Electric Power Systems Research 252 (2026) 112466

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

A deep learning approach to nonconvex optimal energy management for microgrids

Nafiseh Seifi Boghrabadi [a], Zhenbo Wang [a,*], Laxman Timilsina [b], Okan Ciftci [b], Asif Ahmed Khan [b], Behnaz Papari [b], Chris S. Edrington [b]

[a] Department of Mechanical and Aerospace Engineering, University of Tennessee, Knoxville, 37996, Tennessee, USA
[b] Department of Electrical and Computer Engineering, Clemson University, Clemson, 29634, South Carolina, USA

a r t i c l e i n f o

Keywords:
Energy management
Microgrid
Deep neural networks
Alternating direction method of multipliers
Crow search algorithm

a b s t r a c t

Traditional numerical optimization methods have been employed to solve optimal energy management (EM) problems for microgrids; however, their mathematical complexity often creates a gap between theoretical analysis and stable, real-time implementation. This paper introduces a novel use of deep neural networks (DNNs) to predict optimal solutions for nonconvex EM problems in both centralized and distributed control architectures. A significant portion of this work is dedicated to offline sample generation and training of DNNs using high-quality, optimal solutions derived from existing numerical approaches. Once the DNNs are trained, a learning-based controller utilizing these well-trained DNNs is implemented online to produce the optimal power output for each generator within a fraction of a second. If the DNNs lack generalizability or fail to meet accuracy requirements under new operational conditions, an iterative sampling and retraining process will be employed to progressively improve their output convergence, accuracy, and robustness. This novel approach demonstrates significant potential in handling nonconvex and nonlinear EM formulations, where traditional methods may struggle with convergence or computational challenges. Simulation results of a notional microgrid validate the effectiveness and performance of our proposed method and showcase its advantages over existing numerical approaches in both centralized and distributed scenarios. The centralized DNN achieved a 99.98 % success rate with a generalization error below 0.5 MW, operating up to 57× faster than the CSA method and 2.8× faster than the SQP method. Similarly, the distributed DNN reached a 99.98 % success rate with a generalization error under 0.5 MW, while delivering speedups of up to 61× over CSA and 3.6× over SQP.

1. Introduction

Microgrids, including terrestrial microgrids, ship power systems, port microgrids, and community microgrids, have become significantly more electrified and informatized over recent years, driven by the integration of distributed energy resources and advanced energy management (EM) systems that optimize their operation. Research has shown that advanced control architectures can improve operational performance of microgrids, in terms of fuel consumption, maintenance costs, reliability, resiliency, and environmental impact. In contrast to terrestrial and automotive power systems, EM in the maritime domain is extremely challenging because ships usually contain multiple distributed, independently controlled power generators, energy storages, converters, propulsion systems, power electronic apparatus, and high-power ramp rate loads [1]. In addition to advanced power electronics and power distribution architectures [2], resilient EM and power control strategies also play a key role in boosting the microgrid performance [3].

Conventional research on power and energy control primarily focuses on applied mathematics and model-based optimization methods [4]. For instance, rule-based [5] and dynamic programming [6,7] approaches have been utilized to tackle power management problems for ships and terrestrial vehicles. Nevertheless, these methods often encounter difficulties in achieving real-time performance and resilience against disturbances. Model predictive control (MPC) has emerged as a prominent approach for optimal power and energy management in recent years [8–10]. However, implementing MPC-based strategies in real-time poses challenges due to unpredictable performance and high computational complexity associated with nonconvex formulations. This complexity can lead to loss of stability and degradation of control performance [11,12]. Additionally, real-time optimization usually requires oversimplification of underlying models, which can increase the risk of suboptimal solutions. Furthermore, nearly all existing research assumes known systems and given load demand profiles, which are impractical assumptions and difficult to obtain in real-world operations and missions [13].

