



Delft University of Technology

Exploring extrapolation of machine learning models for power system time domain simulation

Arowolo, Olayiwola; Stiasny, Jochen; Cremer, Jochen

DOI

[10.1016/j.segan.2025.101908](https://doi.org/10.1016/j.segan.2025.101908)

Publication date

2025

Document Version

Final published version

Published in

Sustainable Energy, Grids and Networks

Citation (APA)

Arowolo, O., Stiasny, J., & Cremer, J. (2025). Exploring extrapolation of machine learning models for power system time domain simulation. *Sustainable Energy, Grids and Networks*, 43, Article 101908. <https://doi.org/10.1016/j.segan.2025.101908>

Important note

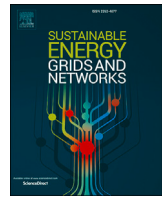
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright



Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.



Exploring extrapolation of machine learning models for power system time domain simulation[☆]

Olayiwola Arowolo^{a,*} , Jochen Stiasny^{a,b}, Jochen Cremer^{a,b} 

^a Electrical Sustainable Energy, Delft University of Technology, Delft, the Netherlands

^b Austrian Institute of Technology, Vienna, Austria

ARTICLE INFO

Keywords:

Time domain simulation
Power system security
Machine learning
Extrapolation
Transfer learning

ABSTRACT

Time domain simulation (TDS) is an important tool for assessing power system security under various disturbances. However, its computational cost limits the number of disturbances that can be assessed. The need for fast assessment of numerous disturbances has increased with the rapid integration of renewable energy sources. Machine learning (ML) methods have been explored to accelerate power system TDS, but these methods are studied in interpolation scenarios, where they predict outputs for inputs within the training data distribution. This work uses a state-of-the-art ML model to explore the extrapolation behavior of ML models for TDS. First, we highlight the importance of ML models' extrapolation capacity for fast assessment of numerous diverse disturbances. Next, we demonstrate that extrapolation for discrete disturbances is more challenging than for continuous disturbances. Subsequently, we investigate how transfer learning (TL) may be used to improve the performance of ML models in TDS extrapolation scenarios. Finally, we outline the limitations of TL for power system TDS and suggest alternative approaches for developing ML models with better extrapolation performance in TDS applications.

1. Introduction

The global power grid faces significant operational challenges due to the integration of renewable energy resources and changing demand patterns. Power system operators must ensure grid security and reliability amidst these changes [1]. Time domain simulation (TDS), which involves solving non-linear differential-algebraic equations (DAEs), is the reference method for assessing the dynamic security of the power grid under various contingencies [2]. Commercial software packages use implicit integration methods, like the trapezoidal method, for solving power system DAEs because of their accuracy and numerical stability. However, implicit integration methods require small step sizes, imposing long computation times for operators [3]. Conventionally, the long computation time of TDS limits security assessments to a few 'credible' contingency scenarios. However, the increasing integration of renewable energy sources renders the current contingency selection sub-optimal. Therefore, new methods are needed to optimize contingency scenario selection or reduce the computation time of TDS for assessing a much larger set of scenarios [4]. We focus here on the latter methods.

Machine learning (ML) methods have been explored for predicting power system dynamics promising significant speedup after training. Most works applying ML for dynamic security assessment (DSA) focus on classifying the system's response to a contingency as stable or unstable [5,6]. Others provide a confidence interval or stability margin in addition to the classifier output [7,8]. However, operators may need to check the evolution of system states and ensure they do not exceed prescribed thresholds. Binary security prediction is insufficient to make important control decisions such as where controlled load outages should be performed to prevent loss of system stability [9]. Thus, TDS may become necessary. We refer interested readers to [10] for more details on the application of ML to DSA. Neural-network-based architectures such as Long Short Term Memory (LSTM) [11], have been explored for predicting state trajectories in power systems. However, these approaches are limited by their dependence on large training datasets. The training datasets must be generated using computationally expensive numerical solvers. Physics Informed Neural Networks (PINNs) attempt to address this limitation by incorporating the physical equations of the system into the learning process, with the potential to reduce the need for large training datasets. PINNs have shown promise in accelerating TDS for small

[☆] This work was funded by the Dutch Research Council, Veni Talent Program Grant 19161.

* Corresponding author.

Email addresses: o.a.arowolo@tudelft.nl (O. Arowolo), j.b.stiasny@tudelft.nl (J. Stiasny), j.l.cremer@tudelft.nl (J. Cremer).

test systems [12–15]. However, it remains challenging to train large networks because of inaccuracies in long-term integration [16] and gradient pathologies inherent to PINNs [17]. While ML approaches show the potential for accelerating power system TDS, their fundamental limitation in extrapolation settings remains understudied.

We define extrapolation as the case where the data distribution during operation differs from the training data distribution and interpolation as the case where the training data distribution is representative of the data during operation [18]. ML models optimize their parameters based on the training data distribution, producing reliable predictions only when the test data has the same distribution as the training data [18,19]. This has led to the existing paradigm of minimizing generalization error evaluated with the test dataset (interpolation). To ensure the ML model always performs interpolation during operation, one would need a high-dimensional training dataset that includes all possible parameter variations (e.g. line configurations or loading conditions). It is here we run into the curse of dimensionality. As the dimensionality of the ML model input increases, the number of data samples needed to train the model increases exponentially, thus the learning problem quickly becomes intractable. However, if ML models could extrapolate reliably to unseen inputs, that is, the model error for operating conditions (OCs) and system topologies (STs) outside the support of the training data distribution is of the same order of magnitude as within-training distribution errors, in that case, the number of training samples can remain tractable as a training dataset that covers all possible parameter variations is no longer required. This is illustrated in Fig. 1.

For power system TDS applications, good extrapolation refers to the flexibility of the ML model to accurately predict the system's response to numerous scenarios (e.g. different fault types, network topologies, and initial conditions) outside the training data distribution. Existing works on ML for TDS typically assume fixed initial conditions and fixed network topology to make the learning problem tractable. These assumptions limit the dimensionality of the model input to a few variables but restrict the learning problem to only specific scenarios. Such models cannot assess many diverse scenarios, thus trading flexibility for improved speed. Consequently, the speed-flexibility trade-off of these ML approaches conflicts with the system operator's goal of faster TDS tools for more comprehensive contingency assessments. Transfer learning is a common method for adapting ML models to changing input distributions and it has been studied extensively for DSA classification [20,21] in power systems. However, to the best of the authors' knowledge, it has

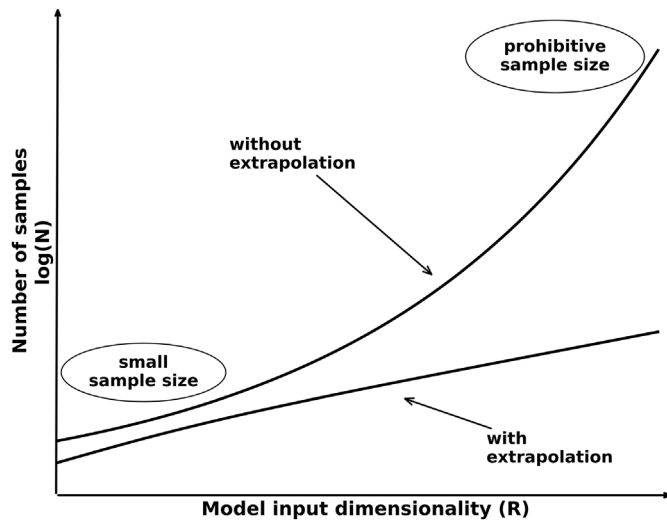


Fig. 1. The number of samples needed for training the machine learning model increases exponentially as the dimension of the input features increases.

not been applied to ML models for power system TDS. Without the capacity to extrapolate reliably, the utility of ML models for TDS applications is severely limited.

