Finding a Closest Saddle-Node Bifurcation in Power Systems: An Approach by Unsupervised Deep Learning

Alexander Marcial
Department of Physics and Electrical Engineering,
Linnaeus University,
Växjö, Sweden.
alexander.svenssonmarcial@lnu.se

Magnus Perninge
Department of Physics and Electrical Engineering,
Linnaeus University,
Växjö, Sweden
magnus.perninge@lnu.se

***Abstract*—We propose a neural network using an unsupervised learning strategy for direct computation of closest saddle-node bifurcations, eliminating the need for labeled training data. Our method not only estimates the worst-case load increase scenarios but also significantly reduces the computational complexity traditionally associated with this task during inference time. Simulation results validate the effectiveness and real-time applicability of our approach, demonstrating its potential as a robust tool for modern power system analysis.**

***Index Terms*—Saddle-node bifurcations, voltage stability, unsupervised learning, deep learning.**

I. Introduction

The rapid growth of intermittent energy sources, primary wind and solar power, over the last decade renders it reasonably to investigate the effect it has on voltage stability. Unpredictable fluctuations in power generation and inaccuracies in forecasting can lead to overloaded transmission lines, insufficient reactive power supply, and failures to meet scheduled demands. These situations pose substantial risks to voltage stability, potentially triggering a voltage collapse.

This paper addresses the aforementioned issue of voltage stability by employing a direct method to compute the closest saddle-node bifurcation. More specifically, we consider the load flow model with k unknown system variables (voltage magnitudes at PQ-buses and voltage angles at non-slack buses) and ℓ uncertain injections (*e.g.* active and reactive power injected at load buses and at buses connecting variable production sources to the grid), as represented by

$$f(x,\lambda)=0, \quad x\in\mathbb{R}^n,\, \lambda\in\mathbb{R}^m. \tag{1}$$

Following an approach similar to [1], we focus on the proximity of a voltage collapse, avoiding assumptions regarding the dynamics of load change and instead estimating the distance to the worst-case load increase.

Submitted to the 23rd Power Systems Computation Conference (PSCC 2024).

To circumvent the online computational burden traditionally associated with loading margin assessments, we introduce an approach using deep learning. Specifically, we define the map Λ^* that takes a nominal parameter vector λ_p to a closest point of the saddle-node bifurcation surface and train a multi-layered (deep) neural network (DNN), denoted as $\hat{\Lambda}$ to approximate Λ^*.

When training neural networks, it's crucial to differntiate between supervised and unsupervised learning. The former involves utilizing a dataset $\mathcal{D}_s = \{(\lambda^i, \Lambda^*(\lambda^i)\}$. Here, λ^i represents an input feature to the neural network while $\Lambda^*(\lambda^i)$ corresponds to the ground truth output that the neural network is expected to produce given the input feature. The process of supervised learning involves using a suitable performance metric to enable the neural network to closely mimic the ground truth. However, acquiring a comprehensive set of ground truths can be difficult and in many instances impractical due to the significant computational resources required. In contrast, unsupervised learning diverges from the mimicking approach and focuses on enabling the training process to autonomously output the true solution, as will be detailed further in this paper.

By employing an unsupervised learning strategy, our method eliminates the necessity for labeled training data, i.e. the training data augmented with the ground truth. The proposed neural network learns directly from the data, effectively harnessing the inherent structure and physical laws of power systems. We assess the effectiveness of the proposed scheme through a numerical simulation on a small power system model, comparing our results to those obtained with conventional iterative methods. The results reveal the proposed method's potential for generating stability margins appropriate for real-time use.

In conclusion, this paper offers a new solution to the issue of voltage stability in power systems. Our approach combines the robustness of saddle-node bifurcation computations with the computational efficiency afforded by deep learning, providing a promising tool for managing the dynamics of today's power grids with increasing renwable energy penetration.

A. Related literature

The examination of voltage stability often entails approximating operational boundaries through loadability surfaces in parameter space [2]–[4]. However, due to the complexity of practical power systems, achieving a parametrization of the loadability surface is practically unattainable

To remedy this, the approach of finding the closest point on the loadability surface has instead gained considerable attention in litterature, [1], [5]–[7]. Exploring the loadability surface by means of continuation methods have been proposed in [5], [8]–[10]. Other methods that have been investigated is loadability surface approximations, such as local approximations [11]–[13] and global approximations based on Galerkin methods [14]. A challenge with local approximations is the non-smoothness, of the saddle-node bifurcation surface. This non-smoothnes arises mostly due to changes in power system limits [13], [15].

In addition to conventional iterative methods, various alternative approaches have been explored in the literature. For instance, genetic algorithms [26] have been employed to find the closest saddle node bifurcation. On the subject of voltage stability assesment, researchers have also utilized neural networks [28]. Here the authors use a combination of graph neural networks with long-short term memory networks, utilizing time series data as inputs to evaluate short-term voltage stability. A method using similar methods to the one proposed in our paper can be found in [27], were the author utilized neural networks for load margin assesment through the interleaving of two neural networks. Care was taken to ensure that constraints were not violated,employing a methodology that combines supervised learning with physics-guided principles.

