

# Smart-PGSim: Using Neural Network to Accelerate AC-OPF Power Grid Simulation

Wenqian Dong  
UC Merced  
California, USA  
wdong5@ucmerced.edu

Zhen Xie  
UC Merced  
California, USA  
zxie10@ucmerced.edu

Gokcen Kestor  
Pacific Northwest National Laboratory  
Washington, USA  
gokcen.kestor@pnnl.gov

Dong Li  
UC Merced  
California, USA  
dli35@ucmerced.edu

**Abstract**—In this work we address the problem of accelerating complex power-grid simulation through machine learning (ML). Specifically, we develop a framework, Smart-PGSim, which generates multitask-learning (MTL) neural network (NN) models to predict the initial values of variables critical to the problem convergence. MTL models allow information sharing when predicting multiple dependent variables while including customized layers to predict individual variables. We show that, to achieve the required accuracy, it is paramount to embed domain-specific constraints derived from the specific power-grid components in the MTL model. Smart-PGSim then employs the predicted initial values as a high-quality initial condition for the power-grid numerical solver (warm start), resulting in both higher performance compared to state-of-the-art solutions while maintaining the required accuracy. Smart-PGSim brings  $2.60\times$  speedup on average (up to  $3.28\times$ ) computed over 10,000 problems, without losing solution optimality.

## I. INTRODUCTION

Artificial Intelligence (AI) and Machine Learning (ML) techniques are revolutionizing the way researchers approach scientific and engineering problems. By employing reverse-engineering and automatic learning methodologies it is often possible to solve complex, unstructured problems with a fraction of the computing power and execution time required by traditional direct and first-principle methods. ML provides researchers with a powerful tool to learn the structure of physical phenomenon directly from Nature, rather than having to explain the causal relationships through direct application of physics law. Many research and engineering fields, from image recognition to autonomous driving, from health to natural language processing (NLP), have experienced a tremendous boost in performance and efficiency over the last few years. Many problems that seemed impossible to be solved, can now be tackled thanks to the use of ML methodologies.

The use of ML methodologies in scientific and engineering applications has been, somehow, limited. By using Neural Network (NN) as a tool to learn and model complicated (non-)linear relationships between input and output data sets, scientists have shown preliminary success in some HPC problems (e.g., detecting neutrinos [1], climate simulations [2], and fluid dynamic simulation [3]). With NN, scientists are able to augment existing scientific simulations by improving simulation accuracy and significantly reducing latency [4]–[10]. However, although there have been successful studies of applied ML to scientific applications, these fields have not

experienced the double- or triple-digit improvements seen in other domains. The reason for such discrepancy is the fundamentally different characteristic of scientific and engineering applications compared to domains such as image recognition and NLP: scientific applications require a level of precision and robustness that may not be provided by most of the current ML methods employed in other domains.

In this work we study the implication of using ML techniques to accelerate the power-grid simulations, the structure of the ML model to be used, the relative importance of the features selected, and, most importantly, the impact of incorporating physics constraints on the performance of the application. The power-grid simulation [11] is a complex nonlinear optimization problem for the management of power flow and is critical to the power industry in electricity dispatch scheduling, reliability analysis, and maintenance planning for power and generators [12], [13]. The alternating current optimal power flow (AC-OPF) simulation is the most fundamental and time-consuming part of the power grid simulation. The problem size of AC-OPF is generally large, in which the scale of the generator node can vary from  $10^3$  to  $10^6$  [14]–[16]. Despite the large problem size, the AC-OPF simulation requires near real-time updates during power scheduling. In a typical scenario, power grid operators repeatedly solve the optimal power flow problem multiple times within every minute throughout a day, every day of the year [17]–[19], for decades, to ensure that the power grid system is operating reliably and safely. The high requirement on the simulation latency and frequent usage of the simulation make the power grid simulation a mission-critical application under active development in the HPC community and within the U.S. Department of Energy (DOE) and the DOE Exascale Computing Project (ECP).

NN has been applied to solve the optimal power flow problem in the past [20]–[24]. However, existing efforts have focused on improving performance by entirely replacing the simulation solver with an approximated NN model or facilitating existing solvers by identifying active constraints. While these approaches provide considerable speedups, NN provides only an *approximation* of the optimal solution or approximates computation in the simulation. As a result, these approaches may not provide the desired precision for the solution or may provide a non-optimal solution. In the context of power-grid simulation, the first case results in an infeasible solution (e.g.,

arXiv:2008.11827v1 [eess.SP] 26 Aug 2020

not being able to provide the required power to satisfy the user demand) while the second case may result in a large economic loss (i.e., solving the problem at a much higher cost.)

In this paper, we introduce a new method to apply NN to the AC-OPF simulation. Unlike the existing studies, we employ NN to generate an initial solution and then inject it to the AC-OPF solver. Because of the high quality of the initial solution and guidance of other outputs generated by the proposed NN, the simulation can run faster (or converge faster) without losing the solution optimality.

There are several challenges to apply our method to the AC-OPF simulation. First, deciding which variables in the AC-OPF simulation should be used as NN output and quantifying the sensitivity of simulation execution time and convergence to those variables is a challenge. The AC-OPF simulation involves a set of variables, including power grid information and multiple variables critical for the computation convergence. We cannot use all of them as NN output because that largely increases network complexity and puts high requirements on training efficiency and sufficiency of training samples. On the other hand, using only the solution of the AC-OPF as NN output, we often lose simulation robustness because of the limited guidance for the simulation from the initial solution. Furthermore, understanding the sensitivity of simulation time and convergence to those variables is useful for deciding NN topology and generating high-quality initial solutions.

Second, how to apply NN to the AC-OPF simulation without disturbing the simulation robustness is a challenge. Due to the non-convex and nonlinear nature of the AC-OPF problem, the simulation process itself is at the risk of a failed convergence with the use of iterative numerical methods. The simulation must be robust enough to handle various power flow cases with computation convergence. Using NN to generate an initial solution, we must make sure that the initial solution makes sense and does not impact the computation convergence in the original simulation.

Third, how to impose physical constraints on NN to ensure the validness of NN prediction. Traditionally, the NN model is manually constructed by computer scientists as a black box with limited or no domain knowledge and without considering domain requirements. Although NN models can be adjusted as a nonlinear tool box to accommodate a change of inputs and generate some approximation, the understanding of the model is lost. Instead of blindly trusting that the data mining algorithm will produce a correct model, we seek for what variables physically mean and which physical laws are driving the interpretable evolution of the analysis paradigm.

To address the above challenges, we introduce, Smart-PGSim, a framework that facilitates the construction of a NN model to accelerate the AC-OPF simulation. Smart-PGSim is based on the following design principles. First, it generates an NN model that uses power grid components as inputs and variables critical for the simulation convergence as the model output. By using Smart-PGSim, we perform a sensitivity study to understand the impact of the output accuracy on execution time and convergence, by using precise or imprecise data for

some variables. This sensitivity study provides guidance on choosing a correct and efficient NN topology.

Second, Smart-PGSim uses a novel multitask-learning NN model to accelerate the AC-OPF simulation. The model topology allows information sharing when predicting multiple dependent variables while including customized layers for each variable. This multi-task model improves the model accuracy, compared with the traditional single-task model, while simplifying the training process.

Third, Smart-PGSim allows embedding physical constraints from the original formulation of the AC-OPF problem into the NN model and imposes those constraints into the training objective function or the last layer based on transformation on equality and inequality in the constraints.

We summarize our major constitutions as follows.

- A systematic approach and a framework (Smart-PGSim) to accelerate optimization problems in general and the AC-OPF power grid simulation in particular;
- A set of techniques to construct NN models for robust, accurate, and high-performance numerical solvers;
- We show that Smart-PGSim achieves  $2.60\times$  speedup on average (with the consideration of NN cost) and up to  $3.28\times$  over the original AC-OPF simulation method (computed over 10,000 problems as the simulation input), without losing the optimality of the final solution.

## II. BACKGROUND

In this section, we review the problem formulation and the primal-dual interior-point method in the AC-OPF problem.

### A. Problem Formulation for AC-Optimal Power Flow

The AC-OPF problem aims at minimizing an objective function by optimizing the power dispatch and transmission decisions. The objective function calculates the cost of power generation, subjecting to physical, operational, and technical constraints including Kirchhoffs laws, operating limits of generators, voltage levels, and loading limits of transmission lines [25]. The standard AC-OPF problem is formulated as:

$$\min_X f(X) \quad (1a)$$

$$s.t. G(X) = 0 \quad (1b)$$

$$H(X) > 0 \quad (1c)$$

$$X_{min} \leq X \leq X_{max}. \quad (1d)$$

where  $f(X)$  is the cost function to be minimized, and  $X$  is an optimization vector as the simulation solution. Eqn. 1b builds an equality constraint, which sets up power balance incorporating variable bounds. The formulation 1c is an inequality constraint that sets up branch flow limits. The optimization vector  $X$  is bounded by  $X_{min}$  and  $X_{max}$  which introduces the constraints on reference bus angles, voltage magnitudes, and generator injections. The optimization vector  $X$  consists of four variables,  $X = \{V_a; V_m; P_g; Q_g\}$ , i.e., voltage angles  $V_a$ , voltage magnitudes  $V_m$ , generator real power injections  $P_g$  and reactive power injections  $Q_g$ .