To this end, we explore the extrapolation behavior of ML models for power system TDS by analyzing the performance of an ML model in an extrapolation setting for continuous and discrete disturbances. In this study, we use the deep operator network (DeepONet), a state-of-the-art ML model for predicting the trajectories of dynamical systems. [4,22,23] have considered operator networks for approximating power system dynamics. However, these works consider the problem of TDS as a function regression where pre-fault and on-fault trajectories of the power system are sampled and used to predict the post-fault trajectories. Assuming the availability of pre-fault and on-fault trajectories works well for offline simulation but becomes challenging in an on-line setting. In this work, we only assume the availability of post-fault initial conditions. We use the DeepONet model to learn a mapping from the initial condition to the full post-fault trajectory. Moreover, the function regression approach can have low generalization capacity and requires recursive prediction. Here, we have considered how DeepONet may be applied more generally for power system TDS by taking initial system conditions as input and producing the state solution trajectories as output similar to numerical solvers. While we have considered DeepONet in this study, our analysis can be easily applied to other existing ML approaches for power system TDS. We briefly outline our contributions:

1. We investigate the challenge of extrapolating ML models for TDS to contingencies outside the training data distribution. We also investigate how common assumptions used to constrain the learning problem compromise the goal of assessing more contingency scenarios with ML.
2. We demonstrate that it is more difficult for ML models to extrapolate to discrete disturbances than to continuous disturbances. We assert that this is a critical limitation as discrete disturbances are more severe.
3. We examine how transfer learning may be used to improve the extrapolation performance of ML models for TDS. We also highlight the limitations of transfer learning for TDS and outline alternative approaches for better extrapolation performance.

The rest of the paper is organized as follows: Section 2 introduces the problem formulation of ML for power system TDS. Section 3 briefly introduces the DeepONet model. In Section 4, we carry out numerical case studies. Section 5 discusses the results from the case studies and highlights future directions. Section 6 concludes the paper.

2. ML for power system TDS

This section describes how a neural network can learn the power system TDS problem through supervised learning. Power system dynamics can be described by a set of non-linear differential-algebraic equations (DAEs) written as:

$$\frac{dx}{dt} = f(x, u, y) \quad (1a)$$

$$0 = g(x, u, y) \quad (1b)$$

where $x \in \mathbb{R}^a$ are the initial conditions of the differential states, $u \in \mathbb{R}^b$ are control inputs and $y \in \mathbb{R}^c$ are the algebraic states. The set of equations can be written more compactly in mass matrix form as:

$$M \frac{dx}{dt} = f(x(t), u) \quad (2)$$

where M is a diagonal matrix and $x = [x, y]^T \in \mathbb{R}^{a+c}$. The reference solution is obtained by discretizing the time domain $t = \{t_0, t_1, \dots, t_{max}\}$

and performing numerical integration:

$$x(t) = M^{-1} \left[\int_0^t f(x(t), u) dt \right] \quad (3)$$

To simplify the notations, let the right-hand side of Eq. (3) be $F(x)$, and let $x(t) = y$ for $t > 0$. As there is no analytical solution for Eq. (3), we aim to approximate the solution of the equation, $F(x)$, by a neural network $\tilde{F}(x)$. Note that $\tilde{F}(x)$ can be any suitable neural network model (DNN, PINNs or DeepONet).

We now formulate a classic supervised learning problem to train the network. Assuming $\tilde{F}(x)$ is the functional hypothesis of the chosen neural network, let $\{(x_i, y_i)\}_{i=1}^n \subset D$ be independent and identically distributed (iid) samples randomly drawn from the joint probability distribution D . The expected loss of $\tilde{F}(x)$ is

$$R(\tilde{F}) = \mathbb{E}_{(x,y) \sim D} [L(y, \tilde{F}(x))] \quad (4)$$

where L can be the mean squared error loss:

$$L = |y - \tilde{F}(x)|^2 \quad (5)$$

As the underlying joint distribution D is not known, we minimize the empirical risk with n samples:

$$\hat{R}(\tilde{F}) = \frac{1}{n} \sum_{i=1}^n L(y_i, \tilde{F}(x_i)) \quad (6)$$

This work assumes that DeepONet represents $\tilde{F}(x)$. The universal approximation theorem of neural networks and operators [24] implies that:

$$|R(\tilde{F}) - \hat{R}(\tilde{F})| < \epsilon \quad \text{for any } \epsilon > 0 \quad (7)$$

This means a sufficiently parameterized neural network will have a low generalization error provided a new sample x_i comes from D . However, there is no guarantee a new sample x_i will come from the same distribution as the training samples in the case of power system disturbances. This can lead to large extrapolation errors in the neural network. To define the extrapolation error, we define a conditional distribution P , such that: $\{\|a - b\| < \epsilon \mid a \in D, b \in P\} \cap \{P \cap D = \emptyset\}$, the extrapolation error is the expected loss of $\tilde{F}(x)$:

$$\mathbb{E}_{(x,y) \sim P} [L(F(x), \tilde{F}(x))] \quad (8)$$

Similarly, this error is estimated empirically by sampling $\{x_i, y_i\} \subset P$. Since there are no established theoretical guarantees on the extrapolation error of neural networks, we empirically investigate the extrapolation error of an example machine learning model for time domain simulation (DeepONet).

3. Deep operator network for TDS

The deep operator network (DeepONet) was proposed by Ref. [25] for approximating continuous non-linear operators mapping two infinite dimensional spaces such as function-to-function mapping. In an operator learning setting, if $u(x)$ is an input function and $G(u)$ is an operator taking $u(x)$ as input, $G(u)(t)$ is a real number output of the operator at point t in the output domain. According to the universal approximation theorem for non-linear operators [26], a neural network with 1 hidden layer can approximate any non-linear continuous operator with arbitrary accuracy. We consider the DeepONet model to approximate the operator mapping initial post-fault conditions to full post-fault trajectory. Our choice of DeepONet is informed by its ability to handle multiscale input and output domains. This is particularly useful in power systems where transient responses for different events can have different time scales. DeepONet and its variants have also been shown to achieve state-of-the-art performance in predicting non-linear dynamics [25]. As shown in

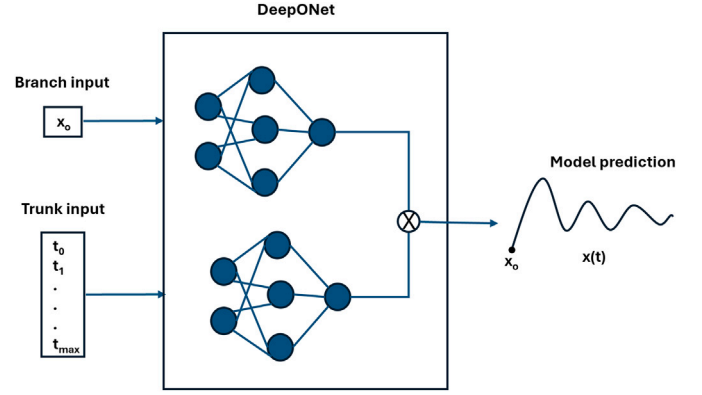


Fig. 2. DeepONet architecture has two separate neural networks for the initial conditions and time inputs respectively, the outputs of both networks are combined with a dot product.