Whereas we have not been able to find any paper that approximates the distance to the saddle-node bifurcation surface using DNNs, the optimal power flow problem has had a recent resurge due to the implementation of DNNs [16]–[18].

II. The loadability surface

In the context of voltage stability analysis, the concept of saddle-node bifurcation loadability limit is one of the most common ways of assessing the stability. Saddle-node bifurcations are crucial as it denotes the boundary at which voltage collapse may occur. A saddle-node bifurcations is a point in parameter space in which the system moves from a stable state into an unstable state. The collection of these points forms the saddle-node bifurcation surface $\Sigma := \partial\mathcal{U}$, with $\mathcal{U} := \{\lambda \in \mathbb{R}^m : \exists x \in \mathbb{R}^n, f(x,\lambda) = 0\}$ in particular, [6] offers an expression of characterizing a saddle-node bifurcation surface as

$$0 = \Psi_{\text{SNB}}(x,\lambda,v) = \begin{cases} f(x,\lambda) \\ f_x(x,\lambda)v \\ v^T v - 1 \end{cases} \tag{2}$$

In a saddle-node bifurcation, the jacobian of the power flow equations $f(x,\lambda)$ becomes singular. This fact is reflected by the eigenvector, v, corresponding to the zero eigenvalue of the jacobian matrix.

Given a nominal parameter vector $\lambda_p \in \mathcal{U}$, it is of interest to find a point $\lambda^* \in \Sigma$ for which the distance $\|\lambda_p - \lambda^*\|$ is minimal, as it can be used to assess the risk of voltage collapse.

III. Proximity to the SNB-surface

With the present papers focus on saddle-node bifurcations we assume an n-bus power system with a predefined set of PQ-buses and voltage controlled buses. Furthermore, let $f : \mathbb{R}^k \times \mathbb{R}^\ell \to \mathbb{R}^k$ denote the power flow equations, under the assumption that the parameter space has dimension ℓ. The saddle-node bifurcation surface can then be defined as

$$\Sigma := \{\lambda \in \mathbb{R}^\ell : \exists (x,v) \in \mathbb{R}^{2k},\ \Psi_{\text{SNB}}(x,\lambda,v) = 0\}.$$

A. An iterative approach

The normal vector $n_\Sigma(\lambda_0)$ to the surface Σ at the point $\lambda_0 \in \Sigma$ can be determined from the jacobian matrix, $f_x(x_0,\lambda_0)$, where $x_0 \in \mathbb{R}^k$ satisfies $f(x_0,\lambda_0) = 0$. Specifically, if $w \in \mathbb{R}^k$ is the left eigenvector (assuming multiplicity one) corresponding to the zero eigenvalue of the jacobian matrix, then

$$n_\Sigma(\lambda_0) = \alpha w^\top f_\lambda(x_0,\lambda_0) \tag{3}$$

where $\alpha \in \mathbb{R}$ is chosen such that $\|n_\Sigma(\lambda_0)\| = 1$ and ensuring that the normal is oriented outwards from the saddle-node bifurcation surface.

In an method involving successive computations of $n_\Sigma(\lambda_0)$, the authors of [1] introduced an iterative method for finding a closest point on the saddle-node bifurcation surface. By initializing the method with a parameter $\lambda_p \in \mathcal{U}$ and a direction $d_0 \in S^{\ell-1}$ ($\ell-1$ dimensional unit sphere.) the authors of [1] use continuation methods to find the point $\lambda_1 \in \Sigma$ for which the ray $\lambda_p + rd_0$, $r \geq 0$, intersect. Effectively the continuation method results in r_0^* such that $\lambda_1 := \lambda_p + r_0^* d_0$, ending the first iteration. In the successive iteration the directional vector of load increase is chosen to $d_1 := n_\Sigma(\lambda_1)$ computed from (3). This algorithm as proposed by [1] generates a sequence of points $(\lambda_i)_{i\geq 0}$ on Σ such that $\lim_{j\to\infty}\lambda_j \in \arg\min\{\|\lambda - \lambda_p\| : \lambda \in \Sigma\}$, provided the assumption of Σ having sufficiently nice properties, such as convex or not too concave.

B. A direct method

The work of [6] proposes an alternative to the iterative approach. In this context, λ^* serves as a fixed point of the previously mentioned iterative algorithm. Consequently, due to the fact that $n_\Sigma(\lambda^*)$ is paralell to $\lambda^* - \lambda$ imples that there are points $(x^*, w, k) \in \mathbb{R}^{2n+1}$ such that

$$\begin{cases} f(x^*,\lambda^*) = 0 \\ w^\top f_x(x^*,\lambda^*) = 0, \\ w^\top f_\lambda(x^*,\lambda^*) - k(\lambda^* - \lambda) = 0, \\ w^\top w - 1 = 0. \end{cases} \tag{4}$$

A direct method emerges by searching for solutions to (4) using standard numerical algorithms.