In power grid simulation,  $G(X) = 0$  is an AC nodal power balance equation and enables the AC-steady conditions of the power system, which can be split into real and reactive parts:

$$P_i(C_g, P_g) = P_d + P_{bus}(Y_{bus}, V_a, V_m) \quad (2a)$$

$$Q_i(C_g, Q_g) = Q_d + Q_{bus}(Y_{bus}, V_a, V_m). \quad (2b)$$

In Eqn. 2,  $C_g$  is the generator connection matrix reflecting generator locations in a power grid network.  $Y_{bus}$  is the bus admittance matrix including all constant impedance elements.  $P_i$  and  $Q_i$  refer to power real and reactive injection for the power system.  $P_d$  and  $Q_d$  are power loads.  $P_{bus}$  and  $Q_{bus}$  are power consumption of transmission lines.

### B. Primal-dual Interior Point Solver

The primal-dual interior point method [26], [27] is an efficient algorithm to solve the non-convex optimization problem for AC-OPF. Matpower [28] is a widely used framework for solving power flow and optimal power flow problems. Matpower uses a solver, called MIPS, to solve those problems.

To solve the AC-OPF problem, MIPS first converts the inequality constraint in Eqn. 1c into an equality constraint with a vector  $Z$ ,  $H(X) + Z = 0$  where  $Z$  is a vector of positive slack variables. MIPS further uses a barrier function  $\ln(Z)$  to bound  $Z$ . Based on that, MIPS uses a Lagrangian formulation to formulate the AC-OPF problem as follows.

$$L^\gamma(X, Z, \lambda, \mu) = f(X) + \lambda^T G(X) + \mu^T (H(X) + Z) - \gamma \sum_{m=1}^{n_i} \ln(Z_m) \quad (3)$$

where  $\lambda$  is called the equality Lagrangian multiplier,  $\mu$  is called the inequality Lagrangian multiplier, and  $\gamma$  is called the perturbation parameter. During the solving process,  $\gamma$  is approaching zero. If  $\gamma = 0$ , the solution to this Lagrangian formulation equals to that of the original form (Eqn. 1).

Matpower uses Newton method to solve Eqn. 3, which iteratively converges to a set of convergence criteria (particularly four terminate conditions) [28]. The Newton Method is computationally intensive and requires constant updates of input and output variables: the method firstly updates  $X$  and  $\lambda$ , then  $Z$  based on  $X$ , and, finally,  $\mu$  based on  $X$  and  $Z$ . As we will see in the next Sections, this structure introduces internal dependencies on the variables that our model exploits for better performance and accuracy.

### III. RELATED WORK

OPF problems can be categorized into three forms: economic dispatch (ED) [29], Direct Current (DC-OPF) [30], and Alternating Current (AC-OPF) problems [31]. The AC-OPF problem is the original OPF problem, which is non-convex and the most challenging one among the three. ED and DC-OPF problems are the relaxed version of the AC-OPF problem, which is obtained by removing or linearizing some constraints in the AC-OPF problem, respectively. Traditionally, numerical iteration algorithms are used to solve the OPF problem [32]–[36]. However, the time complexity of these algorithms might be significant, especially when the scale of the transmission power system becomes large. To deal

with this limitation, researchers have explored learning-based approaches to accelerate solving OPF problems.

Vaccaro et al. [37] use the principal component analysis (PCA) to identify unknown relationships among OPF variables, which reduces the number of variables to be solved for a solution. Ng et al. [24] use a statistical learning-based approach to set up a mapping between input power requirement and output dispatch scheme. However, the approaches mentioned above consider only the prediction accuracy without taking into account the correlation among OPF problem variables, which leads to a solution that can not satisfy all of the problem constraints. Pan et al. [38] use the multilayer perceptron (MLP) to learn the mapping between input and decisions for DC-OPF and apply it to obtain optimized operating decisions upon arbitrary inputs. While this approach is effective for DC-OPF, it has low generalization capacity and cannot be applied to a non-convex problem such as AC-OPF. Previous works [20]–[22] have leveraged machine learning to accelerate the AC problem. Zamzam et al. [22] develop an online method based on machine learning to obtain feasible solutions to the AC problem by loading the optimal generator set-points and enforcing generation limits. However, the AC grid contains more voltage phase angles beyond magnitudes and reactive parts of power generation. Unlike these methods, the proposed approach includes all the inputs of the AC problem and guarantees that the predicted solution is optimal while providing significant performance improvement.

In this work, we use NN models to solve the AC-OPF problem. We follow a radically different approach compared to previous work in that we employ ML to estimate a high-quality initial solution for the solver, greatly speeding up the entire computation, and then leverage traditional AC-OPF solver to guarantee precision and robustness of the solution. We show in the next Sections that our approach can simultaneously provide large performance improvement *and* high-precision solutions.

### IV. OVERVIEW

This section overviews our proposed framework “Smart-PGsim”. The Smart-PGsim framework includes two phases: offline and online phases. Figure 1 shows the workflow of Smart-PGSim.

The offline phase investigates the power grid simulation to find the most crucial features to construct an efficient NN model for online acceleration. In particular, our sensitivity study (Section V) firstly identifies the most important features (in other words, determining variables in MIPS as the output of the prediction model) and quantifies the impact of the imprecise variables (i.e., variables with some accuracy loss) on the success rate of simulation and performance in terms of execution time. The results in sensitivity study are used to guide the NN topology design.

Then, Smart-PGSim constructs a multi-task learning (MTL) model (Section VI) guided by the sensitivity study. The model shares domain information between prediction tasks, while uses a customized topology design for each task. Smart-PGSim prioritizes features to distinguish main tasks and auxiliary

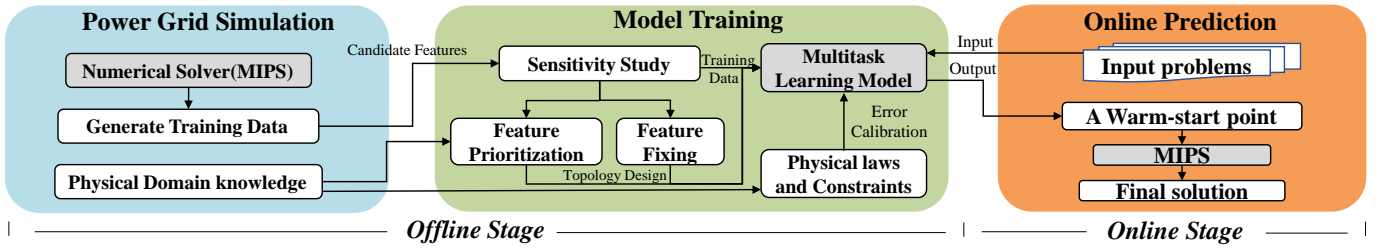


Fig. 1. Workflow of the proposed Smart-PGsim

tasks and applies a physics-dependent hierarchy for those features have domain specific dependency.

Next, Smart-PGSim incorporates physical domain knowledge during model training to improve prediction quality (Section VII). The domain knowledge presents physical constraints providing explicit and implicit error bounds. Using the domain knowledge improves prediction accuracy, interpretability, and defensibility of the MTL model, while simultaneously augmenting physical data as complementary.

After the above offline phase, the well-trained MTL model can be used to generate a warm-start point for MIPS as online prediction. The MIPS (or other numerical solvers) can use these high-quality start points for quick convergence.

## V. SENSITIVITY STUDY

The ability of NN to produce high-quality results is the key to improve simulation performance (making the simulation quickly converged). In this section, we discuss the opportunity available in NN with the assist of a sensitivity study tool that detects and analyzes those variables critical to simulation convergence and execution time.

We introduce two data types, i.e., *imprecise default data* and *precise simulation data*, to study the impact of noisy feature to simulation quality and execution time. By doing so, we can check the (lowest) highest performance brought by these (im-)precise data, which demystifies the contribution of each feature to success rate and speedup.

- 1) Imprecise default data: The default value at the initial point in MIPS.
- 2) Precise simulation data: The exact solution collected in the numerical solver, i.e., MIPS. We take the ground-truth value as the precise data.

Our sensitivity study first checks the convergence criteria in the MIPS code and collects those variables critical to the simulation converge, namely  $X$ ,  $\lambda$ ,  $\mu$ , and  $Z$ . We use these (im-)precise data as initial points to test the importance of each variable in two aspects, i.e., the impact on success rate and speedup. Here, *success rate* refers to the ratio of those initial solution can reach the convergence criteria to the total number of input problems. *Speedup* is time acceleration, namely the rate of actual solving time to the exact solving time in MIPS.

Then, we include eight test systems<sup>1</sup> and generate 10,000 samples for each system by varying input loads to analysis

the impact of using different initial points. Initializing the four variables with precise and imprecise types, we have  $2^4$  combinations to analyze the contribution of each variable. Table I shows the results of 16 combinations. For each combination, we use “0” and “1” to indicate the imprecise data and precise data respectively. To calculate the success rate and speedup, we take a baseline, the combination where all variables using imprecise default data.