Fig. 2, DeepONet consists of two neural networks: the branch and the trunk network. The branch network takes in the initial conditions and control inputs $u(x)$, discretized at a fixed number of points as input, and returns a set of outputs $[b_1, b_2, \dots, b_z] \in \mathbb{R}^z$. The branch network is a fully connected neural network that applies non-linear transformations to $u(x)$ to obtain embeddings $[b_1, b_2, \dots, b_z]$. The embeddings can be viewed as coefficients of a set of basis functions. The trunk network is a fully connected neural network that takes the time t , at which the output function is to be queried as input and returns $[\tau_1, \tau_2, \dots, \tau_z]$, which describes a basis function. The trunk network decomposes the target response into a set of non-linear basis functions conditioned on $u(x)$. This is similar to a Fourier series for periodic functions:

$$\psi(x) = a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + d_n \sin(nx) \quad (9)$$

where a_0 , a_n , and d_n are coefficients of the sine and cosine basis functions. To get the output of the DeepONet, the function $G(u)(t)$ may be obtained from its constituent basis functions by:

$$G(u)(t) = \int_{k=1}^z b_k \cdot \tau_k dt \quad (10)$$

This may be simply interpreted as an inner product. Hence, the output of the DeepONet is obtained by taking a dot product of the outputs of the branch and trunk networks. This can be written as:

$$\tilde{F}(x_0, u)(t) = \sum_{k=1}^z [b_k] \cdot [\tau_k] \quad (11)$$

where z is the number of output neurons per state variable. Note that \tilde{F} approximates the entire solution trajectory in one step as $t = \{t_0, t_1 \dots t_{\max}\}$ corresponds to the time discretization used to obtain the reference numerical solution.

The representational power of DeepONet to learn complex non-linear power system dynamics comes from its explicit decomposition of the target function into a set of parameterized basis functions.

4. Case study

4.1. System settings

We perform all experiments on the IEEE 9 bus system. The network admittance matrix is Kron reduced to a 3x3 matrix where we consider the loads to be constant impedance. We model each generator in the

Table 1
Generator parameters in p.u.

Gen	H	D	X_d	X'_d	X_q	X'_q	P_m
1	23.64	80.364	0.146	0.0608	0.0969	0.0608	0.710
2	6.4	40.28	0.8958	0.1198	0.8645	0.1198	1.612
3	3.01	0.903	1.3125	0.1813	1.2578	0.1813	0.859

network using the two-axis generator model from Ref. [27]:

$$\begin{bmatrix} T'_{d0} \\ T'_{q0} \\ 1 \\ 2H \end{bmatrix} \frac{d}{dt} \begin{bmatrix} E'_q \\ E'_d \\ \delta \\ \Delta\omega \end{bmatrix} = \begin{bmatrix} -E'_q - (X_d - X'_d) I_d + E_{fd} \\ -E'_d + (X_q - X'_q) I_q \\ 2\pi f \Delta\omega \\ P_m - E'_d I_d - E'_q I_q - (X'_q - X'_d) I_d I_q - D\Delta\omega \end{bmatrix} \quad (12)$$

$$\begin{bmatrix} I_d \\ I_q \end{bmatrix} = \begin{bmatrix} R_s & -X'_q \\ X'_d & R_s \end{bmatrix} \begin{bmatrix} E'_d - V \sin(\delta - \theta) \\ E'_q - V \cos(\delta - \theta) \end{bmatrix} \quad (13)$$

We reduce the generator models to a classic model by setting X_q and $X'_q = X'_d$ and making E'_q and E'_d constants. At each generator bus, we obtain the state variables $x = [\delta, \Delta\omega, I_d, I_q, V \text{ and } \theta]$. The state variables are represented in per-unit values. The values for the generator constants are shown in Table 1. To solve the system's DAE, we generate ground truth data using the IDA solver from the Assimulo [28] Python library. We set the solver's absolute tolerance to 10^{-8} and solve each scenario for 2 s using a step size of 0.01 s. We assume the system is initialized from a steady state for all scenarios and we initialize the system states using power flow solutions. In each experiment, we use the mean squared error as the loss function for model training

$$\mathcal{L}_{\text{mse}} = \frac{1}{n} \sum_{i=1}^n |x_i(t) - \hat{x}_i(t)|^2 \quad (14)$$

We report the percentage maximum absolute error for each state to compare them:

$$\max \text{AE} = \max |x_i(t) - \hat{x}_i(t)| \quad (15)$$

As there are three samples of each state for the three generators in the system, we aggregate similar states across multiple model runs and report the percentage maximum absolute error

$$\max \text{AE} = \max_{i \in S, j \in N} (|x_i(t) - \hat{x}_j(t)|) \quad (16)$$

where S is the number of states and N is the number of times the model is run.

4.2. Model implementation

We use the same DeepONet architecture and settings for each experiment. The branch network consists of 4 hidden layers with 360 neurons each. The trunk network also has 4 hidden layers with 360 neurons each. The two networks use a modified multi-layer perceptron (MLP) as proposed by Ref. [17]. The modified MLP has achieved higher accuracy than the standard MLP in non-linear differential equation problems. We use a pointwise sine activation function and initialize the model parameters using the Xavier-Glorot scheme. We train the model with Adam optimizer [29] and a learning rate of $1e^{-3}$ with exponential decay by a factor of 0.998 every 5000 steps. We train the model for $1e^6$ optimizer steps and save the model weights corresponding to the lowest loss on

the validation data to aim for low generalization error. The model is implemented using Python and Jax framework. All experiments were performed on a Linux 4.18 server computer with 32GB CPU RAM and NVIDIA A30 GPU with 24GB RAM. All experiments were repeated 5 times with a random model initialization and a random dataset split into training, validation, and testing sets each time. We report the average metrics except for those stated otherwise.

We generate output trajectories for the state variables using the Assimulo IDA solver in each experiment. We generate 12,000 samples using 10,000 samples for training, 1,000 for validation and 1,000 for testing. To analyse the model's extrapolation capacity on samples outside the training distribution, we also generate 1,000 samples for the extrapolation dataset. We perturb the system with random changes in each generator's power injection for experiments with continuous disturbances. We randomly sample power injection changes between $\pm 50\%$ of the initial steady-state generator power injection from a uniform distribution, for the training and test datasets. For the extrapolation dataset, we randomly reduce each generator's power injection between -50% and -70% of its initial power injection.

In the experiment with discrete disturbances, we consider line outages as discrete disturbances. We randomly select one transmission line for removal. We only consider lines that do not cause islanding of any part of the network. As the IEEE 9 bus system only has 9 transmission lines, we keep lines $\{1, 2, 3\}$ in place because their removal will lead to the islanding of a generator. We select from lines $\{4, 5, 6, 7\}$ for removal in the training, validation and test datasets. We select from lines $\{8, 9\}$ for removal in the extrapolation dataset. All codes for the experiments are publicly available.¹

4.3. Within distribution performance

This case study investigates the model's performance for disturbances from the same distribution as the training data. We consider three settings. 1) System with fixed initial conditions and continuous disturbance 2) System with variable initial conditions and continuous disturbance 3) System with variable initial conditions and discrete disturbance.

4.3.1. Fixed initial conditions and continuous disturbance

Here, we assume the values of system states are fixed before any disturbance is applied. Hence, the system always starts from the same initial condition. The change in the generator power injection, $\Delta P \in \mathbb{R}^3$, is the only change to the system's steady-state conditions. Other system parameters are fixed. Therefore, ΔP is the input to the branch subnetwork. The input to the trunk subnetwork is a vector of all time steps $t \in \{0.0, 2.0\}$ with time steps of 0.01 s. The model outputs the full trajectories of all six states for each generator. We use a 2D t-distributed Stochastic Neighbor Embedding (t-SNE) plot to visualize model input in Fig. 3. A t-SNE plot is a data visualization and analysis tool used to visualize complex, high-dimensional distributions in two or three dimensions. The plot shows that the training/testing data distribution and the extrapolation data distribution do not overlap. We conclude from the plot that prediction for a sample in the testing dataset is a form of high-dimensional interpolation, which differs from extrapolation. The trained model achieves low generalization error on the test dataset with a maximum absolute error of 0.10 for the worst-case state variable as shown in Table 2. A comparison of the model's prediction and the ground truth trajectory for an example in the test dataset is shown in Fig. 4. The predicted trajectories match well with the ground truth data for all state variables.