* Corresponding author.
E-mail address: zwang124@utk.edu (Z. Wang).

https://doi.org/10.1016/j.epsr.2025.112466
Received 22 July 2025; Received in revised form 14 September 2025; Accepted 29 October 2025
Available online 7 November 2025

The EM problem can be addressed using either centralized or distributed approaches. A centralized optimal EM for grid connected DC microgrids was presented in [14], which aimed to forecast optimal power generation based on varying loads in different operational conditions . In [15], a centralized MPC method was proposed to optimize the coordination of energy storage and generators. Additionally, [16] explored the use of the alternating direction method of multipliers (ADMM) as an effective approach for distributed convex optimization, particularly in large-scale statistical or machine learning problems. Various distributed optimization techniques, including ADMM, have been suggested for solving power management and control problems [17,18]. However, these methods have primarily focused on achieving accurate solutions, often overlooking the challenges associated with real-time implementation. Consequently, existing approaches typically encounter issues such as high computational and communication expenses and slow convergence rates. This is largely because most distributed algorithms rely on gradients and require numerous iterations, with each iteration involving at least one consensus operation, necessitating extensive communication. To reduce computational load, [19] transformed the nonconvex EM problem into a relaxed convex second-order cone programming model. This transformation reduces computational costs and provides an optimal decision-making strategy for the global operation of the microgrid.

Recently, reinforcement learning (RL) methods, including distributed Q-learning and consensus-based distributed RL, have emerged for optimal power allocation. These methods treat the problem's input (typically a specific load) as the state and its corresponding power allocation solution as the action, effectively transforming the problem into learning the discrete state-action relationship [20,21]. However, these RL algorithms often lack generalization ability and real-time performance. Particularly when dealing with continuous, time-varying loads, these algorithms struggle to address the problem in a distributed manner in real time.

To address the aforementioned challenges, this study proposes a novel learning-based optimization framework to solve nonconvex EM problems of microgrids in real time. The core concept of this approach is to treat the solution from mature yet slow numerical optimization algorithms as an unknown nonlinear mapping from inputs (load demand) to outputs (power output of each generator). This mapping is approximated by deep neural networks (DNNs) within a framework, where either one DNN serves as a global centralized controller or several distributed DNNs that act as local controllers that communicate with the neighboring controllers. Unlike RL's trial-and-error policy learning, our supervised DNN learns a direct input-output mapping from optimizer-generated labels, avoiding reward design and exploration overhead. Given that the primary objective of the EM optimization problem in microgrids is to supply the necessary power to meet the total load demand, the proposed DNN approach operates based on the average load. This is achieved through the dynamic average consensus (DAC) algorithm, which enables a group of agents to track the average of their reference inputs (average load demand) in a distributed manner.

While DNNs have been applied to various aspects of EM in microgrids such as fault detection, energy consumption, energy forecasting, and state estimation of power systems [22], research on DNNs for optimal EM or optimal power allocation is rare. A similar approach was investigated recently in [23] for a convex optimal power allocation problem in islanded microgrid systems; however, our work significantly advances the state-of-the-art by extending the approach to highly nonconvex EM problems for microgrids, which may suffer from significant convergence issues and locally optimal solutions. The main contribution of this work is threefold:

1) A novel approach is proposed to EM of microgrids that integrates data-driven DNN models with formal model-based methods to provide the optimal power outputs of dispatchable generators with guaranteed convergence and real-time performance.

2) The developed DNN-based EM approach eliminates the need to solve highly complex nonconvex mathematical optimization problems online, minimizing human supervision and allowing autonomous generation of optimal actions for microgrids in real environments, despite learning being performed on a limited subset of solutions.

3) The resulting EM controller is practical to implement on local computing devices for real-time optimal EM with guaranteed convergence, as both sample generation and network training are performed offline, and only the well-trained networks and simple matrix-vector multiplications are required online.

The rest of the paper is structured as follows. Section 2 presents the problem formulation of EM optimization based on the generator dynamics of microgrid in both centralized and distributed control configurations. Section 3 outlines the numerical approaches utilized for sample generation and comparison, focusing on the application of the ADMM method for distributed optimization and the utilization of the crow search algorithm (CSA) for solving the underlying nonconvex EM problems. Section 4 introduces the load demand determination technique using the DAC method and details the development of DNNs, including their structure, training, testing, and implementation. Section 5 provides an example microgrid model and simulation results, showcasing the outcomes of CSA, a nonlinear programming algorithm, and the developed novel DNN-based framework for comparison. Finally, concluding remarks are given in Section 6.