Table I reveals that precision improvement on these initial variables does not always bring benefits to simulation performance in terms of success rate and speedup. A high improvement on precision might even decrease the success rate of the simulation. For example, the baseline case I (using all default parameters) has a success rate of 100%, while improving the precision on the feature  $Z$  (e.g., the case II) alone leads to failed convergence at most input problems. Hence, blindly building a NN model to numerically approach those precise values may reduce success rate and lose simulation performance.

We further analyze the performance results in Table I and summarize some interesting observations.

- **Observation 1:** Using precise  $X$  leads to a 100% success rate (see case IX), while the features  $X$ ,  $\lambda$ ,  $\mu$  and  $Z$  jointly contribute to high speedup see case XVI).
- **Observation 2:** The contribution of  $Z$  to the success rate and speedup strongly depends on whether  $\mu$  use precise data. For example, the success rate is dropped down when involve a precise  $Z$  without a precise  $\mu$  (see case XII with respect to X).
- **Observation 3:** The contribution of  $\lambda$  to the success rate and speedup is independent of whether the other features use precise data or not. For example, the success rate of initialing with a precise  $\lambda$  does not change with/without a precise  $X$  or  $\mu$  and  $Z$  (see case V, VIII, XIII and XVI).
- **Observation 4:** Features  $X$ ,  $\lambda$ ,  $\mu$  and  $Z$  have implicit dependency. Improving accuracy of one feature cannot guarantee the overall performance improvement to the success rate and speedup. For example, improving the accuracy of  $\lambda$ ,  $\mu$ , or  $Z$  on a precise  $X$  can not guarantee the improvement of success rate and speedup (comparing case IX with cases X, XI and XIII.)

Observations 1 and 4 indicates that the analyzed features are highly inter-dependent, which can be targeted on multi-task prediction. We build a MTL model to enable the information

<sup>1</sup> Refer to Table II for a more detailed illustration of test systems.

TABLE I  
 ABLATION STUDY ON THE INPUT SIGNALS  
 THE SR AND SU REPRESENT SUCCESS RATE(%) AND SPEEDUP( $\times$ ) RESPECTIVELY.

	$X$	$\lambda$	$\mu$	$Z$	bus 5		bus 9		bus 14		bus 30		bus 39		bus 57		bus 118		bus 300		Observation		
					SR	SU	SR	SU	SR	SU	SR	SU	SR	SU	SR	SU	SR	SU	SR	SU			
I	0	0	0	0	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	baseline
II	0	0	0	1	0	-	10	0.67	0	-	89	0.66	0	-	6	0.99	0	-	0	-	0	-	
III	0	0	1	0	100	1.04	100	0.73	100	0.92	100	0.93	95	0.95	88	0.60	99	1.03	100	1.24	100	1.24	
IV	0	0	1	1	100	1.09	100	0.96	100	0.85	3	0.22	75	1.02	99	0.72	0	-	68	1.24	100	1.24	
V	0	1	0	0	100	0.98	100	0.99	100	1.00	30	1.06	100	1.00	100	0.98	100	0.98	98	0.99	100	0.99	OBS 3
VI	0	1	0	1	0	-	8	0.73	0	-	79	0.70	0	0.61	9	0.90	0	-	0	-	0	-	
VII	0	1	1	0	100	0.99	0	-	80	0.61	98	0.90	100	1.09	100	0.89	100	0.93	100	1.08	100	1.08	
VIII	0	1	1	1	100	1.05	100	1.24	100	1.32	30	0.19	100	1.39	100	1.25	100	1.59	100	1.75	100	1.75	OBS 3
IX	1	0	0	0	100	1.17	100	0.99	100	0.99	100	0.95	100	1.06	100	1.01	100	0.99	100	1.08	100	1.08	OBS 1, 4
X	1	0	0	1	0	-	0	-	0	-	0	-	0	-	11	0.94	0	-	0	-	0	-	OBS 2, 4
XI	1	0	1	0	100	1.28	100	0.85	100	0.88	100	1.33	95	1.48	100	0.67	90	0.84	96	1.24	100	1.24	OBS 4
XII	1	0	1	1	100	1.45	100	1.03	100	0.80	95	1.26	80	1.22	100	0.65	0	-	53	0.87	100	0.87	OBS 2
XIII	1	1	0	0	100	1.18	100	1.00	100	0.98	100	0.94	100	1.06	100	1.00	100	0.99	100	1.07	100	1.07	OBS 3, 4
XIV	1	1	0	1	0	-	0	-	0	-	0	-	0	-	10	0.97	0	-	100	1.18	100	1.18	
XV	1	1	1	0	100	1.28	100	0.79	100	0.90	100	1.09	100	1.22	100	0.93	100	0.94	100	1.32	100	1.32	
XVI	1	1	1	1	100	5.21	100	4.58	100	3.74	100	6.15	100	6.60	100	4.58	100	7.63	100	14.6	100	14.6	OBS 1, 3

sharing for the inter-dependency. We decide feature priority by *dependency between features*, namely, the contribution of a feature to success rate and speedup is changed with a variation of another feature. For example, since  $\lambda$ ,  $\mu$  and  $Z$  have dependency on  $X$ , we should make  $X$  very accurate. We give  $X$  the highest priority.

Observation 2 and 3 reveals features show differences on dependency. Some features (e.g.,  $\lambda$ ) are relatively independent while others (e.g.,  $\mu$  and  $Z$ ) have dependency, which implies customized design for different feature prediction should be considered in modeling. Also, we observe that  $\lambda$  is an equality factor while  $\mu$  and  $Z$  contribute to inequality together in Eqn 3. Such information from physical law validates feature dependency and maybe can be used to deal with the dependency in model training inversely.

Driven by these observations, we introduce an interactive learning model for multi-objective modeling (discussed in Section VI) and impose domain knowledge to strength physical understanding for prediction quality (discussed in Section VII).

## VI. AN INTERACTIVE LEARNING MODEL

In this section, we develop a MTL model to enable multitask prediction. Based on the observation from sensitivity study, we implement domain specific design through prioritizing features and enforcing a physics-dependent hierarchy in the MTL model. After that, we depict the details of MTL parameters.

### A. Multitask Learning

Multitask learning is an inductive transfer learning method [39], [40]. A MTL model is typically composed of shared layers and task-specific layers. Unlike using multiple separate models for each task, the MTL model enables us to share information from common layers while customizing specific layers for corresponding tasks. The training signals of different tasks can be learned as inductive biases to facilitate the learning of all tasks, which achieves a unification of the shared information and the task-specific information.

**Information-sharing in shallow layers.** Observation 1 reveals that there is correlation among these four features and we would like to leverage these relations in our model. To incorporate this correlation, we utilize information sharing in MTL by parameter sharing and loss sharing.

- **Parameter sharing.** The common layers share the same weights and topology between tasks. By sharing parameters, tasks share low-level semantic information to complement domain knowledge with each other. Meanwhile, parameter sharing can alleviate the risks of overfitting due to the noise brought by multiple tasks.
- **Loss sharing.** The tasks to predict  $X, \lambda, \mu, Z$  share a common loss function to update training gradient in the MTL. The minimal loss of different tasks are usually in different positions. By sharing losses, the MTL passes information and avoids being trapped in a local optimal.

**Task-specific learning in deeper layers.** Besides the features  $X, \lambda, \mu, Z$  being correlated, observations 2 and 3 shows that specific design for different feature prediction should be considered. In task-specific layers, we introduce specific model topologies (estimators) for each task based on the task demands. For example, a task requires a positive output. We apply a rectified linear unit (ReLU), a type of activation functions, at the last layer to bound the output always positive.

Figure 2 shows the topology of the proposed MTL. Given a power network topology, we use the power load (including both the active part  $P_d$  and reactive part  $Q_d$ ) as model inputs and estimate seven tasks (four variables in  $X$ ). In the MTL, the shared layers are extracting information from different tasks, while the task-specific layers (estimators) utilize customized designs to generate their own dedicated results.

### B. Domain-Specific Design

Besides using shared layers and task-specific layers, we introduce a domain-specific design into MTL: this design is

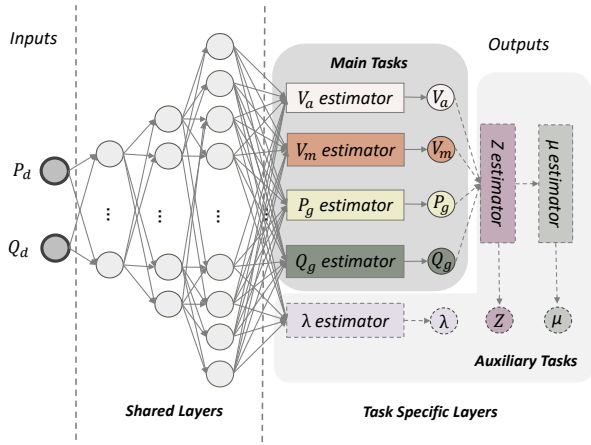


Fig. 2. Topology of the MTL.

driven by our observations on (1) the contribution difference of the four features to success rate and speedup and (2) the dependency between features. The domain-specific design includes two techniques, feature prioritization and a physics-dependent hierarchy, discussed as follows.