4.3.2. Variable initial conditions and continuous disturbances

Most existing works assume a fixed initial condition for the system. The assumption of a fixed initial condition simplifies learning but limits the model's utility. Here, we consider the more challenging scenario

¹ [GitHub Repository](#).

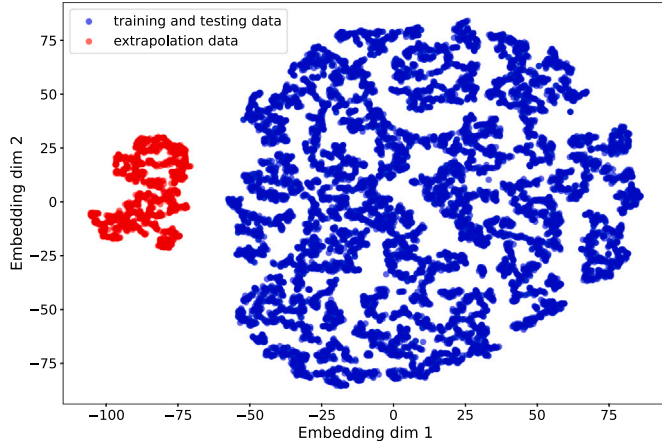


Fig. 3. Model inputs for the system with fixed initial condition and continuous disturbance shown in 2D. The extrapolation dataset does not overlap with the training/testing dataset.

of a variable system state, where the initial condition is sampled from a distribution. The assumption of an initial steady-state condition remains. We randomly sample initial voltage magnitude V , at generator buses from a uniform distribution $\mathcal{U} \sim (0.95, 1.05)$, the voltage angle at bus 1, θ_1 is sampled from $\mathcal{U} \sim (1.0, 3.0)$, θ_2 is sampled from $\mathcal{U} \sim (18.0, 20.0)$ and θ_3 is sampled from $\mathcal{U} \sim (12.0, 14.0)$. The power injections and other state variables are initialized from the power flow solution. The system is perturbed by randomly changing the initial power injection ΔP as in the previous experiment. The model input combines the initial power injection, change in power injection and initial states. This increases the dimensionality of the input from \mathbb{R}^3 to \mathbb{R}^{24} . The trained model achieves low generalization error on the test data. The predicted trajectories match the ground truth trajectories as shown in Fig. 5. The maximum absolute error is 0.03 for the worst-case state variable as shown in Table 2. We also provide mean absolute percentage error (MAPE) for all experiment settings in Table 3.

4.3.3. Discrete disturbance

Line outages are considered one of the most severe faults to the power system and the system is conventionally designed to withstand

the sudden outage of any one line (N-1 security). Here, we consider line outages as discrete disturbances and assume the system's initial condition is variable, as in the previous experiment. To generate a real-number representation of the system topology, we extract the equivalent line resistances and reactances of the Kron-reduced admittance matrix. We then reinitialize the state variables and simulate the evolution of the state variables in response to the fault. Fig. 6 shows the t-SNE plot of the model inputs for the training/testing and extrapolation datasets. Note that the plot shows the discrete nature of the data distributions. The training/testing dataset shows four separate distributions corresponding to the four network topologies in the dataset, while the fifth distribution corresponds to the extrapolation dataset.

Thus, the model may learn a unique representation of data from each discrete distribution/topology. Similarly to previous experiments, the trained model achieves low generalization error on the test dataset with a maximum absolute error of 0.01 for the worst-case state variable as shown in Table 2. A comparison of the model's predicted trajectories and the ground truth trajectories for a sample in the test dataset is shown in Fig. 7. The predicted trajectories match the ground truth data for all state variables.

4.4. Out-of-distribution performance

This case study investigates the model's performance for disturbances outside the training distribution. We evaluate the model's performance with the extrapolation datasets. We consider the same three settings as in the previous case study.

4.4.1. Fixed initial conditions and continuous disturbance

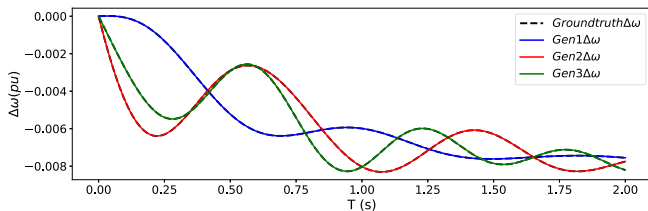
The model errors for the extrapolation dataset are larger than the errors for the test dataset. Note that the model predictions in the extrapolation setting still match the ground truth trajectories quite well with some discrepancies as shown in Fig. 8. This suggests the model learns useful representations beyond the training data distribution. The maximum absolute error on the extrapolation dataset remains comparable to the maximum absolute error on the test dataset as shown in Table 2.

4.4.2. Variable initial conditions and continuous disturbances

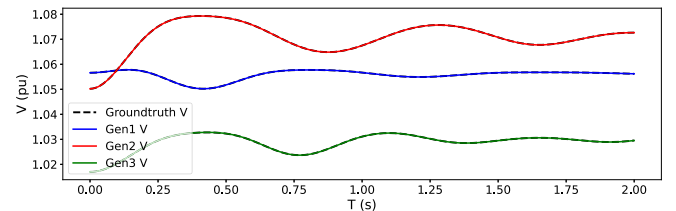
Here, we also observe that model predictions still match the ground truth trajectories quite well with some discrepancies as shown in Fig. 9. Notably, despite the more challenging setting, the trained model achieves comparable performance on the test and extrapolation datasets,

Table 2
Model error metrics within and outside training distribution.

Setting	No of training samples	Max absolute error within-distribution						Max absolute error out-of-distribution					
		δ	$\Delta\omega$	I_d	I_q	V	θ	δ	$\Delta\omega$	I_d	I_q	V	θ
1	10000	0.09	$6e^{-4}$	0.10	0.10	$1e^{-2}$	0.09	0.12	$7e^{-4}$	0.12	0.08	$9e^{-3}$	0.11
2	10000	0.03	$7e^{-5}$	0.01	0.01	$1e^{-3}$	0.03	0.15	$6e^{-4}$	0.11	0.07	$7e^{-3}$	0.15
3	10000	0.01	$1e^{-4}$	0.00	0.01	$1e^{-4}$	0.01	4.14	$2e^{-2}$	1.29	1.76	$8e^{-2}$	4.22

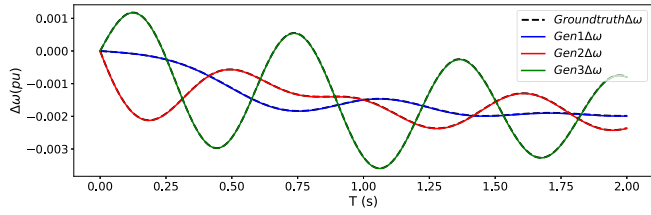


(a) Frequency deviation trajectory

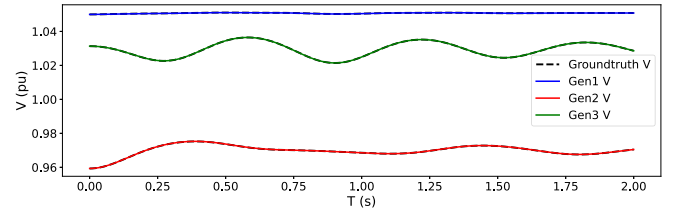


(b) Voltage trajectory

Fig. 4. Predicted trajectories and ground truth trajectories for a representative sample in the test dataset for a system with fixed initial condition and continuous disturbance.