IV. A DEEP LEARNING METHOD

The iterative and the direct methods described in the previous section can be used to find the closest point on the SNB surface to a given parameter vector λ_p. Effectively, this gives us a single point of the map $\Lambda^* : \mathcal{U} \to \Sigma$, where $\mathcal{U} \subset \mathbb{R}^m$ is the set of feasible parameter vectors, satisfying

$$\Lambda^*(\lambda_p) \in \arg\min\{\|\lambda - \lambda_p\| : \lambda \in \Sigma\}. \tag{5}$$

In power system operation, the randomness of demand and variable production sources introduce uncertainty in the system parameters and for online supervision and planning purposes, the system operator would ideally like to have full access to the entire map Λ^*. In this section we explain how unsupervised deep learning can be used to find an accurate approximation $\hat{\Lambda}$ of Λ^* using limited computational sources at inference time.

The main ingredient in the algorithm that we propose is to rewrite the optimization problem

$$\begin{aligned} \min_{(x,\lambda,w)\in\mathbb{R}^{2n+m}} \ & \|\lambda - \lambda_p\| \\ \text{s.t. } & f(x,\lambda) = 0, \\ & f_x(x,\lambda)w = 0, \\ & w^\top w - 1 = 0 \end{aligned} \tag{6}$$

using an augmented Lagrangian formulation to get an unconstrained minimization problem and then use to corresponding objective function to train our neural network.

A. Power system model

In the context of this paper, an n-bus power system with fixed network topology is considered. The indices for the nodes of the power system are assumed to be in the set $\mathcal{N} = \{i : i = 0, 1, ..., n-1\}$. It is further assumed that the power system in consideration has m voltage controlled buses, whose bus indices are located in $\mathcal{N}_{PV} \subset \mathcal{N}$. Consequently, there are $n - m - 1$ load buses located at $\mathcal{N}_{PQ} \subset \mathcal{N}$.

The focus in present paper is on steady state analysis of power systems, it is therefore reasonable to assume constant power loads. Hence, it is assumed that there are subsets $\mathcal{D}_p \subseteq \mathcal{N}_{PQ}$ and $\mathcal{D}_q \subseteq \mathcal{N}_{PQ}$ on which there are non-zero active and reactive loads respectively. The load vector is represented as $\lambda = (\lambda^P, \lambda^Q) \in \mathbb{R}_+^{|\mathcal{D}_p|} \times \mathbb{R}^{|\mathcal{D}_q|}$.

For each $i \in \mathcal{N}_{PV}$, the generator at corresponding bus will produce the fixed active power $p_{g,i} \in \mathbb{R}_+$ and variable reactive power $q_{g,i} \in \mathbb{R}$, as determined by the solution of load flow equations. The active- and reactive power output of the generators can compactly be denoted as the vectors $p_G = (p_{g,i})_{i\in\mathcal{N}_{PV}}$ and $q_G = (q_{g,i})_{i\in\mathcal{N}_{PV}}$. For the scope of this paper we will not assign any upper- or lower bounds on the reactive power generation and thus let q_G take values in $\mathbb{R}^m$.

Given this power system model, the state can be defined as $x = (V_m, \delta)$ where, $V_m \in \mathbb{R}_+^{n-m-1}$ and $\delta \in [0, 2\pi]^{n-1}$.

With the model described as above, the power flow equations can be written as

$$f(x,\lambda) = \begin{pmatrix} f_G(x) - P_G \\ f_0(x) \\ f_L(x) + \lambda \end{pmatrix}$$

where

- f_G corresponds to the active power injection at PV nodes in $\mathcal{N}_{PV}$
- f_0 gives the active and reactive injected power at PQ nodes where there is no generation or load present. I.e. in buses $\mathcal{N}_{PQ}\backslash\mathcal{D}_p$ and $\mathcal{N}_{PQ}\backslash\mathcal{D}_q$ respectively.
- f_L outputs the active and reactive power injection at nodes where these are uncertain, that is in $\mathcal{D}_p$ and $\mathcal{D}_q$ respectively.

It should be noted that while the model presented in this section has the load exclusively on PQ-buses, the method presented in subsequent section allow for the more general case in where arbitrary buses can be equipped with passive loads. The chosen model in present paper is merely for ease of notation.

B. Augmented Lagrangian formulation

The previous section introduced (6), arising from the objective of finding a $\lambda^* \in \arg\min_\lambda\{\|\lambda - \lambda_0\| : \lambda \in \Sigma\}$ such that the conditions in (2) are satisfied.