**Feature prioritization.** Observation 1 shows that precise  $X$  guarantees the success of simulation convergence, while precise  $\lambda$ ,  $\mu$ , and  $Z$  contribute to simulation acceleration. We prioritize the four features by specifying the prediction of  $X$  as the main task while the prediction of other three features as auxiliary tasks. The auxiliary tasks are used as an augmentation to provide additional information for the main task. Through “eavesdropping” the main tasks, the auxiliary tasks interact with the main task implicitly. More importantly, learning the direct solution  $X$  in the main task gives the user high simulation quality while estimating the Lagrangian factors ( $\lambda$ ,  $\mu$ , and  $Z$ ) in the auxiliary tasks maximize performance speedup. Technically, we apply “detach()” operation [41] for these auxiliary tasks periodically. The detach operation blocks the gradient back-propagation to the shared layers (which are contribute to the main task  $X$ ). In other word, we set a knob of detach operation to alternately train the main task or the entire model. In particular, the MTL focuses on improving main tasks when we activate the detach operation, while facilitates the interaction between main tasks and auxiliary tasks when the detach operation is disabled.

**A physics-dependent hierarchy.** Observations 2 and 3 reveal the dependence between  $Z$  and  $\mu$  and the independence of  $\lambda$ , respectively. We find these observations are consistent with the computation order in the solving process. In particular, the simulation process takes the order of (1) computing  $X$  and  $\lambda$ ; (2) computing  $Z$  based on  $X$ ; and (3) computing  $\mu$  based on  $X$  and  $Z$ . This is consistent with the existing work [42], [43].

To fully exploit the benefit of information sharing, we enforce a physics-dependent hierarchy in MTL. As shown in Figure 2, we first infer the main task  $X$  ( $V_a$ ,  $V_m$ ,  $P_g$ ,  $Q_g$ ) and an independent auxiliary task  $\lambda$ . Then, we predict task  $Z$  based on  $X$ . After that, we estimate  $\mu$  based on the predicted  $Z$ .

### C. Details on Multitask Learning Model

In this section, we present details about the topology parameters, the loss function and the pre-processing method of the MTL model. We use a power grid system of 300 buses as an example, but the MTL modeling method is general for any other power grid systems. Figure 2 generally depicts the model topology.

The shared layers take the power load  $P_d$  and  $Q_d$  at each bus as input, totaling 600 inputs. There are five fully-connected layers as the shared layers. We set the numbers of neurons in the five layers as 600, 720, 840, 960, and 1080 respectively. The five fully-connected layers extract shared features and feed them to seven specific estimators (four in  $X$  and  $\lambda$ ,  $\mu$ ,  $Z$ ), each of which is a fully-connected network customized for a task. We use ReLU as the activation function to increase the model nonlinearity. We use a variant of  $L_1$  loss [44], the Charbonnier loss, as our loss function. This is a supervised loss function calculating the difference between each of the predicted output variables  $v$  and the corresponding ground-truth value  $v_{gt}$  collected in the MIPS solver. Our loss function is defined as follows.

$$L = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} W_v \sqrt{(v - v_{gt})^2 + \epsilon^2} \quad (4)$$

where  $\mathcal{V}$  is a set consisting of  $V_a$ ,  $V_m$ ,  $P_g$ ,  $Q_g$ ,  $Z$ ,  $\lambda$  and  $\mu$ ;  $W_v$  is a weight for a task  $v$  and  $\epsilon$  is a small constant for numerical stability. We set  $\epsilon$  as  $1e - 9$  in our study.

## VII. PHYSICS-INFORMED LEARNING

The solution in power grid simulation must respect several physical constraints, such as power generation, line flow, and bus voltage constraints. Incorporating these constraints into the MTL model not only improves model accuracy but also increases the model interpretability.

In general, the constraints are classified into hard and soft constraints. The *hard constraint* includes some strict bounds on the variable ranges in applications; The *soft constraint* includes domain knowledge to improve model accuracy, such as physical principles, conservation laws, and others gained from theoretical or computational studies. We introduce four objective functions to incorporate domain knowledge and impose those constraints by minimizing the objective functions.

### A. Embedding AC Nodal Power Balance Equations

The power grid simulation includes a power flow equality constraint shown in Eqn. 2 to make the simulation of the power grid system stable and make the simulated solution feasible. We integrate the AC nodal power balance equations (Eqn. 2) into the objective function  $f_{AC}$  to guide model training.

$$f_{AC} = |P_d + P_{bus}(Y_{bus}, V_a, V_m) - P_i(C_g, P_g)| + |Q_d + Q_{bus}(Y_{bus}, V_a, V_m) - Q_i(C_g, Q_g)| \quad (5)$$

The above objective function bridges model inputs ( $P_d$ ,  $Q_d$ ), outputs ( $X$ ,  $\lambda$ ,  $\mu$ ,  $Z$ ) and the physics information ( $C_g$ ,  $Y_{bus}$ ) of power networks to yield quantitatively better physical connection. In particular, the generator connection matrix  $C_g$



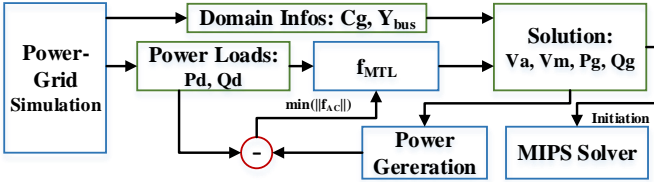


Fig. 3. Embedding AC physical laws in MTL training

and the bus admittance matrix  $Y_{bus}$  are critical information determined by the physical network of power system. Eqn. 2 shows the AC power system keeps stable only if the power generation equals to the power load. In the objective function  $f_{AC}$ , we calculate the differences between power load and generation and minimize the difference approaching to zero.

Figure 3 shows how the objective function  $f_{AC}$  works in MTL training. In the power grid simulation, we utilize domain information  $C_g$  and  $Y_{bus}$ , which provide power-grid bus topology and resistance information respectively. Power loads ( $P_d$  and  $Q_d$ ) are the input fed to the MTL to produce solutions  $V_a, V_m, P_g, Q_g$ . We integrate the AC power balance law (Eqn. 5) to calibrate the training loss in  $f_{MTL}$ . In particular, we calculate the power generation based on the prediction solution  $X$  and domain information  $C_g$  and  $Y_{bus}$ , and subtract power generation from the power loads. We then calculate the difference between the power loads and power generation, and try to minimize the difference within the objective function  $f_{AC}$ . The only block with training parameters is the  $f_{MTL}$  and all blocks are differentiable.

The above training process is driven by the predicted data and facilitates the prediction inversely. Such a data-driven architecture  $f_{AC}$  mitigates the risk of obtaining infeasible solutions, such as those predicted solution misled by the noise of training data and violating the basic AC power balance law. Embedding the objective function  $f_{AC}$  has two significant benefits. First, since the information of model inputs is limited to predict its outputs, we incorporate non-trivial data-augmentation as a complementary to increase prediction accuracy. Second, we can efficiently perform transfer learning with fewer training data even if the typology of power network is modified, e.g., a transmission line in the power-grid bus is suddenly broken. With this, we can improve our MTL prediction and facilitate the solution robustness.

### B. Guarding Inequality Constraints

The AC-OPF formulation includes two inequality constraints: one is explicit, quantitatively bounding  $X$  by  $X_{min} < X < X_{max}$  (Eqn. 1d), and the other is implicit, limiting branch flow by  $H(X) > 0$  (Eqn.1c). For the implicit inequality, we impose physics information  $C_g$  and  $Y_{bus}$  to calculate the branch flow state  $H(X)$  and check if the  $H(X)$  violates the bounds. We utilize exponential functions to punish the overflow error in these inequality constraints and force the prediction to be bounded by the expected, normalized range. Eqn. 6 shows how we use the exponential functions. In the equation, we build an objective function  $f_{ieq}$  to incorporate

the inequality equations as a penalty loss.

$$f_{ieq} = e^{-H(X)} + e^{(X-X)} + e^{(X-X)} \quad (6)$$

Where  $X$  is a predicted feature in MTL. Once the predicted  $X$  violates the inequality constraints, for example,  $H(X_k) < 0$ , the overflow error will be visibly shown up in the objective function  $f_{ieq}$  and calibrated through backpropagation in the training phase. Hence, the main task  $X$  is restricted in a quantitative way to improve simulation quality.

Guarding inequality constraints in our model mitigates the overflow error in inequality constraints while facilitates the feasibility of model prediction.

### C. Optimization of Cost Function

The ultimate goal of the AC-OPF is to minimize the cost function  $f(X)$  (Eqn. 1a). We explore the physics information in  $f(X)$  to construct an objective function  $f_{f(X)}$  and minimize the loss between the predicted cost and ground truth cost.

$$f_{f(X)} = |f(X) - f_0| \quad (7)$$

where  $f_0$  is the ground-truth value of the cost collected by the numerical solver, MIPS. Feature  $X$  is our model prediction. In  $f(X)$ , we utilize the characteristics of energy consumption on generators to calculate the predicted cost  $f(X)$ . Then, the objective function  $f_{f(X)}$  calibrates the predicted cost  $f(X)$  with the ground-truth cost  $f_0$  to reach the optimal solution.