(a) Frequency deviation trajectory



(b) Voltage trajectory

Fig. 5. Predicted trajectories and ground truth trajectories for a representative sample in the test dataset for a system with variable initial condition and continuous disturbance.

Table 3

Mean absolute percentage error (MAPE) within and outside training distribution.

Setting	No of training samples	MAPE within-distribution						MAPE out-of-distribution					
		δ	$\Delta\omega$	I_d	I_q	V	θ	δ	$\Delta\omega$	I_d	I_q	V	θ
1	10000	0.25	$7e^{-5}$	0.08	0.07	$7e^{-4}$	0.28	1.21	$6e^6$	$9e^2$	1.14	0.04	3.94
2	10000	1.33	$7e^{-5}$	0.18	0.20	$3e^{-3}$	0.99	1.73	$4e^6$	4.87	0.76	0.03	1.97
3	10000	1.08	$4e^{-5}$	0.53	0.06	$4e^{-4}$	1.98	$1e^4$	$1e^8$	$5e^2$	$5e^2$	1.04	$1e^4$

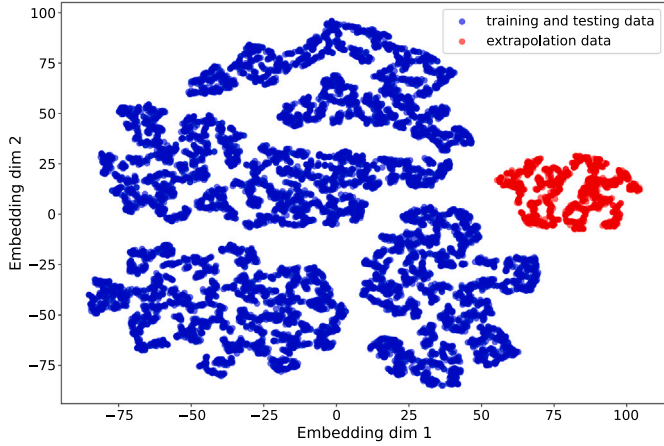
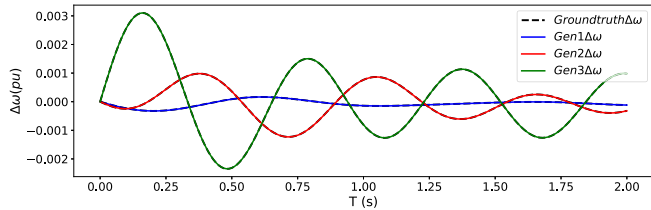
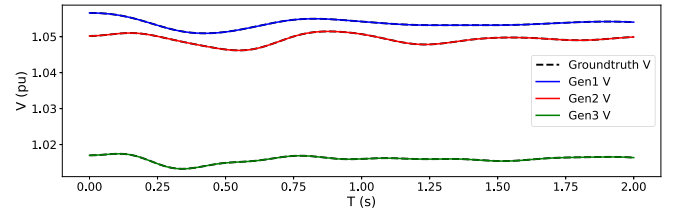


Fig. 6. Model inputs for the system with variable initial condition and discrete disturbance shown in 2D. Separate distributions highlight the different topologies in the dataset.

using the same number of training samples and roughly equivalent number of model parameters as in the previous experiment. The model error on the extrapolation dataset is also greater than the error on the test dataset as shown in Fig. 11.



(a) Frequency deviation trajectory



(b) Voltage trajectory

Fig. 7. Predicted trajectories and ground truth trajectories for a representative sample in the test dataset for a system with discrete disturbance.

4.4.3. Discrete disturbances

Large discrepancies between the predicted and ground truth trajectories can be observed in Fig. 10. The model's prediction can thus lead to the wrong security conclusions in an out-of-distribution setting. The model error on the extrapolation dataset is orders of magnitude greater than the error on the test dataset as shown in Fig. 13.

4.5. Transfer learning

We investigate how transfer learning may be used to improve extrapolation performance. The model is trained on the training dataset and fine-tuned with 'sparse' samples from the extrapolation (new) distribution. To this end, we generate 100 new samples from the extrapolation distribution. We saved the weights of the trained model. We then fine-tuned the model by initializing from the saved weights and training on the original data augmented with 'sparse' samples from the extrapolation data distribution for relatively few optimizer iterations ($1e^4$). We avoid fine-tuning the model with only a few new samples to prevent overfitting the model on the new samples, which can cause catastrophic forgetting [30]. Instead, we fine-tuned the model with the augmented training data. We analyse the impact of transfer learning on extrapolation performance under the following two scenarios.

4.5.1. Continuous disturbance

Here, we consider the system with variable initial conditions and a continuous disturbance. First, we compare the model's performance on the test and extrapolation datasets before transfer learning. Fig. 11

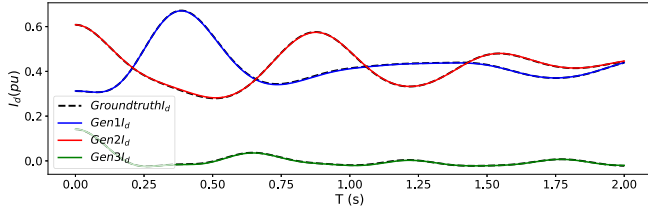
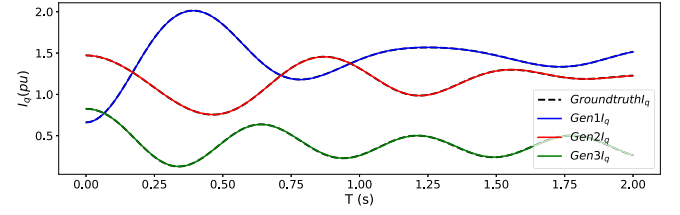
(a) I_d current trajectory(b) I_q current trajectory

Fig. 8. Predicted trajectories and ground truth trajectories for a representative sample in the extrapolation dataset for a system with fixed initial conditions and continuous disturbance.

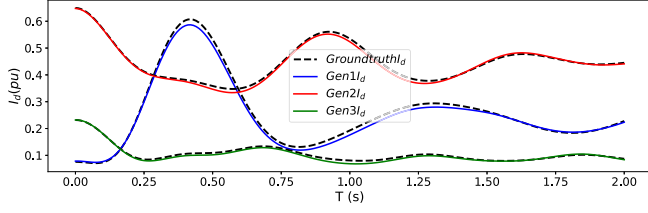
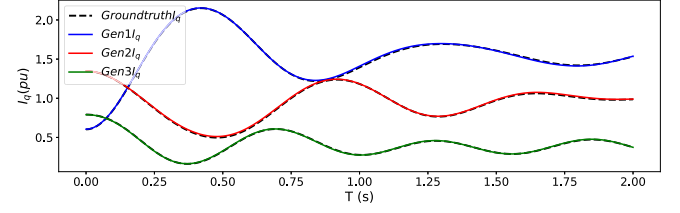
(a) I_d current trajectory(b) I_q current trajectory

Fig. 9. Predicted trajectories and ground truth trajectories for a representative sample in the extrapolation dataset for a system with variable initial conditions and continuous disturbance.

