencies are incorporated to reduce both the size of SCOPF and the number of iterations of the solution process. Recognizing the markedly different time impact of false negatives and false positives of critical contingency predictions, a novel time-aware loss function is designed and calibrated for training critical contingency predictors that directly minimizes overall run-time. A multi-objective loss function that incorporates both the time-aware loss and the binary cross entropy loss is designed and tuned so that efficient learning is achieved. We evaluated the developed method comprehensively based on the IEEE 118-bus system. Compared with the full SCOPF optimization, we observed an over 80% run-time reduction using the developed time-aware predictor-assisted iterative algorithm. The employment of the time-aware loss itself offers a 9% run-time reduction compared with the standard BCE loss, and achieves an average run-time very close to having perfect prediction.

REFERENCES

[1] Advanced Research Projects Agency-Energy, "Grid optimization competition challenge 1: Problem formulation," 2020, challenge concluded February 18, 2020. [Online]. Available: https://gocompetition.energy.gov/challenges/challenge-1

[2] A. Gholami, K. Sun, S. Zhang, and X. A. Sun, "An admm-based distributed optimization method for solving security-constrained alternating current optimal power flow," *Operations Research*, vol. 71, no. 6, pp. 2045–2060, 2023.

[3] M. Vistnes, V. V. Vadlamudi, and O. Gjerde, "A fast and scalable iterative solution of a socio-economic security-constrained optimal power flow with two-stage post-contingency control," *IET Generation, Transmission & Distribution*, vol. 19, no. 1, p. e70055, 2025.

[4] M. I. Alizadeh and F. Capitanescu, "A tractable linearization-based approximated solution methodology to stochastic multi-period ac security-constrained optimal power flow," *IEEE Transactions on Power Systems*, vol. 38, no. 6, pp. 5896–5908, 2022.

[5] D. T. Phan and X. A. Sun, "Minimal impact corrective actions in security-constrained optimal power flow via sparsity regularization," *IEEE Trans. on Power Systems*, vol. 30, no. 4, pp. 1947–1956, 2014.

[6] H. Sharadga, J. Mohammadi, C. Crozier, and K. Baker, "Optimizing multi-time step security-constrained optimal power flow for large power grids," in *2024 IEEE Texas Power and Energy Conference (TPEC)*. IEEE, 2024, pp. 1–6.

[7] H. Khazaei and Y. Zhao, "Physics-aware fast learning and inference for predicting active set of DC-OPF," in *2022 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, 2022, pp. 1–5.

[8] A. Velloso and P. Van Hentenryck, "Combining deep learning and optimization for preventive security-constrained dc optimal power flow," *IEEE Trans. on Power Systems*, vol. 36, no. 4, pp. 3618–3628, 2021.

[9] S. Park and P. Van Hentenryck, "Self-supervised learning for large-scale preventive security constrained dc optimal power flow," *IEEE Transactions on Power Systems*, 2024.

[10] Z. Yan and Y. Xu, "A hybrid data-driven method for fast solution of security-constrained optimal power flow," *IEEE Transactions on Power Systems*, vol. 37, no. 6, pp. 4365–4374, 2022.

[11] L. Zeng, M. Sun, X. Wan, Z. Zhang, R. Deng, and Y. Xu, "Physics-constrained vulnerability assessment of deep reinforcement learning-based SCOPF," *IEEE Transactions on Power Systems*, vol. 38, no. 3, pp. 2690–2704, 2022.

[12] B. N. Giraud, A. Rajaei, and J. L. Cremer, "Constraint-driven deep learning for n-k security constrained optimal power flow," *Electric Power Systems Research*, vol. 235, p. 110692, 2024.

[13] S. Liu, Y. Guo, W. Tang, H. Sun, W. Huang, and J. Hou, "Varying condition SCOPF based on deep learning and knowledge graph," *IEEE Transactions on Power Systems*, vol. 38, no. 4, pp. 3189–3200, 2022.

[14] MATPOWER Development Team, "IEEE 118-bus test case," https://github.com/MATPOWER/matpower/blob/master/data/case118.m, 2024, mATPOWER software package.