By leveraging on the approximation capabilities of neural networks, a neural network $\hat{\Lambda}(\lambda_p, \theta)$, is in this paper proposed to approximate Λ^*. The parameter θ is a vector of trainable weights for the neural network. Therefore, the objective of the training process is to find a parameter θ^* such that the neural network output is a good approximation of Λ^*

Acquiring labeled data for the problem at hand would entail finding a multitude of mappings $\Lambda^*(\lambda_p)$ from a sample data set $\mathcal{D}$. To circumvent this, the neural network training procedure proposed in this paper is trained using unsupervised learning. This necessitates the replacement of labels, typically used in a supervised setting, with a strategy that enables the network to learn from the data itself, guided by physical laws and other constraints. Therefore, (6) is relaxed into an unconstrained optimization problem through the application of the Augmented Lagrangian Method (ALM).

The ALM iteratively solves a sequence of sub-optimization problems. The Lagrangian of (6) is supplemented with a penalty term to form the augmented Lagrangian, represented as:

$$\begin{aligned} \mathcal{L}^k(x,\lambda,w,\mu_k,\rho_k) = & \frac{1}{2}\|\lambda - \lambda_0\|^2 \\ & + \mu_k h(x,\lambda,w) + \frac{\rho_k}{2}\|h(x,\lambda,w)\|_2^2 \end{aligned} \tag{7}$$

where

$$h(x,\lambda,w) = \begin{pmatrix} f(x,\lambda) & w f_x(x,\lambda) & w^T w - 1 \end{pmatrix}^T$$

is the vector of equality constraints and

$$\mu = \begin{pmatrix} \mu_1^T & \mu_2^T & \mu_3 \end{pmatrix}$$

the vector of Lagrange multipliers.

The augmented Lagrangian penalizes violations of the equality constraints, each weighted by a factor ρ_k. Each sub-optimization problem is indexed by k and μ_k serve as a parameter to approximate the Lagrange multiplier μ^*. Hence, for each k we solve for

$$x_k^* = \arg\min_x \mathcal{L}^k(x, \lambda, w, \mu_k, \rho_k) \tag{8}$$

For each iteration k, μ_k and ρ_k are kept constant, while minimizing the augmented Lagrangian. Once a (local) minimum has been obtained, multipliers are updated according to the update rule

$$\mu_{k+1} = \mu_k + \rho_k h(x_k^*, \lambda_k^*, w_k^*)$$

and

$$\rho_{k+1} = c\rho_k, \ c \geq 1$$

Then, under mild regularity conditions, see e.g. [19], one can show that $\{x_k\} \to x^*$ and $\{\mu_k\} \to \mu^*$ giving the necessary conditions for local minima, $\nabla_x \mathcal{L}(x^*, \mu^*) = 0$ and $\nabla_\mu \mathcal{L}(x^*, \mu^*) = 0$.

C. Unsupervised learning

A supervised approach to the present problem would be to train the neural network using a dataset $\mathcal{D} = \{\lambda_p^i, \lambda^{*,i}\}_{i=1}^M$, with λ_p^i being the input feature and $\lambda^{*,i} := \Lambda^*(\lambda_p^i)$ the ground truth obtained by employing either of the techniques proposed in the previous section. The network is trained by minimizing the empiric risk,

$$\min_\theta \quad \frac{1}{M} \sum_{i=1}^M \|\hat{\Lambda}(\lambda_p^i; \theta) - \lambda^{*,i}\|_2^2. \tag{9}$$

While this is a common regression approach for neural networks, it does not ensure that the output is feasible. To remedy this, a regularization term can be added, which in this constrained setting is a function penalizing constraint violations. Similar approaches have been used within the realm of deep learning based optimal power flow, see *e.g.* [17].

A key observation above is that our optimization problem does not need to involve a direct search over parameters $\lambda \in \Sigma$. To approach the deep learning approximation of Λ^* we consider a different network $\hat{F}(\lambda_p, \theta) = (\hat{X}(\lambda_p, \theta), \hat{W}(\lambda_p, \theta))$ the output of which represents the vector of unknown voltage components $x \in \mathbb{R}^{2n-m-2}$ and eigenvector w to the power flow Jacobian corresponding to the zero eigenvalue. The principal layout of the neural network used is illustrated in Fig. 1.