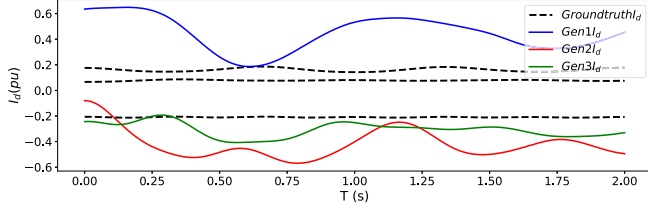
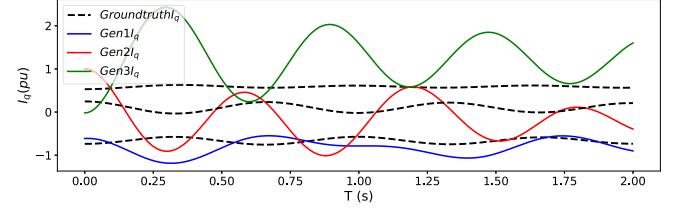
(a) I_d current trajectory(b) I_q current trajectory

Fig. 10. Predicted trajectories and ground truth trajectories for a representative sample in the extrapolation dataset for a system with discrete disturbance.

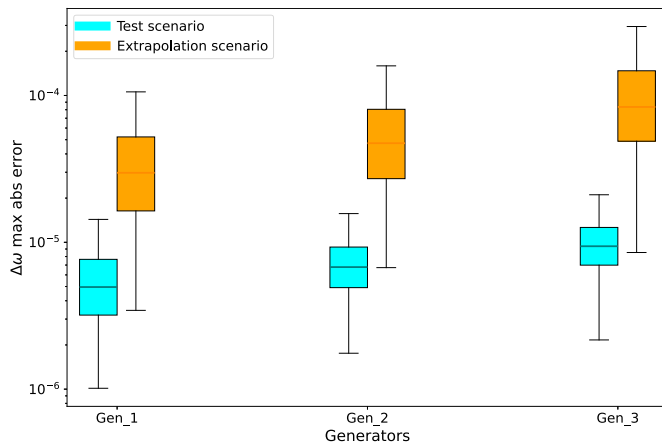


Fig. 11. Comparison of the maximum absolute error in frequency deviations on the test dataset and the extrapolation dataset for the system with continuous disturbance.

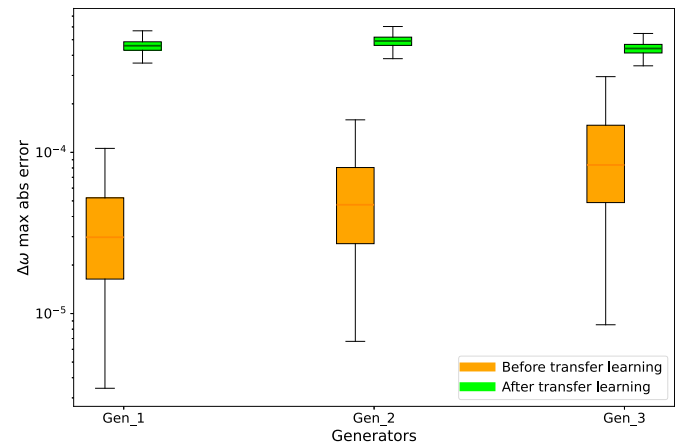


Fig. 12. Comparison of the maximum absolute error in frequency deviations on the extrapolation dataset before and after transfer learning for the system with continuous disturbance.

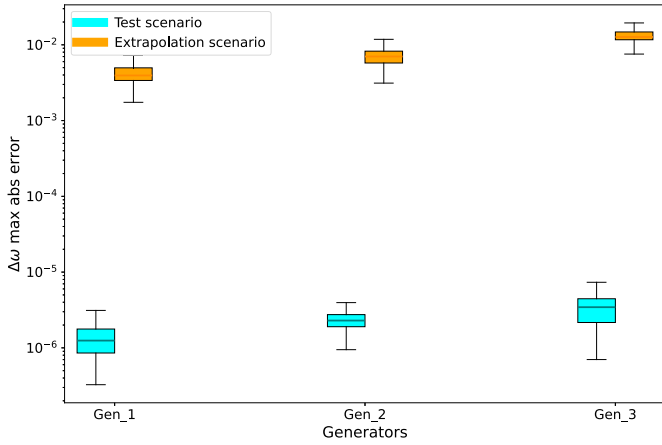


Fig. 13. Comparison of the maximum absolute error in frequency deviations on the test dataset and the extrapolation dataset for the system with discrete disturbance.

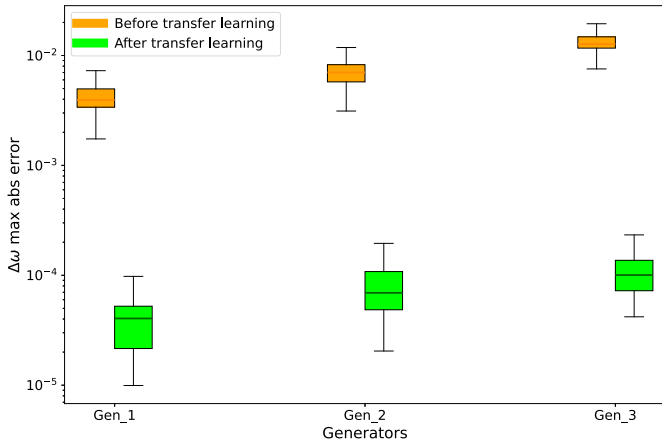


Fig. 14. Comparison of the maximum absolute error in frequency deviations on the extrapolation dataset before and after transfer learning for the system with discrete disturbance.

shows the distribution of maximum absolute errors for frequency deviation on the test and extrapolation datasets. The model error on the extrapolation data is greater than the error on the test data. Next, we perform transfer learning and show the model's performance on the extrapolation dataset before and after transfer learning in Fig. 12. While transfer learning helps reduce the range of possible errors as measured by the maximum absolute error, the performance of the fine-tuned model became slightly worse than the original model on the extrapolation data. While this may be counterintuitive, we hypothesize that this behavior may be explained by the 'double descent' phenomenon of ML models where model performance may first get worse before it gets better, even with the addition of more data [31]. Crucially, the double descent hypothesis depends on the function f learned by the network, the size of the network and training dynamics [32]. We hypothesize that fine-tuning the model with a few additional samples increases the model variance, leading to slightly worse performance on the extrapolation dataset. We refer the interested reader to [31,32] for more on the double descent hypothesis.

4.5.2. Discrete disturbance

We also perform transfer learning in the case of discrete disturbances. Fig. 13 compares the model's performance on the test and extrapolation datasets before transfer learning, with the maximum absolute error on

the extrapolation data being significantly higher than the error on the test data. Transfer learning is then used to fine-tune the model as in the previous experiment. As shown in Fig. 14, transfer learning significantly reduces extrapolation error by two orders of magnitude in this case. The model initially performs poorly in the extrapolation scenarios. We hypothesize that the initial model is in the high-bias, low-variance regime with respect to the extrapolation dataset. Thus, transfer learning increases the model variance, improving model performance in this case. We leave further theoretical investigations on the effect of transfer learning on model performance to future work.

4.6. Computational speed

The computational speed of the numerical solver is compared to the DeepONet model. The numerical solver takes an average of 0.20 s to complete 1 simulation while the ML model predicts the solution for all 1000 simulation samples in the test dataset in a single batch, taking 0.63 s. The average time to complete 1 simulation for the ML model is thus 0.0063 s. The computational advantage of the ML model stems from the parallelization of the computations. Matrix multiplication of the model weights can be trivially parallelized on many GPU cores, while numerical solvers perform iterative computations which are more difficult to parallelize. If the data generation and model training times are discounted, the ML model is more than 300x faster than the numerical solver. When the data generation and model training times are considered, the training dataset generation took 2,056 s while the model training took 2,308 s. Therefore, the ML approach has an upfront time cost of 4,365 s. This upfront cost can be amortized over subsequent model inferences provided model inference is performed a sufficiently large number of times. Also, transfer learning only took an additional 46 s to generate the new samples and fine-tune the model.