### D. Implying Lagrangian Conservation

In Eqn. 3, the AC-OPF problem can be solved as the equality constraints  $G(x) = 0$  and slacked inequality constraints  $H(x) + Z = 0$  approach zero. Here, we apply two ways to imply the Lagrangian formulation into MTL training. First, we reconstruct the inequality and equality constraints as soft constraints, which is imposed in the loss function to guide the training. Then, we refer the variable bounds  $Z > 0$  and  $\mu > 0$  as hard constraints and apply an activation function to strictly bound model prediction. We construct an objective function  $f_{Lag}$  to guide the training subject to the soft constraints.

$$f_{Lag} = |\lambda^T G(X)| + |\mu^T (H(X) + Z)| \quad (8)$$

We incorporate the hard constraints during the training phrase by projecting predictions onto a region induced by the constraints. In particular, we first pre-process the raw data of ground truth into the normalized range  $[0, 1]$ . Then, we apply a ‘‘sigmoid’’ activation function at the last layer to bound the output range of  $Z$  and  $\mu$  to be positive and into the same range  $[0, 1]$ . The above techniques provide hard upper and lower bounds on prediction and guarantee its feasibility.

Incorporating  $f_{AC}$  and  $f_{ieq}$  improves the feasibility and robustness of the simulation solution  $X$ ; Incorporating  $f_{f(X)}$  improves the accuracy of  $X$ ; Incorporating  $f_{Lag}$  can optimize auxiliary tasks  $\lambda, \mu$  and  $Z$ . Hence, we arithmetically compose these objective functions into the loss function  $L$ (Eqn. 4).

$$L_{total} = L + L_{eqn} + L_{ieq} + L_{lag} + L_{f(X)} \quad (9)$$

TABLE II  
CONFIGURATIONS IN IEEE BUS SYSTEMS.

Problem size	14-bus	30-bus	57-bus	118-bus	300-bus
Buses	14	30	57	118	300
Generators	5	6	7	54	69
Branches	20	41	80	185	411
$\#\lambda$	29	61	115	237	601
$\#\mu(Z)$	48	166	142	452	876

The  $L_{total}$  efficiently combines supervised learning (with ground-truth labels) and unsupervised learning (without ground-truth labels) to guide the MTL training. By doing so, we maintain the prediction accuracy while increase the model feasibility and interpretability.

### VIII. EVALUATION

We evaluate our framework by examining its impacts on performance and simulation quality of power grid simulation. **Platform.** We conduct all experiments on an NVIDIA DGX-1 cluster with 16 nodes, and each node is equipped with two Intel Xeon E5-2698 v4 CPUs (40 cores running at 2.20GHz) and 8 NVIDIA TESLA V100 (Volta) GPUs. We use CUDA 10.1/cuDNN 7.0 [45] to run NNs on NVIDIA GPUs. We use Pytorch for model training and inference.

**Matpower.** Matpower 6.0 is an open-source Matlab power system simulation package [28], which is used widely in research and education for AC- and DC- power flow simulations. The default OPF solver, i.e., Matlab Interior Point Solver (MIPS), is a high-performance primal-dual interior-point solver.

**Load Sampling.** We sample the loads within  $[(1-t) \times P_{di}, (1+t) \times P_{di}]$  uniformly at random, where  $P_{di}$  is the default power load at the  $i$ -th bus, and  $t$  is the variation percentage, i.e., 10% in this paper, consistent with state-of-the-art [20], [22], [38].

**Input Datasets.** To comprehensively evaluate the performance, we use five power networks in Table II as test systems. We generate 10,000 input problems for each test system, in which 8,000 of them are for training and 2,000 for validation. These samples are fed into Matpower to produce the optimal solutions as the supervision ground truth signal. Uniform sampling is applied to avoid over-fitting issues common in generic DNN approaches [46].

#### A. Smart-PGSim Performance Evaluation

In our approach, we use Smart-PGSim to generate a high-quality initial condition for the MIPS solver, thereby drastically reducing the overall time-to-solution. We introduce the following performance metric to calculate the achieved speedups by Smart-PGSim:

$$SU = \frac{T_{MIPS}}{T_{MTL} + T'_{MIPS} + T_{MIPS} \times (1 - SR)} \quad (10)$$

where  $T_{MIPS}$  represents the solving time when using the traditional approach with MIPS,  $T_{MTL}$  represents the inference time of the MTL model, and  $T'_{MIPS}$  represents the convergence time in MIPS initializing with the output of Smart-PGSim.  $T_{MIPS} \times (1 - SR)$  calculates the restart execution time with the default initial point in MIPS if the simulation fails.

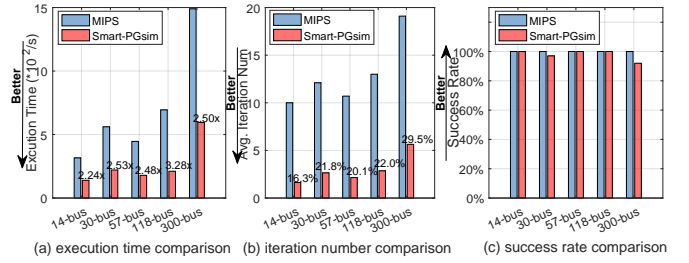


Fig. 4. Comparison of three aspects between MIPS and Smart-PGSim.

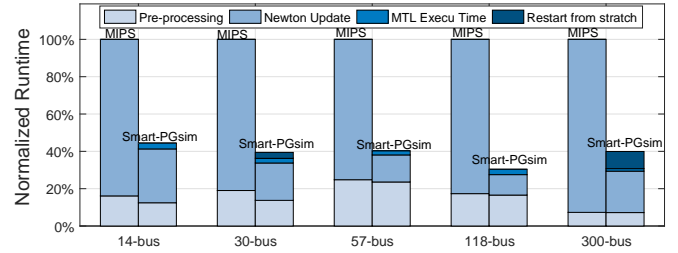


Fig. 5. Execution time breakdown

SR represents the overall success rate,  $SR = N_{suc}/N_{total}$ , where  $N_{suc}$  represents the number of problems successfully solved by MTL and  $N_{total}$  represents the total number of input problems. Whenever the initial condition provided by Smart-PGSim does not lead to the simulation converge successfully, we fall back to the traditional MIPS solver to guarantee the final convergence. Hence our method always provides 100% guarantee on simulation convergence, though it might come at an additional cost of re-executing overhead in the workflow.

Figure 4(a) compares the execution time of the traditional numerical simulation performed with MIPS and that of our framework in terms of the  $SU$  metric described above. Each test system is run on 2,000 input problems. Performance measurements of Smart-PGSim comprise the end-to-end runtime, including the time to produce the warm-start points in MTL, the convergence time in MIPS with the warm-start points, and the restart execution time in MIPS if the simulation fails. In the Figure 4(a), we also label the speedup at the top of Smart-PGSim bar. The Smart-PGSim speedups over MIPS observed in the plot are considerable, ranging from nearly a  $2.24 \times$  speedup up to over a  $3.28 \times$  speedup. Furthermore, the performance benefit of Smart-PGSim is more evident as the size of power networks increases, which indicates a notable potential in accelerating large-scale power grid systems. It is important to note that using Smart-PGSim as warm-start for MIPS generates the same solution as produced by MIPS directly. Figure 4(b) presents the average iteration number of MIPS and Smart-PGSim across different test systems, in which we only consider the iteration acceleration produced by Smart-PGSim. We measured the average iteration number during the convergence process until the terminate criteria are reached. The iterative process is the most computationally intensive part of the power grid simulation. We also label the ratio of the



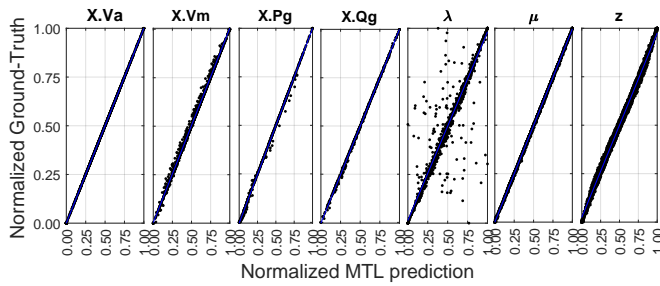


Fig. 6. Prediction accuracy of each feature used in the proposed MTL model.

Smart-PGSim iteration number to the MIPS iteration number on Smart-PGSim bar. The results in Figure 4(b) show that Smart-PGSim dramatically reduces the number of iterations required to converge, taking only 16.3% to 29.5% iterations of previous conduction (MIPS). The accelerated convergence drives the overall performance improvement of Smart-PGSim.

### B. Performance Breakdown

To further explore the performance improvement provided by Smart-PGSim, Figure 5 shows the runtime breakdown of MIPS and Smart-PGSim, normalized to the overall runtime where the problems run with MIPS. The pre-processing refers the execution time of problem construction and data preparing for power grid simulation, in which MIPS and Smart-PGSim show almost the same processing time. The Newton update represent the execution time spent in Newton iteration. Smart-PGSim have extra overheads about the inference time of the MTL model for generating warm-start solution and the restart time with failure cases. Note that we restart the failed cases with the default setting in the numerical solver MIPS to guarantee the final convergence. As Figure 5 depicts, Smart-PGSim is effective at reducing the time spent on the convergence, i.e., Newton Update. Smart-PGSim demonstrates significant performance improvement for the tested input problems despite the extra overhead introduced by the MTL model.