We will then search for a weight θ^* such that $\|f_L(\hat{X}(\lambda_p; \theta^*)) - \lambda_p\| \approx \Lambda^*(\lambda_p)$ for each λ_p within the set of feasible parameters. Effectively, this corresponds to setting

$$\hat{\Lambda}(\lambda_p; \theta) := \|f_L(\hat{X}(\lambda_p; \theta^*)) - \lambda_p\|,$$

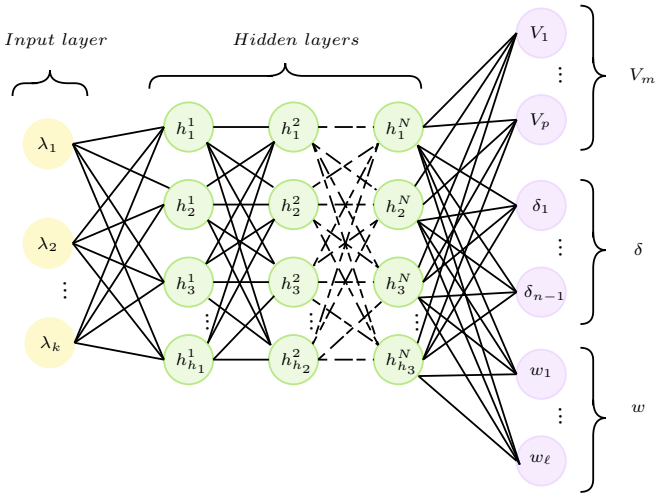

Fig. 1. Neural network architecture

a formulation designed to implicitly guarantee that load-flow feasibility is retained in all load buses.

Given a dataset $\mathcal{D} = \{\lambda_p\}_{i=1}^M$, the distance $\|\hat{\Lambda}(\lambda_p^i, \theta) - \lambda_p\|$ should be minimized over all samples in $\mathcal{D}$ while adhering to the constraints (6). We can therefore formulate the training as the following optimization

$$\arg\min_\theta \quad \sum_{i=1}^M \|f_L(\hat{X}(\lambda_p^i; \theta)) - \lambda_p\| \tag{10}$$

$$\text{s.t.} \quad \tilde{f}(\hat{X}(\lambda_p^i; \theta)) = 0, \quad \forall \lambda_p^i \in \mathcal{D} \tag{11}$$

$$f_x(\hat{X}(\lambda_p^i; \theta), \cdot)\hat{V}(\lambda_p^i; \alpha) = 0, \quad \forall \lambda_p^i \in \mathcal{D} \tag{12}$$

$$(\hat{W}(\lambda_p^i; \theta))^\top \hat{W}(\lambda_p^i; \theta) - 1 = 0, \quad \forall \lambda_p^i \in \mathcal{D} \tag{13}$$

where $\hat{W}(\lambda_p^i; \theta)$ is an auxiliary output of the neural network that, when properly trained, should output the eigenvector to the power flow Jacobian corresponding to the zero eigenvalue at the closest saddle-node bifurcation point.

To incorporate the above optimization problem in the backpropagation of the networks, we first need to relax the above problem into an unconstrained optimization problem. Lagrangian relaxation is a common method in conventional constrained optimization that has been successfully applied in the training of neural networks for constrained optimization problems. In [16] it is used to solve supervised AC-OPF using two networks for predicting the primal and dual variables respectively. In [17], [18] a penalty function is used as a regularization term in the loss function for penalizing deviations from the constraints. In this paper we will resort to use the ALM.

In the field of constrained optimization using machine learning, the augmented lagrangian has been reported successful. In particular, physics informed neural networks such as in [20] [21] [22] make use of ALM as relaxation.

D. Neural network architecture

Common with the previous cited work of using deep neural networks for approximating optimization problems, the neural

network architecture employed in this paper is a feed-forward deep neural network.

The input to the neural network is the load vector $\lambda = (\lambda^P, \lambda^Q)$, giving a total of $|\mathcal{D}_p| + |\mathcal{D}_q|$ input neurons. The input layer is followed by a sequence of hidden layers, h^i each having h_i neurons. The number of neurons per hidden layer is dependent on the power grid model under study. In general, power grids with large amount of nodes often call for more neurons in the hidden layers for better approximations. In terms of activation functions, the recticifer linear unit (ReLU), $\phi(x) = \max\{0, x\}$, is used throughout the hidden layers.

For the output layer, $2|\mathcal{N}|-|\mathcal{N}_{PV}|-1$ neurons are dedicated for representing the state vector $\hat{X} = (x_v, x_\delta)$. In terms of the voltage magnitude prediction, the set for which the amplitude take values from is the set $[V_{min}, V_{max}]^{n-m-1}$. Similarily, the voltage angles are defined to take values in the set $[-\frac{\pi}{2}, \frac{\pi}{2}]^{n-1}$. To enforce that the voltage magnitude and angles fall within these sets, we make use of the sigmoid activation function.

While it is possible to directly output the state vector in terms of votage magnitude and angle, it was found during testing that a more suitable approach in terms of numerical stability was to use an implicit representation of the state vector, x. As such, a complex representation of the voltages is used, $V_i = V_{m,i}(x_i + jy_i)$ with $|x_i + jy_i| = 1$. Thus, by letting $y_i \in [-1, 1]$, effectively renders $\arg V_i$ to be confined in the specified range.