5. Discussions

This study demonstrated that ML models for TDS may give good performance in out-of-distribution scenarios when disturbances are from a continuous distribution. For example, a model trained to predict the system's response to changes in power injections or load variations within a certain distribution may produce accurate trajectories for power injections or load variations outside the training distribution. We hypothesize that this is because of the similarity between the system's responses to disturbances within and outside the training distribution. Our experiments are limited to the case where the system remains stable and further studies are needed to verify the ability of ML models to predict unstable trajectories.

We observe that within-distribution predictions for the case with discrete disturbances give the most accurate results with the worst-case MAE for all states. We infer that this is simply due to the complexity of the ground truth trajectories to be learned in this case. As [13] noted, the accuracy of ML for TDS depends on the complexity of the system responses to be approximated. As an example, it is much easier to learn to predict a flat signal compared to a complex signal with non-periodic oscillations. Conversely, we observe that out-of-distribution predictions for the case with discrete disturbances give the most inaccurate results, with the worst-case MAE significantly higher than in both cases with continuous disturbances. [33] noted that MLP-based neural networks such as DeepONet extrapolate linearly outside of the support of the training distribution. Our results suggest that such extrapolation behavior is insufficient to capture the different system responses that may arise in the case with discrete disturbances. As such, generalizing to out-of-distribution instances in the case of discrete disturbances remains challenging. As indicated in Table 2, the maximum model extrapolation error for a discrete disturbance can be orders of magnitude greater than the extrapolation error for continuous disturbances. This may lead to wrong security conclusions. The limitations of ML models in handling discrete changes have been observed in ML models for security

classification. We observe a similar limitation of a state-of-the-art ML model (DeepONet) for TDS.

Furthermore, the ML model offers up to 300x improvement at the inference stage if data collection and model training times are discounted. However, we note that if data collection and model training times are considered, the ML model would need to be evaluated a large number of times to justify the upfront cost. For example, with a combined data generation and model training time cost of 4,365 s, the model will require 6.9×10^6 evaluations to pay off the upfront cost. The restricted setting of fixed initial conditions typically considered in the literature is unlikely to satisfy the requirement for numerous evaluations. Additionally, we have considered the more realistic setting with variable initial conditions in this study. Even though the dimensionality of the model input increases by x6, the model can achieve high accuracy with the same number of training samples as in the case with fixed initial conditions. The model also has a comparable extrapolation performance to that of the case with fixed initial conditions. This suggests the model has high sample efficiency. We also note that a model trained with variable initial conditions is more likely to be used for a large number of evaluations, thereby justifying its upfront costs.

Furthermore, we demonstrated how transfer learning may be used to reduce model errors where there is a shift in the input distribution. While transfer learning significantly reduces the computational expense of training a new model from scratch, the fine-tuned model suffers a slight decrease in performance in the case of continuous disturbances. Moreover, we assert that transfer learning may be operationally infeasible in practice because it requires foreknowledge of the 'new' data distribution for sampling. For example, if system operators know the full range of potential disturbances, they can directly sample training data from this range, eliminating the need for transfer learning. In the case of discrete disturbances, we found that transfer learning significantly reduces extrapolation errors. However, it remains challenging to know which 'new' network topology variations to sample in large power systems, as all variations cannot be sampled. Hence, transfer learning in the case of discrete disturbances may be infeasible in practice for a large system. System operators typically use heuristics to select discrete disturbances to simulate. Some studies such as [4] have proposed that ML models may allow operators to comprehensively assess disturbance scenarios due to faster simulations. As we have noted in this study, ML models may only produce the right conclusions for topologies that are part of the training distribution. Hence, simulation speed is not the only bottleneck in the way of more comprehensive TDS with machine learning, flexibility of ML models to different topologies and extrapolation capacity is also an important consideration. The use of better scenario selection heuristics may also reduce the range of scenarios ML models need to assess, thereby striking a better balance between fast scenario evaluations and operational risks.

While ML models for TDS can accelerate simulation of power system dynamics, the lack of flexibility limits the scope of their use. Our approach reduces the need for real-time sampling of inputs and we use only a single point, i.e. the initial condition, to predict the power system response. It can also deliver near real-time prediction for scenarios that are part of the training data distribution, however, more work is needed to improve the generalizability of ML models for power system time domain simulation, especially for discrete events where data generation is hard. We have shown that while updating the model via techniques such as transfer learning can be quickly done as evidenced by our case study on transfer learning, such approaches are inherently limited in power system security applications where data for such updates is not available during real operations. Training a single model that reliably generalizes to unseen discrete and continuous scenarios remains a fundamental challenge. One approach is to have many small models where each model only handles certain scenarios, and the models can be made to interface with each other. Graph neural networks, which have a similar 'distributed computation' approach, are worth further exploration for TDS.

6. Conclusion

In this study, we have explored the extrapolation capacity of ML models for TDS. ML models fail in extrapolation settings for discrete disturbances but may give good performance for continuous disturbances. Transfer learning can improve the accuracy of the ML model at predicting discrete disturbances in extrapolation settings, but it remains impractical to use transfer learning in real power systems since it assumes the availability of inaccessible knowledge. We propose that subsequent studies on the use of ML for TDS should consider extrapolation, especially to discrete disturbances, as part of the evaluation criteria for the models. In this regard, we hold the view that the investigation of ML models which show stability to topological variations such as graph neural networks may be a promising future direction. Approaches which improve the scalability of ML models for TDS as well as case studies on larger networks can be investigated. Further, it should be investigated whether additional power systems domain knowledge can be exploited during training. For example, one can consider power system stability indicators such as voltage stability margin or frequency deviation. The impact of training data distributions and sampling strategies on extrapolation performance should also be analysed.

CRedit authorship contribution statement

Olayiwola Arowolo: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Conceptualization. **Jochen Stiasny:** Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Jochen Cremer:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to my data/code at the attach file step.

References

- [1] P. Panciatici, G. Bareux, L. Wehenkel, Operating in the fog: security management under uncertainty, *IEEE Power Energy Mag.* 10 (5) (Sep. 2012). [Online]. Available: 40–49, <http://ieeexplore.ieee.org/document/6269204/>.
- [2] R. Schainker, P. Miller, W. Dubelday, P. Hirsch, G. Zhang, Real-time dynamic security assessment: fast simulation and modeling applied to emergency outage security of the electric grid, *IEEE Power Energy Mag.* 4 (2) (Mar. 2006). [Online]. Available: 51–58, <http://ieeexplore.ieee.org/document/1597996/>.
- [3] F. Milano, *Power System Modelling and Scripting*, Springer Berlin Heidelberg, 2010, vol. 0. [Online]. Available: Power Systems: <http://link.springer.com/10.1007/978-3-642-13669-6>.
- [4] W. Cui, W. Yang, B. Zhang, A frequency domain approach to predict Power System transients, *IEEE Trans. Power Syst.* 39 (1) (Jan. 2024). [Online]. Available: 465–477, <https://ieeexplore.ieee.org/document/10078020/>.
- [5] A.-A.B. Bugaje, J.L. Cremer, G. Strbac, Split-based sequential sampling for real-time security assessment, *Int. J. Electr. Power Energy Syst.* 146, Mar. 2023. [Online]. Available: 108790, <https://linkinghub.elsevier.com/retrieve/pii/S0142061522007864>.
- [6] F. Bellizio, J.L. Cremer, G. Strbac, Machine-learned security assessment for changing system topologies, *Int. J. Electr. Power Energy Syst.* 134, Jan. 2022. [Online]. Available: 107380, <https://linkinghub.elsevier.com/retrieve/pii/S0142061521006190>.
- [7] T. Zhang, M. Sun, J.L. Cremer, N. Zhang, G. Strbac, C. Kang, A confidence-aware machine learning framework for dynamic security assessment, *IEEE Trans. Power Syst.* 36 (5) (Sep. 2021). [Online]. Available: 3907–3920, <https://ieeexplore.ieee.org/document/9354032/>.
- [8] J.L. Cremer, I. Konstantelos, S.H. Tindemans, G. Strbac, Data-driven Power System operation: exploring the balance between cost and risk, *IEEE Trans. Power Syst.* 34 (1) (Jan. 2019). [Online]. Available: 791–801, <https://ieeexplore.ieee.org/document/8447254/>.
- [9] H. Chiang, 1st ed), *Direct Methods for Stability Analysis of Electric Power Systems: Theoretical Foundation, BCU Methodologies, and Applications*, Wiley,