### C. Prediction Accuracy

Figure 4(c) compares MIPS and Smart-PGSim in terms of the success rate, in the case in which we do not restart Smart-PGSim after a failed execution. As we discuss above, Smart-PGSim guarantees *100% success rate in practice* by re-executing those computations that do not provide high-enough accuracy, while still considerably outperforming MIPS execution. The success rate is how many input problems can converge successfully in the simulation. Figure 4(c) reveals that Smart-PGSim leads to a high percentage of success rate in all case. Smart-PGSim provides 100% success rate on 14-bus, 57-bus, 118-bus while maintains a relatively high success rate as 97% and 92% on 30-bus, 300-bus respectively.

Figure 6 presents the prediction accuracy of each feature used in Smart-PGSim. We compare the accuracy of warm-start points predicted by Smart-PGSim with the exact solution in MIPS (Ground-truth). The prediction and ground-truth are normalized to the range [0,1]. The x-axis is the predicted value

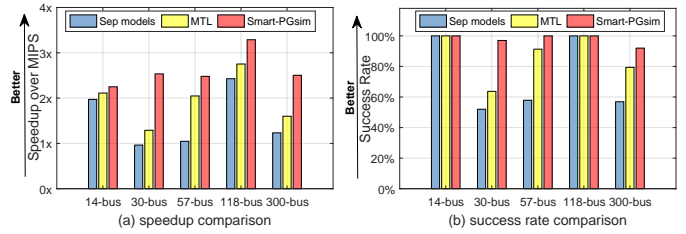


Fig. 7. Performance comparison

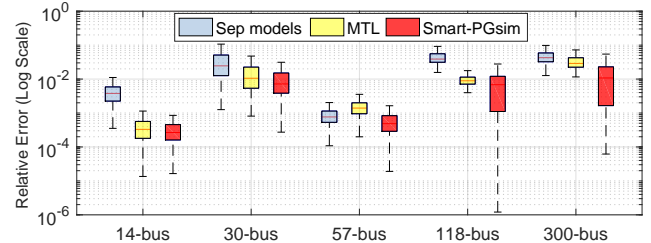


Fig. 8. Accuracy comparison.

of Smart-PGSim and the y-axis is the ground-truth value. If the prediction of Smart-PGSim is perfect, all points should lie on the  $y = x$  line. There is negligible accuracy lost in the prediction of  $X.Va$ ,  $X.Vm$ ,  $X.Pg$ ,  $X.Qg$ ,  $\mu$  and  $z$ . For  $\lambda$ , there is a larger variation in the predicted values representing over-prediction and under-prediction. Such variation in  $\lambda$  is acceptable because  $\lambda$  is the equality constraints factor in Eqn. 3, which will not affect the final convergence if the equality constraints are satisfied.

### D. Efficiency of Multitask Learning and Physical Constraints

In this section, we analyze the effectiveness of multitask learning and imposing physics constraints. First, we develop a model of multiple separate NNs without information sharing. For peer comparison, we use the same number of layers and neurons as MTL model in the multiple separated networks. Then, to show the efficiency of physical constraints, we remove physics constraints in MTL model as a comparison.

Figure 7(a) shows the speedup comparison with the multiple separated NNs, MTL model and Smart-PGSim. Here, “MTL” refers to the multitask learning model without physical constraints whereas “Smart-PGSim” refers the multitask learning model with physical constraints. Note that all speedup are measured with our performance metric  $SU$  in Eqn. 10. Figure 7 shows that the performance of the speedup  $SU$  and the success rate  $SR$  are significantly improved by the multitask learning and incorporation of the physical constraints. MTL provides notable speedup and success rate improvement over the multiple separated models, average speedup of  $1.36\times$  and 22.5% success rate improvement. In particular, the multiple separated NNs show inefficiency on the test system 30-bus with a  $0.96\times$  speedup, in which the 52.0% success rate produce a soaring overhead on restart. Adding the physical constraints further improve the speedup and success rate by 40% and 18.3% over MTL. In summary, our proposed framework Smart-PGSim, a

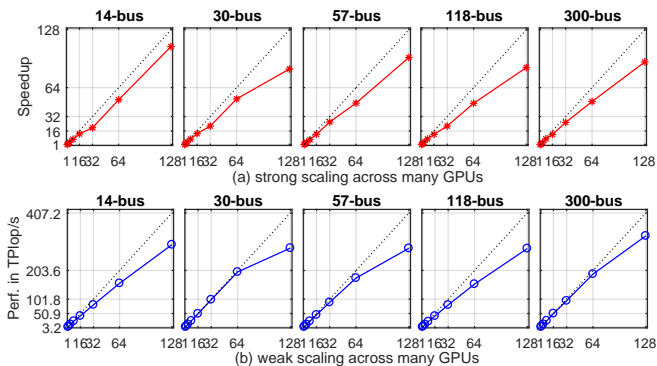


Fig. 9. Scaling across many GPUs

multitask model with physical constraints offers the highest average speedup and solution feasibility.

Moreover, Figure 8 presents box-plots<sup>2</sup> to show the results of the prediction accuracy in different models. We use relative error  $RE = |V_{predict} - V_{gt}|/V_{gt}$  to measure the prediction accuracy.  $V_{predict}$  refers the prediction values of MTL and  $V_{gt}$  refers the exact solution in MIPS (i.e., ground-truth). The lower relative error means the prediction is more accurate and closer to the ground-truth. We draw two observations from Figure 8: (1) The prediction provided by Smart-PGSim has the lowest average error with all five test systems; (2) Most of the predictions in Smart-PGSim is under the error line of  $10^{-2}$ , which shows Smart-PGSim consistently produces prediction within 1% relative error. These two observations reveal that Smart-PGSim can provide more *consistent acceleration* than multiple separate models and MTL, which is crucial for dealing with widely diversified input problems in real-time.

### E. Scalability Analysis on Multi-Node Systems

As we stated earlier, the AC-OPF problem is solved many times per day by power operators throughout the life of the power grid. Additionally, the real-life problem is further complicated by the uncertainty involved by equipment security, the reliability of alternative power sources (solar, eolic, and hydro power), and the stability of power generators and power transmission lines. Considering all those factors together is generally referred to solving Security-Constrained ACOPF (SC-ACOPF) [48], [49] and it originates very large and complex uncertain scenario trees that need to be analyzed to maintain the robustness of the global solution, i.e., an optimal solution that survives under all uncertain scenarios. From a computational perspective, these scenarios are largely independent (although some similarities can be exploited to reduce computational requirements) and result in a computational problem that is largely embarrassingly parallel and, thus, inherently scalable on parallel computers (e.g., assigning a batch of scenarios to each compute node, and then assigning a set of scenarios from the batch to each GPU).

<sup>2</sup>In the box-plots, the boxes are bounded by 25-th and 75-th percentiles of the variables; The central marks of the boxes indicate the median [47].

TABLE III  
PREDICTION PERFORMANCE COMPARISON.

Zamzam's [22]	Test system	–	39-bus	57-bus	118-bus	–
	SF	–	15.38×	9.49×	7.97×	–
$L_{cost}$	–	0.326%	0.457%	0.821%	–	–
Smart-PGSim	Test system	14-bus	30-bus	57-bus	118-bus	300-bus
	SF	21.17×	40.19×	21.72×	36.15×	105.64×
$L_{cost}$	0.007%	0.074%	0.040%	0.003%	0.008%	

While the focus of this work is mainly on accelerating each AC-OPF instance of a larger SC-ACOPF problem by providing high-quality initial conditions for the numerical solver, one can easily imagine that speedup similar to the ones reported in Section VIII-A can be expected for the SC-ACOPF problem. To verify such assertion, we conducted experiments on a 16-node compute cluster, where each node is an NVIDIA DGX-1 equipped with eight NVIDIA V100 GPUs (128 GPUs in total). We study both *strong scalability* and *weak scalability*. Strong scaling is measured with a fixed number of scenarios, while weak scaling linearly increases the number of scenarios with respect to the number of processors. Smart-PGSim is expected to generate an initial solution for each scenario. In these experiments, we use data parallelism for scaling out the Smart-PGSim workload, in which each GPU has an identical copy of the entire network and each computes results for a separate set (the local batch) of input scenarios. We focus on scaling Smart-PGSim, which emulates the use case where there are needs to generate initial solutions for a large number of scenarios for the SC-ACOPF problem.

Figure 9(a) shows strong scaling behavior for each of the five test systems with a fixed problem size (10k scenarios) from 1 to 128 GPUs. The black dotted lines in the plots represent ideal scaling for data parallelism. As expected, increasing the number of GPUs naturally leads to a higher speedup and shows an almost linear tendency. However, the speedup is not linear.