Specifically, on the neurons representing the state vector $\hat{X}$ (V_m and δ neurons in Figure 1), the sigmoid activation function is applied,

$$\sigma(\hat{X}) = (\sigma(x_v), \sigma(x_\delta)) \in [0, 1]^{n-m-1} \times [0, 1]^{n-1}$$

The predicted voltage magnitude can then be recovered as

$$\sigma(x_v) \mapsto V_{min} + \sigma_v(x_v)(V_{max} - V_{min})$$

The voltage angles are determined implicitly by considering the complex representation of voltages, by letting the neural network output the imaginary part of a complex number with unit magnitude.

$$\sigma(x_v) \mapsto 2\sigma(x_\delta) - 1$$

An additional set of $2|\mathcal{N}| - |\mathcal{N}_{PV}| - 1$ neurons are devoted for representing the eigenvector w.

Given that the network is designed to output the eigenvector w, we can enforce the constraint (13) at the output directly, by normalizing the output.

E. Training

Based on a base case power system model with a feasible load λ_0, the set of load samples $\mathcal{D}$ is created by randomly sample points from the set $\mathcal{B}_r(\lambda_0) = \{\lambda : \|\lambda - \lambda_0\| < r\} \cap \mathbb{R}_+^{|\mathcal{D}_p|} \times \mathbb{R}^{|\mathcal{D}_q|}$. $\mathcal{D}$ is further split into two disjoint sets $\mathcal{D}_\mathcal{T}$ and $\mathcal{D}_\mathcal{E}$, where the former is used for training the neural network and the latter for evaluating the performance of the trained neural network.

Following the ALM, the training of $\hat{F}$ involves solving a sequence of optimization problems. A pre-defined amount,κ of outer-iterations is performed. Within each outer-iteration we randomly sample a mini-batch $\mathcal{B} \subset \mathcal{D}$ and update θ_k using the Adam optimizer [23]. The entire dataset is processed up to n_{epoch} times, hence the inner iteration follows a conventional training process. However, in an effort to avoid overfitting and reduce training time, early stopping with a fixed patience is used. The loss is evaluated in between epochs and if the loss is not improved for a fixed number of epochs, the inner iteration is interrupted. Further, an adaptive learning rate is used, and the learning rate is reduced if the loss is not improved. Note, the learning rate is is reverted back to the initial learning rate in each new outer iteration. Once an inner iteration is completed, the lagrange multipliers are updated in accordance with the ALM algorithm,

$$\gamma_{k+1} = \gamma_k + \rho_k g(x_k^*) \tag{14}$$
$$\mu_{k+1} = \max\{0, \rho_k h(x_k^*)\} \tag{15}$$

Data: Training dataset $\mathcal{D}_\mathcal{T}$
Result: optimal parameter θ^*
$\mu_0 \leftarrow 0$;
$\rho_0 \leftarrow \rho_{init}$;
for $k \leftarrow 0$ **to** $K - 1$ **do**
  minimize $\mathcal{L}^k(\theta, \mu_k, \rho_k)$ using Adam.;
  $\mu_{k+1} \leftarrow \max\{0, \mu_k + \rho_k h(\hat{F}(S_D; \theta_k)\}$;
  if $\|h(x)\|_\infty > \eta\|h^*\|$ **then**
    $h^* \leftarrow \|h(x)\|_\infty$ $\rho_{k+1} \leftarrow \min\{\gamma\rho_k, \rho_{max}\}$
  end
end

Algorithm 1: Training the neural network

V. Numerical results

For illustrating the proposed approach, we consider the same power system topology as the one used in the IEEE-24 bus system, with 11 voltage controlled buses and 12 PQ-buses.

The neural network is implemented using the pyTorch library and trained on a MacBook pro 2 GHz Quad-Core Intel core i5, 16 GB RAM.

A. Hyperparameters

Each hidden layer comprises of 512, 256 and 128 neurons respectively. The training dataset consists of a total of $|\mathcal{D}| = 10986$ samples, out of which 986 samples are reserved for testing the neural network. The samples were obtained by randomly selecting points in a ball centered at the base case load λ_0 with a radius, $r = 50$ (MVA).

In terms for the loss function, and the augmented Lagrangian algorithm, the outer-loop was set to 50 iterations whereas the inner iteration count was set to 50 epochs. The learning rate was set to 10^{-4}, however an exponential decay strategy were used with a decay factor, $\gamma = 0.9$. The learning rate is decayed if the loss is not improved for 5 consecutive

TABLE I
RESULTS OF TESTING THE TRAINED NEURAL NETWORK.