2. Problem statement

2.1. Energy management for microgrid

When formulating the EM optimization problem, multiple objectives may be considered, including fuel consumption, power loss through transmission lines, and other factors related to generators. As more devices are incorporated into the EM calculations, the complexity of the computations increases. However, it is advisable to prioritize crucial components such as generators, energy storage systems, and significant loads.

As discussed above, an EM layer is needed to ensure that the total load demand will be met during mission operations with the optimal output power for each generator. The optimization process will be based on the minimization of the cost for power generated by each generator and power transferred from other neighboring areas to each node. In this study, a nonconvex and nonlinear cost function is considered as follows [24]:

$$C_i(P_i) = \left(\alpha_0 + \alpha_1 P_i + \alpha_2 P_i^2 + \cdots + \alpha_m P_i^m\right) P_i \tag{1}$$

where P_i is the generated power of the i^{th} generator and C_i is the cost function calculated to produce power P_i, and α_k $(k = 1, 2, 3, \ldots, m)$ are scalar coefficients assigning both positive and negative amounts.

This study considers a total of N generators in the microgrid and a controller for each of these generators that controls the generated power of each generator and the interchanged power with neighboring nodes. The cost function of the interchanged power from a neighboring area can be explained as the cost that node i needs to pay node j when buying an amount P_{ji}, which can be expressed as follows:

$$\sum_{j\in N_i} f_i(P_{ji}) = C\left(\sum_{j\in N_i} P_{ji} + \sum_{j\in N_i} \alpha_{ji} P_{ji}^2\right) \tag{2}$$

where P_{ji} is the transmitted power from node j to node i, C is the price factor, and α_{ji} is related to power loss. Due to the size of the electric network of the microgrid, this study neglects the power loss of transmission lines. The optimization problem in each controller becomes:

$$\begin{aligned} &\min_{P_i, P_{ji}} \left(C_i(P_i) + \textstyle\sum_{j\in N_i} f_i(P_{ji})\right) \\ &\text{subject to} \quad \textstyle\sum_{j\in N_i} P_{ji} + P_i = P_{L,i}, \quad P_i^{\min} \le P_i \le P_i^{\max} \\ &P_{ji}^{\min} \le P_{ji} \le P_{ji}^{\max}, \quad P_{ji}^{\min,1} \le \textstyle\sum_{j\in N_i} P_{ji} \le P_{ji}^{\max,1} \end{aligned} \tag{3}$$

where P_i, $P_i^{\min}$, $P_i^{\max}$, P_{ji}, $P_{ji}^{\min}$, $P_{ji}^{\max}$, and $P_{L,i}$ are the generator output power, generator minimum output power limit, generator maximum output power limit, the power transferred from node j to node i, minimum and maximum limit of transferred power, and the load demand in each area needed to be met by the generators, respectively.

2.2. Centralized and distributed control architectures

When it comes to system control such as EM of microgrids, various methods can be utilized. In a centralized architecture as shown in Fig. 1, all agents in the system communicate with a single main controller. In this setup, all system

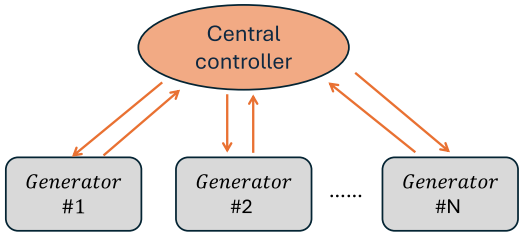

Fig. 1. Centralized control.

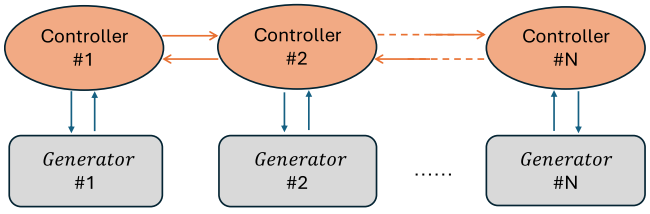

Fig. 2. Distributed control.

measurements are sent to the central controller, which solves a large-scale problem and performs the necessary calculations for the control algorithm. Then the controller dispatches the control commands to each agent to achieve the desired control action. Based on Fig. 2, in a distributed control configuration, multiple controllers manage their respective local agents. A communication network enables information exchange between the agents, allowing for coordination to ensure that overall system goals are met. Unlike centralized control, distributed agents do not need to solve the control algorithm for the entire system, reducing the computational tasks for individual controllers while providing the benefit of redundancy for the network. In this paper, the design and training of DNNs are explored to learn optimal EM solutions for both the centralized and distributed control architectures and evaluate the performance of the well-trained controllers under various load demands.