2010. [Online]. Available: Nov.: <https://onlinelibrary.wiley.com/doi/book/10.1002/9780470872130>.
- [10] L. Duchesne, E. Karangelos, L. Wehenkel, Recent developments in machine learning for Energy Systems reliability management, *Proc. IEEE* 108 (9) (Sep. 2020). [Online]. Available: 1656–1676, <https://ieeexplore.ieee.org/document/9091534/>.
- [11] J. Li, M. Yue, Y. Zhao, G. Lin, Machine-learning-based Online transient analysis via iterative computation of generator dynamics, in: 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), IEEE, Tempe, AZ, USA, Nov. 2020, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/9302975/>.
- [12] G.S. Misyris, A. Venzke, S. Chatzivasileiadis, Physics-informed neural networks for Power Systems, in: 2020 IEEE Power & Energy Society General Meeting (PESGM), IEEE, Montreal, QC, Canada, Aug. 2020, pp. 1–5. [Online]. Available: <https://ieeexplore.ieee.org/document/9282004/>.
- [13] J. Stiasny, S. Chatzivasileiadis, Physics-informed neural networks for time-domain simulations: accuracy, computational cost, and flexibility, *Electr. Power Syst. Res.* 224, Nov. 2023. [Online]. Available: 109748, <https://linkinghub.elsevier.com/retrieve/pii/S0378779623006375>.
- [14] H. Xie, F. Bellizio, J.L. Cremer, G. Strbac, Regularised learning with selected physics for Power System dynamics, in: 2023 IEEE Belgrade PowerTech, IEEE, Belgrade, Serbia, Jun. 2023, pp. 1–7. [Online]. Available: <https://ieeexplore.ieee.org/document/10202688/>.
- [15] I.V. Nadal, J. Stiasny, S. Chatzivasileiadis, Integrating physics-informed neural networks into Power System dynamic simulations, *arxiv:2404.13325* [cs, eess]. [Online]. Available: Apr. 2024) <http://arxiv.org/abs/2404.13325>.
- [16] S. Wang, P. Perdikaris, Long-time integration of parametric evolution equations with physics-informed DeepONets, *J. Comput. Phys.* 475, Feb. 2023. [Online]. Available: 111855, <https://linkinghub.elsevier.com/retrieve/pii/S0021999122009184>.
- [17] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM J. Sci. Comput.* 43 (5) (Jan. 2021). [Online]. Available: A3055-A3081, <https://epubs.siam.org/doi/10.1137/20M1318043>.
- [18] E. Barnard, L.F.A. Wessels, Extrapolation and interpolation in neural network classifiers, *IEEE Control Syst.* 12 (5) (Oct. 1992). [Online]. Available: 50–53, <https://ieeexplore.ieee.org/document/158898/>.
- [19] P.J. Haley, D. Soloway, Extrapolation limitations of multilayer feedforward neural networks, in: [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, vol. 4, IEEE, Baltimore, MD, USA, 1992, pp. 25–30. [Online]. Available: <https://ieeexplore.ieee.org/document/227294/>.
- [20] C. Ren, Y. Xu, Transfer learning-based Power System Online dynamic security assessment: using one Model to assess many unlearned faults, *IEEE Trans. Power Syst.* 35 (1) (Jan. 2020). [Online]. Available: 821–824, <https://ieeexplore.ieee.org/document/8871201/>.
- [21] R. Zhang, W. Yao, Z. Shi, X. Ai, Y. Tang, J. Wen, Towards multi-scenario Power System Stability analysis: an unsupervised transfer learning method combining DGAT and Data augmentation, *IEEE Trans. Power Syst.* 38 (6) (Nov. 2023). [Online]. Available: 5367–5380, <https://ieeexplore.ieee.org/document/9942331/>.
- [22] C. Moya, S. Zhang, G. Lin, M. Yue, DeepONet-grid-UQ: a trustworthy deep operator framework for predicting the power grid's post-fault trajectories, *Neurocomputing* 535, May 2023. [Online]. Available: 166–182, <https://linkinghub.elsevier.com/retrieve/pii/S0925231223002503>.
- [23] K. Ye, J. Zhao, X. Liu, C. Moya, G. Lin, DeepONet based uncertainty quantification for Power System dynamics with stochastic loads, in: 2023 IEEE Power & Energy Society General Meeting (PESGM), IEEE, Orlando, FL, USA, Jul. 2023, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/10252937/>.
- [24] K.-I. Funahashi, On the approximate realization of continuous mappings by neural networks, *Neural Netw.* 2 (3) (Jan. 1989). [Online]. Available: 183–192, <https://linkinghub.elsevier.com/retrieve/pii/0893608089900038>.
- [25] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nat. Mach. Intell.* 3 (3) (2021) 218–229.
- [26] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, *IEEE Trans. Neural Netw.* 6 (4) (Jul. 1995). [Online]. Available: 911–917, <https://ieeexplore.ieee.org/document/392253/>.
- [27] P.M. Anderson, A.-A.-A. Fouad, (2nd ed), *Power System Control and Stability*, IEEE Press [u.a.], Piscataway, NJ, 2003., ser. IEEE Press power engineering series.
- [28] C. Andersson, C. Führer, J. Åkesson, Assimulo: a unified framework for ODE solvers, *Math. Comput. Simulat.* 116, Oct. 2015. [Online]. Available: 26–43, <https://linkinghub.elsevier.com/retrieve/pii/S0378475415000701>.
- [29] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, *arxiv:1412.6980* [cs]. [Online]. Available: Jan. 2017) <http://arxiv.org/abs/1412.6980>.
- [30] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, R. Hadsell, Overcoming catastrophic forgetting in neural networks, *Proc. Natl. Acad. Sci. USA* 114 (13) (Mar. 2017). [Online]. Available: 3521–3526, <https://pnas.org/doi/full/10.1073/pnas.1611835114>.
- [31] M. Belkin, D. Hsu, S. Ma, S. Mandal, Reconciling modern machine-learning practice and the classical bias–variance trade-off, *Proc. Natl. Acad. Sci. USA* 116 (32) (Aug. 2019). [Online]. Available: 15 849–15 854, <https://pnas.org/doi/full/10.1073/pnas.1903070116>.
- [32] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, I. Sutskever, Deep double descent: where bigger models and more data hurt*, *J. Stat. Mech. Theory Exp.* 2021 (12) (Dec. 2021). [Online]. Available: 124003, <https://iopscience.iop.org/article/10.1088/1742-5468/ac3a74>.
- [33] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? *arxiv:1810.00826* [cs, stat]. [Online]. Available: Feb. 2019) <http://arxiv.org/abs/1810.00826>.