	Mean	Min	Max
Opt. Gap [%]	1.24	$7 \cdot 10^{-4}$	8.74
$f(\hat{x})$	$3 \cdot 10^{-4}$	$2.39 \cdot 10^{-8}$	$9 \cdot 10^{-4}$

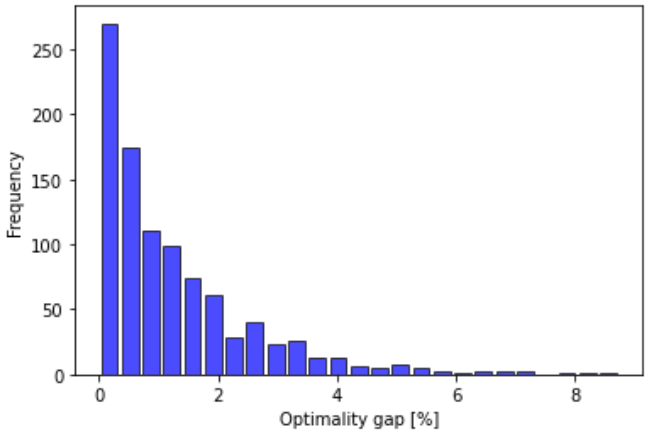

Fig. 2. Optimality gap

epochs, only to be reset between outer-iterations. Furthermore, early stopping is practiced, where the inner iteration is halted if the loss is not improved for 10 consecutive epochs. The initial penalty factor for the augmented Lagrangian algorithm was set to $\rho_0 = 100$. To avoid numerical instability, the penalty factor is capped at $\rho_{max} = 12000$. Finally, the Lagrange multipliers are initialized to zero.

B. Performance

To evaluate the accuracy of the neural networks accuracy in approximating Λ^*, the samples in $\mathcal{D}_E$ were used as input for the iterative approach presented in section III-A to serve as ground truth. The results obtained using the test set are displayed in Table 1, as well as the histograms over the optimality gap, shown in Figure 2.

The optimality gap prestented both in the table and the histogram is computed according to

$$O_{gap} = \frac{|\hat{\Lambda}(\lambda, \theta^*) - \Lambda^*(\lambda)|}{\Lambda^*(\lambda)} \tag{16}$$

VI. CONCLUSION

Based on the iterative- and direct approaches detailed in [6] and [1] we proposed the parametric approximation $\hat{\Lambda}(\lambda_p, \theta)$ of $\Lambda^*(\lambda_p)$, to find the minimal distance to the saddle-node bifurcation surface, given a stable point (x_p, λ_p). While accurate, the iterative and direct methods are computational intensive which may pose a problem in time-senstivie applications, such as real-time assesment of the margin to voltage collapse. While training neural networks is associated with needing high computational resources in terms of memory and processing power, a trained neural network, given an input, provides outputs momentarily.

Owing to the fact that the saddle-node bifurcation surface can be characterized by a set of equality constraints, the problem of finding the closest saddle-node bifurcation can be formulated as a constrained optimization problem. As such, the problem can be relaxed using the Augmented Lagrangian method to formulate a concise loss function, suitable for training the neural network. The Augmented Lagrangian formulation allows the training to learn an optimal parameter θ^* by penalizing constraint violations while minimizing the distance to the saddle-node bifurcation surface. While possible to train the neural network using supervised learning, generating a large sample set $\mathcal{D}$ and the corresponding labels $\Lambda^*(\mathcal{D})$ is highly time-consuming. However comparing the different approaches is indeed interesting for a future work.

In the numerical example, the IEEE-24 bus system was used. Using the iterative approach as ground truth in evaluating the accuracy of the neural network, 80% of the points in the test dataset had a relative error of less than 2%. While this work focused on the common fully connected feedforward neural networks, there may be other architectures worth exploring in future work. In particular, practical power systems, comprising thousands of nodes may be difficult for the neural network architecture used in this paper. In terms of potential scalability, it is evident from the literature that architectures similar to the one used in this paper have been utilized to address various computational tasks within the scope of power system analysis. Specifically, solving the Optimal Power Flow (OPF) problem, characterized by its non-convexity and non-linearity has been approached in several papers using deep neural networks to approximate the results. In [24] the authors predicts the OPF solution using neural networks and showcase their findings on the PEGASE 1354 bus system. Similarily, in [25] the authors demonstrate the application of deep neural networks using a power system model comprising 30 000 buses.

A key aspect of this paper is that we demonstrate a neural networks ability to approximate the solution of the SNB problem without a large dataset containing ground truths. Specifically, we showcase a method allowing the network to autonomously learn the approximate solution through power flow equations, therby circumventing the need for ground truth all together. It is well-known that training neural networks is a computational demanding task. However, this process occurs offline, allowing a trained network produce accurate solutions to the problem on-demand. This capability is in our view a key strength to the proposed method as we envision scenarios where one need numerous solutions for various load situations aiding in prediction task.

REFERENCES

[1] F. Alvarado, I. Dobson, and Y. Hu, "Computation of closest bifurcations in power systems," *IEEE Trans. Power Syst.*, vol. 9, no. 2, pp. 918–928, 1994.

[2] T. V. Cutsem and C. Vournas, *Voltage Stability of Electric Power Systems*. Kluwer Academic Publishers, 1998.

[3] C. D. Vournas, M. Karystianos, and N. Maratos, "Exploring power system loadability surface with optimization methods," in *Bulk Power System Dynamics and Control V*, 2001.