3. Numerical approaches for sample generation and comparison

The selected numerical optimization methods play a key role in our proposed DNN-based EM approach. A number of numerical approaches have been applied to EM of microgrids. In this paper, ADMM and CSA are chosen from our previous work to solve the formulated EM problems for sample generation and comparison purposes. An overview of these approaches is provided in this section.

3.1. Alternating direction method of multipliers (ADMM)

In dealing with optimization problems, it is important to note that if the objective function is convex, then the solution would be a global minimum or maximum. However, a nonconvex function will have more than one local minimum or maximum, which will hinder the optimization algorithm in finding the global minimum or maximum. Utilizing the ADMM algorithm among existing methods can be helpful in solving nonconvex problems for distributed control methodology [16,25,26]. It will allow the control system to expand to include more agents and fits for solving a nonconvex optimization problem in a distributed framework. By applying ADMM that involves the method of multipliers and dual ascent method, problem (3) is rewritten as follows [25,26]:

$$\min_{P_i,P_{ji},Z_{ji}} \left(C_i(P_i) + \sum_{j\in N_i} \left(f_i(P_{ji}) + g_i(Z_{ji}) \right) \right)$$
$$\text{subject to} \quad \sum_{j\in N_i} P_{ji} + P_i = P_{L,i}, \quad P_{ji} - Z_{ji} = 0, \quad P_{ji} \in C_i \tag{4}$$

where Z_{ji} is a variable introduced to guarantee that node i agrees with its neighbor j in determining the optimal solution for the power transmitted P_{ji} and is calculated as the average power of node i and neighbor node j, and g_i is the indicator function of the constraint set C_i that is rewritten as: $C_i = \{P_{ji} | P_{ji}^{\min} \le P_{ji} \le P_{ji}^{\max}$ and $P_{ji}^{\min,1} \le \sum_{j\in N_i} P_{ji} \le P_{ji}^{\max,1}\}$. Using the augmented Lagrangian for problem (4), the distributed algorithm is expressed as follows [16,25,26]:

$$P_{ji}^{(h+1)} = \arg\min_{P_{ji}} \left(C_i(P_i) + f_i(P_{ji}) + \frac{\rho}{2} \left\| P_{ji} + Z_{ji}^h + U_{ji}^h \right\|_2^2 \right) \tag{5}$$

$$Z_{ji}^{(h+1)} = \frac{P_{ji}^{(h+1)} - P_{ij}^{(h+1)}}{2} \tag{6}$$

$$U_{ji}^{(h+1)} = U_{ji}^h + P_{ji}^{(h+1)} - Z_{ji}^{(h+1)} \tag{7}$$

where ρ is a positive number called the penalty parameter, $U_{ji} = \frac{1}{\rho}\lambda_{ji}$ is called the dual variable, λ is the Lagrangian multiplier, and h denotes the iteration number. To obtain the optimal solution of the nonlinear Eq. (5), several methods could be used such as particle swarm optimization, genetic algorithm, ant colony optimization, and crow search algorithm (CSA). In this research the CSA method is chosen due to its simplicity, flexibility, superior convergence speed, and capability of escaping local optima for nonlinear and non- convex problems [27].