Such a non-linear speedup is caused by our work distribution strategy: While our distribution algorithm has been designed to equally distribute scenarios between GPUs, communication effects can skew this balance. Specifically, when running in a node with 8 GPUs, we first copy the MTL model and data to the first GPU device and then copy it to the other GPUs leveraging GPUDirect and NVLINK, which generates some load imbalance that translates into efficiency loss.

Figure 9(b) shows similar results for the weak scaling experiments, where the number of scenarios increases from 10k to 1,280k when increasing the number of GPUs from 1 to 128 (10k scenarios per GPU). The scalability shown in the plots for weak scaling is better, compared to that for strong scaling. This is because the weak scaling uses larger problems which amortizes the load imbalance problem, but we still notice similar issues as the strong scaling experiments.

Overall, Smart-PGSim scales up to 128 GPUs: for the test system of “300-bus” (the largest system we evaluated), we achieve a peak performance of 604.7 TFLOPS and a sustained

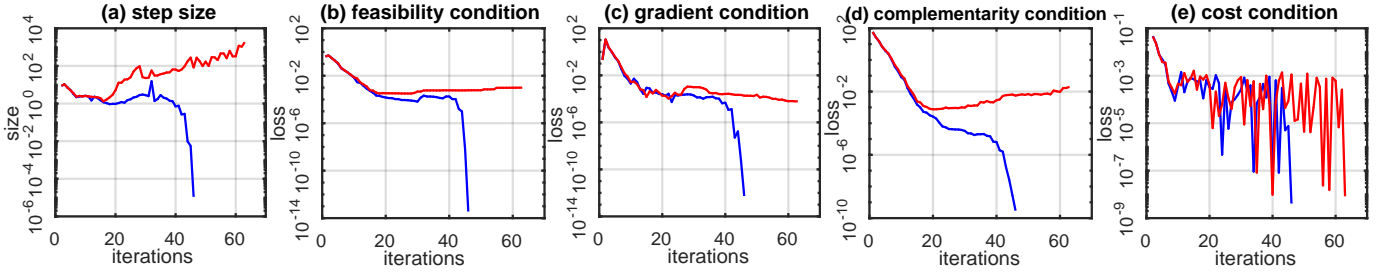


Fig. 10. The asymptotic convergence of the tracking loss along the iterations

performance of 326.1 TFLOPS, reaching 43% of the peak performance of Volta V100 (double precision).

### F. Comparison with Prior Work

Previous work [20], [22] use ML to directly replace the exact solver to achieve a high speedup. For a fair comparison with the state-of-the-art method, i.e., Zamzam et al.’s model [22] that leverages DNN for prediction, we assume that the prediction of Smart-PGSim is the final solution, effectively replacing the entire solving computation. In Table III, we compare performance and optimality loss to what has been used in Zamzam’s model. Cost deviation measures simulation quality. We donate a speedup factor (SF) to measure the computational improvements:  $SF = \frac{1}{n} \sum_{i=1}^n (T_i^{MTL} / T_i^{MIPS})$ , where  $T_i^{MTL}$  refers the execution time of the MTL and  $T_i^{MIPS}$  represents the execution time of the numerical solver (MIPS) for each input problem  $i$ . We measure the loss using the average fractional difference between the predicted cost  $C'$  and the true cost  $C$ :  $L_{cost} = \frac{100\%}{n} \sum_{i=1}^n |1 - C'_i / C_i|$ . The results presented in Table III show that our framework outperforms the state-of-the-art even in the case in which we directly use Smart-PGSim output as final solution of the computation. Smart-PGSim achieves an average  $44.97 \times$  speedup which provides 310.7% improvement comparing the average speedup of Zamzam’s model ( $10.95 \times$ ). Moreover, Smart-PGSim decreases the cost loss  $12.16 \times$  by average comparing with Zamzam’s model. Although Smart-PGSim show significant improvement over the state-of-the-art, we remark that the solutions produced by both models might not satisfy the strict requirements of power-grid simulations, hence our approach further refines Smart-PGSim output in the traditional solver MIPS at the back end and achieving high-quality solutions, although with reducing the speedup.

## IX. DISCUSSIONS

In this section, we discuss the generality of our techniques and analyze solving processes with and without convergence.

### A. Generality of Proposed Approach

The three major techniques, including sensitivity study (Section V), multi-tasking learning (Section VI), and incorporating domain knowledge (Section VII), can be broadly applied to many scientific HPC applications, and are not limited to the optimization problem in power grid simulations. In

this section, we highlight some potential application of our techniques to other scientific applications.

**Fluid dynamic simulation** aims to study the flow of fluid materials. Smart-fluidnet model [3] is a convolutional NN model to accelerate fluid simulation. We can build a multitask learning model by predicting the output velocity field  $\vec{u}$  as a main task and pressure field  $p$  as an auxiliary task since  $p$  impacts the fluid movement ( $\vec{u}$ ). Moreover, we can incorporate the incompressibility condition,  $\nabla \cdot \vec{u} = 0$ , as a physical constraint regularized as a soft constraint  $L_{loss} = \nabla \cdot \vec{u}$ .

**Molecular Dynamics (MD) simulation.** The DPMD model [50] is an NN model to accelerate MD simulation. This model can be enhanced by using the techniques presented in this work by developing a multi-task model where the main task predicts the potential energy and an auxiliary task predicts the symmetry-preserving descriptor. Also, the potential energy should be positive, which can be enforced as a hard constraint.

**Cosmology modeling.** CosmoFlow [6] is an NN model to predict three cosmological parameters that can be directly implemented as multi-task learning. The Cosmic Microwave Background [51] can be enforced as a hard constraint to bound the projection range of modeling.

### B. Analysis of Diverging Cases

The solving process for the AC-OPF problem can fail to converge. Figure 10 shows the inconvergence process given a bad initial solution and compares it with the convergence process given a good initial solution. Figure 10 shows the variance of step size and four convergence conditions across iterations. The step size  $|\Delta x|$  refers to the length of the updating step during the simulation; The four conditions are used to determine if the simulation is converged in each iteration.

Figure 10 shows that, for the case with bad initial solution, the step size rapidly increases. Accordingly, the four convergence conditions remain relatively stable without being able to converge. For the case with good initial solution, the step size and three conditions (feasibility, gradient and complementary) decrease quickly. We notice that the cost condition goes through great variance in both cases, which makes it difficult to correlate to convergence.

The step size is critical to determine the direction to explore to find the optimal solution. If the initial solution is bad, the solving process aims to use a larger step size to find a

promising direction. However, using a large step size could lead a failure of convergence (Figure 10.a).

As our results show, it is difficult to guess whether the numerical solver will converge based on the first iterations: both good and bad initial conditions behave similarly during the initial iterations of the power grid simulation and there is no clear indication that some computation will later fail. Given this complexity, we resort to re-initialize and re-execute the numerical solver from the beginning without employing the initial conditions generated by the MTL model. Overall, as our results demonstrate, even considering restart time, Smart-PGSim still significantly outperforms state-of-the-art solutions.

## X. CONCLUSIONS

Using NN to approximate and/or accelerate high performance computing applications has shown promising results. However, how to effectively apply a NN to those applications is still an open question. The approximations introduced by the NN models need to be carefully analyzed, so that the simulation quality in the application is not lost and even improved; at the same time, the execution time of the application should be reduced after applying NN. In this paper, we apply a NN to accelerate a specific power grid simulation problem, AC-OPF. As a simulation to solve complex nonlinear optimization problems based on iterative numerical methods, AC-OPF raises challenges on simulation robustness (i.e., ensuring the optimality of the simulation solution for various input problems) and respecting the physical constraints imposed by the power flow. We introduce a framework, Smart-PGSim, that facilitates the construction of a NN model by studying the impact of the output accuracy on simulation convergence and execution time and automatically imposing the physical constraints. Using a novel multitask-learning NN model generated by Smart-PGSim, we produce high-quality initial solutions for 10,000 input problems. Based on those solutions, the AC-OPF simulation reduces simulation time by an average of  $2.60\times$  (up to  $3.28\times$ ) without losing the optimality of the solution.

## XI. ACKNOWLEDGEMENT

This research is supported by the U.S. Department of Energy (DOE) Advanced Scientific Computing Research (ASCR), U.S. National Science Foundation (CNS-1617967, CCF-1553645 and CCF-1718194), award 74756, Co-design of Reconfigurable Accelerators for Sparse, Irregular Computations Underlying Machine Learning and Graph Analysis, award 66150, CENATE - Center for Advanced Architecture Evaluation, Chameleon cloud and XSEDE resource.

## REFERENCES

- [1] Alexander Radovic. Neutrino Identification with a Convolutional Neural Network in the NOvA Detectors. In *International Conference on High Energy Physics*, 2016.
- [2] Evan Racah, Christopher Beckham, Tegan Maharaj, Samira Kahou, Mr. Prabhat, and Chris Pal. ExtremeWeather: A Large-scale Climate Dataset for Semi-supervised Detection, Localization, and Understanding of Extreme Weather Events. In *NIPS*, 2017.