[4] M. E. Karystianos, N. G. Maratos, and C. D. Vournas, "Maximizing power-system loadability in the presence of multiple binding complementarity constraints," *IEEE Transactions on circuits and systems*, vol. 54, pp. 1775–1787, 2007.

[5] I. A. H. Y. V. Makarov, "A Continuation Method Approach to Finding the Closest Saddle Node Bifurcation Point," in *Proc. NSF/ECC Workshop on Bulk Power System Voltage Phenomena III*, 1994.

[6] I. Dobson and L. Lu, "New methods for computing a closest saddle node bifurcation and worst case load power margin for voltage," *IEEE Trans. Power Syst.*, vol. 8, no. 3, pp. 905–913, 1993.

[7] M. J. Chen, B. Wu, and C. Chen, "Determination of shortest distance to voltage instability with particle swarm optimization algorithm," *Euro. Trans. Electr. Power*, vol. 19, no. 8, pp. 1109–1117, 2009.

[8] I. A. Hiskens and R. J. Davy, "Exploring the Power Flow Solution Space Boundary," *IEEE Transactions on Power Systems*, vol. 16, pp. 389–395, Aug 2001.

[9] Y. V. Makarov, D.J. Hill, Z.-Y. Dong , "Computation of Bifurcation Boundaries for Power Systems: A New Δ-Plane Method," *IEEE Transactions on Circuits and Systems*, vol. 47, pp. 536–544, 2000.

[10] M. Perninge and L. Söder, "Optimal distribution of primary control participation with respect to voltage stability," *Electric Power Systems Research*, vol. 80, no. 11, pp. 1357–1363, 2010.

[11] C. Hamon, M. Perninge, and L. Söder, "A stochastic optimal power flow problem with stability constraints; part I: Approximating the stability boundary," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1839–1848, 2013.

[12] M. Perninge, "Finding points of maximal loadability considering post-contingency corrective controls," *Elect. Power Syst. Res.*, vol. 116, pp. 187–200, 2014.

[13] ——, "Approximating the loadability surface in the presence of SNB-SLL corner points," *Elect. Power Syst. Res.*, vol. 96, pp. 64–74, 2013.

[14] Y. Qiu, H. Wu, Y. Song, and J. Wang, "Global approximation of static voltage stability region boundaries considering generator reactive power limits," *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 5682–5691, 2018.

[15] M. Perninge and L. Söder, "On the validity of local approximations of the power system loadability surface," *IEEE Trans. Power Syst.*, vol. 26, no. 4, pp. 2143–2153, 2011.

[16] L. Zhang and B. Zhang, "Learning to solve the ac optimal power flow via a lagrangian approach," in *2022 North American Power Symposium (NAPS)*. IEEE, 2022, pp. 1–6.

[17] X. Pan, T. Zhao, and M. Chen, "Deepopf: Deep neural network for dc optimal power flow," 2020.

[18] X. Pan, M. Chen, T. Zhao, and S. H. Low, "Deepopf: A feasibility-optimized deep neural network approach for ac optimal power flow problems," 2022.

[19] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2016, vol. 4.

[20] F. Djeumou, C. Neary, E. Goubault, S. Putot, and U. Topcu, "Neural networks with physics-informed architectures and constraints for dynamical systems modeling," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 263–277.

[21] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, "Physics-informed neural networks with hard constraints for inverse design," *SIAM Journal on Scientific Computing*, vol. 43, no. 6, pp. B1105–B1132, 2021.

[22] A. Dener, M. A. Miller, R. M. Churchill, T. Munson, and C.-S. Chang, "Training neural networks under physical constraints using a stochastic augmented lagrangian approach," *arXiv preprint arXiv:2009.07330*, 2020.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[24] A. Lofti, M. Pirnia, "Constraint-guided Deep Neural Network for solving Optimal Power Flow," in *Electric Power Systems Research.*, vol. 211, pp. 108353.

[25] W. Chen, M. Tanneau, P. Van Hentenryck, "End-to-end feasible optimization proxies for large-scale economic dispatch," *arXiv preprint arXiv:2304.11726*, 2023.

[26] A.R Phadke, M. Manoj, K.R. Niazi, "A new technique for computation of closest saddle-node bifurcation point of power system using real coded genetic algorithm," *International Journal of Electric Power & Energy Systems*, vol. 33, no. 5, pp. 1203–1210, 2011.

[27] M. E.C. Bento, "Physics-Guided Neural Network for Load Margin Assessment of Power Systems," *IEEE Transactions on Power Systems*, vol. 39, no. 1, pp. 564–575, 2024.

[28] G. Wang, Z. Zhang, Z. Bian, Z. Xu "A short-term voltage stability online prediction method based on graph convolutional networks and long short-term memory networks," *International Journal of Electrical Power & Energy Systems*, vol. 127, pp. 106647, 2021.