3.2. Crow search algorithm (CSA)

The CSA method was created to introduce a powerful heuristic optimizer for solving nonlinear and non- convex optimization problems based on crow's behavior [27]. In the model representing crow's behavior, each crow's position is a subset of features from the global set. Each variable of the optimization problem is a dimension of the general variable X_l. So, if there are three variables in the problem statement, then the dimension of X_l will be three. In this method the X_l is the location of a crow and there are N crows in a group. Also, each one has a location, and this location will be updated during the process by the flight length f_l. The best location will be introduced as the optimal solution based on its minimum or maximum cost function value. The update process of crow location X_l can be denoted as [27]:

$$X_l^{(i,\text{iteration}+1)} = \begin{cases} X_l^{(i,\text{iteration})} + r_i \cdot f_l \cdot \left(m^{(j,\text{iteration})} - X_l^{(i,\text{iteration})} \right), \\ \qquad \text{if } r^{(j,\text{iteration})} \ge AP^{(j,\text{iteration})} \\ \text{a random number}, \quad \text{otherwise} \end{cases} \tag{8}$$

where $X_l^{(i,\text{iteration})}$ is the location of the i-th crow in the current iteration, f_l is the flight length, $m^{(j,\text{iteration})}$ is identified as the memory location of the j-th crow, which is the best location for the j-th crow to hide food after the iteration. $\text{AP}^{(j,\text{iteration})}$ represents the awareness probability of the j-th crow, and r_i is a random number subject to uniform distribution in $[0, 1]$. When $r^{(j,\text{iteration})} > AP^{(j,\text{iteration})}$, the i-th crow tracks the j-th crow to a new hiding-food location according to Eq. (8). Otherwise, the j-th crow will distract the i-th crow to a new randomly generated position. At the end of each iteration, the matrix for storing the hiding food locations is updated as follows [27]:

$$m^{(j,\text{iteration}+1)} = \begin{cases} X_l^{(i,\text{iteration}+1)} & \text{if cost}\left(X_l^{(i,\text{iteration}+1)} \right) \\ & < \text{cost}(m^{(i,\text{iteration})}) \\ m^{(i,\text{iteration})} & \text{otherwise} \end{cases} \tag{9}$$

In each iteration, the best crow's memory (minimum or maximum cost) is saved and after all iterations completed, the final solution is selected as the optimum cost from all stored memories. Therefore, each controller starts the procedure by applying the CSA algorithm to Eq. (5) to obtain P_{ji} in each iteration. It then sends P_{ji} back to its neighbors and receives P_{ij} from the neighbors in the same iteration. In the next step, Eq. (6) will be solved to obtain Z_{ji}, which will lead to an updated U_{ji} using Eq. (7). This procedure is repeated until reaching the stopping criteria as described below [25,26]:

$$\|r_{i,h}\|_2^2 = \sum_{j=1}^{N_j} \left\| P_{ji,h} - Z_{ji,h} \right\|_2^2 \le \epsilon_i^{\text{pri}}, \quad \|S_{i,h}\|_2^2 = \rho^2 \sum_{j=1}^{N_j} \left\| Z_{ji,h+1} - Z_{ji,h} \right\|_2^2 \le \epsilon_i^{\text{dual}} \tag{10}$$

where ϵ_i^{pri} and ϵ_i^{dual} are small positive numbers in the range of $[10^{-5}, 10^{-3}]$ in a per-unit system. In this paper, the CSA is used to solve the EM problem in both the centralized and distributed (in combination with ADMM) control architectures to generate sample data for DNN training as detailed in the following section.

4. DNN-based optimal energy management

In this section, our proposed novel learning-based framework based on DNNs is introduced to solve the optimal EM problem in real time under both centralized and distributed architectures depending on the desired conditions. The development, design, and implementation of the framework and approach are detailed in this section.

4.1. A two-step DNN-based EM method

The proposed DNN-based EM algorithm is decomposed among all the generators and each sub-algorithm consists of two steps. In the first step, it deals with load demand determination that computes and updates the average load demand by synchronously monitoring and measuring the load profiles of local generators. To this end, a first-order dynamic average consensus (DAC) algorithm is applied to determine the time-varying load demand in a distributed manner through communications among dispatchable generators (DG) [23]. Supposing the reference signal for the i-th DG is denoted as $r_i(t)$, then the incremental change of $r_i(t)$ in the most recent sample can be calculated as: 1

$$\Delta r_i(t) = r_i(t) - r_i(t-h) \quad \text{for } t \geq 0 \tag{11}$$

where h is the time discretization unit, and $\Delta r_i(0) = r_i(0)$. Denote the state of the i-th DG at time instant t as $x_i(t)$. Then, the DAC protocol can be written as:

$$x_i(t+h) = x_i(t) + \sum_{\substack{j=1 \\ j\neq i}}^{N} a_{ij}\left(x_j(t) - x_i(t)\right) + \Delta r_i(t) \tag{12}$$

with a_{ij} being the interacting gain between the i-th and j-th DG. It has been proved in [28] that under the DAC protocol in Eqs. (11) and (12), all the DGs can track the average of their input references asymptotically, that is,

$$\lim_{t\to\infty} x_i(t) = x_j(t) = \frac{\sum_{i=1}^{N} r_i(t)}{N} \quad \forall i, j = 1, 2, \ldots, N \tag{13}$$

Considering $r_i(t)$ as the local load consumption at the i-th DG at time t, the discrete-time DAC algorithm Eqs. (12) and (13) are utilized to realize the global load consumption discovery in a distributed way. The flowchart in Fig. 3 shows the DAC algorithm as the first step of the proposed approach.

Once the average load demand $\frac{\sum_{i=1}^{N} P_{L,i}}{N}$ is ready, it will be fed into the DNN as the input for DNN training. During the training procedure, the training data set is utilized to optimize the weights of the DNN so that it gradually approaches the desired output. Once the offline training of the DNN finishes, the distributed learning-based optimization algorithm based on well-trained DNN will be implemented online to calculate the optimal power P_i for each DG in real time as will be detailed in the following subsections.

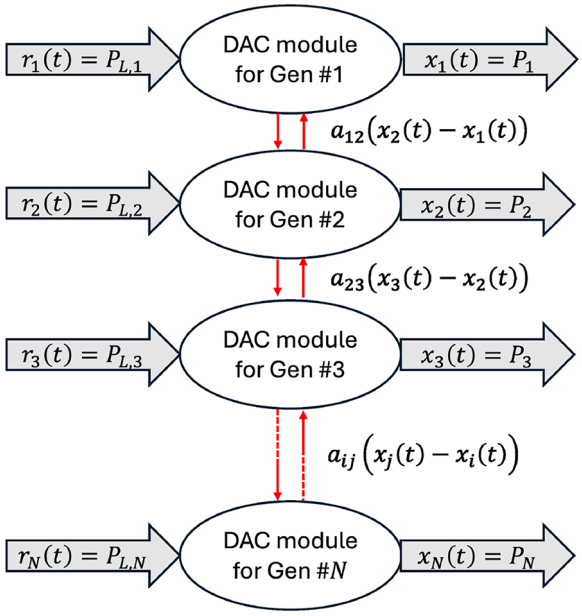

Fig. 3. DAC algorithm sketch.

4.2. Training of DNNs

A significant step of our method is to train the local DNNs offline letting them learn the nonlinear mathematical relationship between input (load demand) and the optimal output (output power of each generator). With a training data set, the DNNs can learn and estimate the output with the best approximation and accuracy.

In centralized control, $\sum_{i=1}^{N} P_{L,i}$ is determined as the input, and P_i ($i = 1, 2, 3, \ldots, N$) represents the output of the training data set obtained from optimal numerical solutions using the CSA algorithm. Regarding distributed control, the training sample consists of the optimal output power of each generator and the corresponding average load demand $\frac{\sum_{i=1}^{N} P_{L,i}}{N}$ obtained using numerical methods such as ADMM and CSA. The training dataset is constructed by taking as many data samples as possible to improve generalization purposes. To this end, this study uses a constant step sampling technique, in which approximately 20 % of the total data is selected with a fixed sampling interval. This approach ensures uniform coverage across the entire dataset, thereby preserving the statistical representativeness and diversity of the original data distribution while maintaining the quality and generalizability of the learned model.