- [3] Wenqian Dong, Jie Liu, Zhen Xie, and Dong Li. Adaptive neural network-based approximation to accelerate eulerian fluid simulation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–22, 2019.
- [4] L Savoldi Richard, R Bonifetto, Stefano Carli, A Froio, A Foussat, and R Zanino. Artificial neural network (ann) modeling of the pulsed heat load during iter cs magnet operation. *Cryogenics*, 63:231–240, 2014.
- [5] P B. Wigley, P J. Everitt, Anton Hengel, John Bastian, M A. Sooriyabandara, Gordon McDonald, Kyle Hardman, C D. Quinlivan, Manju Perumbil, Carlos claiton Noschang kuhn, I R. Petersen, Andre Luiten, J Hope, N Robins, and Michael Hush. Fast machine-learning online optimization of ultra-cold-atom experiments. *Widley*, 6, 07 2015.
- [6] Amrita Mathuriya, Deborah Bard, Peter Mendygral, Lawrence Meadows, James Arneemann, Lei Shao, Siyu He, Tuomas Kärrnä, Diana Moise, Simon J Pennycook, et al. Cosmoflow: Using deep learning to learn the universe at scale. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 819–829. IEEE, 2018.
- [7] Prasanna Balaprakash, Romain Egele, Misha Salim, Stefan Wild, Venkatram Vishwanath, Fangfang Xia, Tom Brettin, and Rick Stevens. Scalable reinforcement-learning-based neural architecture search for cancer deep learning research. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–33, 2019.
- [8] Liu Yang, Sean Treichler, Thorsten Kurth, Keno Fischer, David Barajas-Solano, Josh Romero, Valentin Churavy, Alexandre Tartakovsky, Michael Houston, Mr Prabhat, et al. Highly-scalable, physics-informed gans for learning solutions of stochastic pdes. In *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*, pages 1–11. IEEE, 2019.
- [9] Zhengchun Liu, Tekin Bicer, Rajkumar Kettimuthu, Doga Gursoy, Francesco De Carlo, and Ian Foster. Tomogan: Low-dose x-ray tomography with generative adversarial networks. *arXiv preprint arXiv:1902.07582*, 2019.
- [10] Zhengchun Liu, Tekin Bicer, Rajkumar Kettimuthu, and Ian Foster. Deep learning accelerated light source experiments. In *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*, pages 20–28. IEEE, 2019.
- [11] Cong Liu, Jianhui Wang, and Jiaxin Ning. Optimal power flow (opf) in large-scale power grid simulation. In *FERC Conference*, pages 23–24. Citeseer, 2010.
- [12] Steven J Fernandez, Mallikarjun Shankar, James J Nutaro, Yilu Liu, Aleksandar D Dimitrovski, Olufemi A Omitaomu, Christopher S Groer, Kyle L Spafford, and Ranga R Vatsavai. Real-time simulation of power grid disruption, July 25 2013. US Patent App. 13/747,779.
- [13] Tsung-Hao Chen and Charlie Chung-Ping Chen. Efficient large-scale power grid analysis based on preconditioned krylov-subspace iterative methods. In *Proceedings of the 38th annual Design Automation Conference*, pages 559–562, 2001.
- [14] Y Huang, T Kashiwagi, and S Morozumi. A parallel opf approach for large-scale power systems. *IEEE*, 2002.
- [15] Yi Guo, David J Hill, and Youyi Wang. Nonlinear decentralized control of large-scale power systems. *Automatica*, 36(9):1275–1289, 2000.
- [16] James A Momoh and JZ Zhu. Improved interior point method for opf problems. *IEEE Transactions on Power Systems*, 14(3):1114–1120, 1999.
- [17] Savu C Savulescu. *Real-time stability in power systems: techniques for early detection of the risk of blackout*. Springer, 2014.
- [18] Cosmin G Petra, Olaf Schenk, and Mihai Anitescu. Real-time stochastic optimization of complex energy systems on high-performance computers. *Computing in Science & Engineering*, 16(5):32–42, 2014.
- [19] Ghulam Mohi Ud Din and Angelos K Marnierides. Short term power load forecasting using deep neural networks. In *2017 International Conference on Computing, Networking and Communications (ICNC)*, pages 594–598. IEEE, 2017.
- [20] Neel Guha, Zhecheng Wang, Matt Wytock, and Arun Majumdar. Machine learning for ac optimal power flow. *arXiv preprint arXiv:1910.08842*, 2019.
- [21] Kyri Baker. Learning warm-start points for ac optimal power flow. *arXiv preprint arXiv:1905.08860*, 2019.
- [22] Ahmed Zamzam and Kyri Baker. Learning optimal solutions for extremely fast ac optimal power flow. *arXiv preprint arXiv:1910.01213*, 2019.

- [23] Deepjyoti Deka and Sidhant Misra. Learning for dc-opf: Classifying active sets using neural nets. *arXiv preprint arXiv:1902.05607*, 2019.
- [24] Yeesian Ng, Sidhant Misra, Line A Roald, and Scott Backhaus. Statistical learning for dc optimal power flow. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–7. IEEE, 2018.
- [25] David E. Johnson, Johnny R. Johnson, John L. Hilburn, and Peter D. Scott. *Electric Circuit Analysis (3rd Ed.)*. Prentice-Hall, Inc., USA, 1997.
- [26] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992.
- [27] Hongye Wang. On the computation and application of multi-period security-constrained optimal power flow for real-time electricity market operations. *Cornell University*, 2007.
- [28] Ray D Zimmerman and Carlos E Murillo-Sánchez. Matpower 6.0 users manual. *PSERC: Tempe, AZ, USA*, 2016.
- [29] Jason M Cohen and Douglas B Page. System and method for economic dispatching of electrical power, April 15 1997. US Patent 5,621,654.
- [30] B. Stott, J. Jardim, and O. Alsac. Dc power flow revisited. *IEEE Transactions on Power Systems*, 24(3):1290–1300, Aug 2009.
- [31] R. D. Christie, B. F. Wollenberg, and I. Wangensteen. Transmission management in the deregulated environment. *Proceedings of the IEEE*, 88(2):170–195, Feb 2000.
- [32] A. A. Sousa, G. L. Torres, and C. A. Caizares. Robust optimal power flow solution using trust region and interior-point methods. *IEEE Transactions on Power Systems*, 26(2):487–499, May 2011.
- [33] Steven H. Low. Convex relaxation of optimal power flow part ii: Exactness. *IEEE Transactions on Control of Network Systems*, 1:177–189, 2014.
- [34] R. A. Jabr. Radial distribution load flow using conic programming. *IEEE Transactions on Power Systems*, 21(3):1458–1459, Aug 2006.
- [35] N. Chiang, C. G. Petra, and V. M. Zavala. Structured nonconvex optimization of large-scale energy systems using pips-nlp. In *2014 Power Systems Computation Conference*, pages 1–7, Aug 2014.
- [36] Miles Lubin, JA Julian Hall, Cosmin G Petra, and Mihai Anitescu. Parallel distributed-memory simplex for large-scale stochastic lp problems. *Computational Optimization and Applications*, 55(3):571–596, 2013.
- [37] Alfredo Vaccaro and Claudio Canizares. A knowledge-based framework for power flow and optimal power flow analyses. *IEEE Transactions on Smart Grid*, PP:1–1, 04 2016.
- [38] Xiang Pan, Tianyu Zhao, and Minghua Chen. Deepopf: A deep neural network approach for security-constrained dc optimal power flow. *arXiv preprint arXiv:1910.14448*, 2019.
- [39] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.
- [40] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global, 2010.
- [41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.
- [42] Baljinnayam Sereeter, Cornelis Vuik, and Cees Witteveen. On a comparison of newton–raphson solvers for power flow problems. *Journal of Computational and Applied Mathematics*, 360:157–169, 2019.
- [43] Baljinnayam Sereeter, Werner van Westering, Cornelis Vuik, and Cees Witteveen. Linear power flow method improved with numerical analysis techniques applied to a very large network. *Energies*, 12(21):4078, 2019.
- [44] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.
- [45] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.
- [46] Alex Gittens and Michael W Mahoney. Revisiting the nystrom method for improved large-scale machine learning. *The Journal of Machine Learning Research*, 17(1):3977–4041, 2016.
- [47] Robert Dawson. How significant is a boxplot outlier? *Journal of Statistics Education*, 19(2), 2011.
- [48] Naiyuan Chiang and Andreas Grothey. Solving security constrained optimal power flow problems by a structure exploiting interior point method. *Optimization and Engineering*, 16(1):49–71, 2015.
- [49] M. Schanen, F. Gilbert, C. G. Petra, and M. Anitescu. Toward multiperiod ac-based contingency constrained optimal power flow at large scale. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–7, 2018.
- [50] Denghui Lu, Han Wang, Mohan Chen, Jiduan Liu, Lin Lin, Roberto Car, Weile Jia, Linfeng Zhang, et al. 86 pflops deep potential molecular dynamics simulation of 100 million atoms with ab initio accuracy. *arXiv preprint arXiv:2004.11658*, 2020.
- [51] Peter AR Ade, N Aghanim, M Arnaud, Mark Ashdown, J Aumont, C Baccigalupi, AJ Banday, RB Barreiro, JG Bartlett, N Bartolo, et al. Planck 2015 results-xiii. cosmological parameters. *Astronomy & Astrophysics*, 594:A13, 2016.