The obtained data set is divided randomly into a training set, a validation set, and a test set with a ratio of 70 %, 15 %, and 15 %, respectively. Additionally, normalization of data is suggested in the case of a large amount of data. The proposed DNNs used in this research are fully connected feedforward networks consisting of multiple layers and neurons. The number of hidden layers (depth) of a DNN allows it to learn complex, hierarchical patterns and often improves generalization but makes training harder and slower, while increasing the number of neurons (width) can make training more stable and improve fine-grained learning but increases the risk of overfitting and computational cost. The appropriate number of hidden layers and neurons in each layer is chosen by cross-validation to prevent underfitting caused by a small number of hidden layers and overfitting seen as the result of a large number of hidden layers. During the training and validation steps, the cost function used in this study is the Mean Square Error (MSE) between the actual values y_i and the predicted values $\hat{y}_i$, defined as:

$$\text{MSE} = \frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 \tag{14}$$

while M is the total number of sample points. Also, the optimization algorithm utilized for training is Adam, and the rectifier linear unit (ReLU) is adopted as the activation function of hidden layers, while appropriate learning rate and batch size are determined through cross-validation, which will be investigated and detailed in Section 5.

4.3. Testing of DNNs

After training, the DNN is tested by feeding in load demand and comparing its output with results from numerical optimization. The error between them indicates the model's accuracy (smaller errors mean better performance). If the error exceeds a threshold, the model is retrained with adjusted layers, neurons, or activation functions. Once accuracy is sufficient, the framework's computational speed is evaluated for real-time application.

4.4. Implementation

An overview of the proposed two-step approach is depicted in Fig. 4. As shown in this figure, once the training and testing of the DNNs are completed, the well-trained DNNs are used to replace the numerical, time-consuming optimization algorithms to calculate the optimal output power for each generator based on the load profile in either centralized or distributed manner. Using this

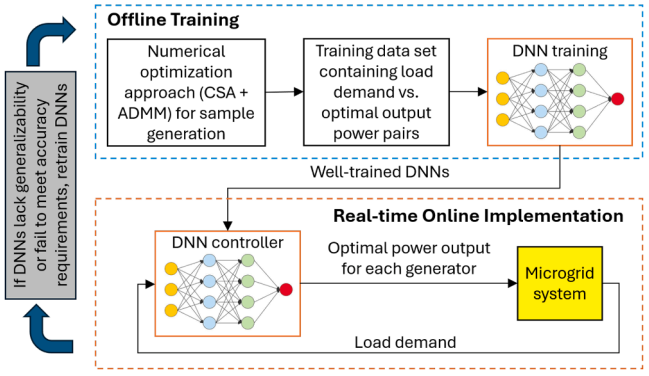

Fig. 4. A DNN-based approach to EM for microgrids.

DNN-based optimal controller is expected to reduce the processing time by a considerable amount. If the initial version of the DNN lacks generalizability, fails to converge, or does not efficiently meet load demand under new operational conditions, an iterative sample generation and learning mechanism will be triggered [29,30]. In such cases, the CSA algorithm will run again to obtain new optimal solutions according to the updated conditions. These newly generated samples will be added to the existing data set, making the model adapt to changing operational environments. Hence, the DNN will be retrained using this expanded dataset, which includes both previous and new data. Therefore, this iterative learning strategy progressively improves the model's predictive accuracy and robustness, resulting in more dependable forecasts of optimal power generation across a wider range of conditions.

4.5. Theoretical foundations

Theoretical foundations are provided in this subsection by examining the architecture and functionality of the DNNs through key components including perceptron, activation functions, loss functions, backpropagation, and their capacity and expressiveness. To this end, we consider a single perceptron shown in Fig. 5, where $x_i \in \mathbb{R}$ are the inputs, ω_i are the weights that constitute $W^{(l)}$, the weighting matrix of layer l, $b^{(l)}$ is the bias vector for layer l, σ is the activation function applied in each neuron, y is the target output, and $\hat{y}$ is the network output. The operation of a DNN in achieving a successfully converged output can be described in the following sequence: forward pass equations and mathematical formulations, loss function calculation, optimization algorithm performance, and backpropagation derivative calculations. Each of these steps is discussed in detail below.

Based on the main network characteristics in this study with Relu activation function $\sigma(z) = \max(0, z)$, the Adam optimizer algorithm, and the MSE loss function, we consider all neurons in the first layer:

$$z^{(1)} = W^{(1)}x + b^{(1)} \tag{15}$$

$$a^{(1)} = \sigma\left(z^{(1)}\right) = \max\left(0, z^{(1)}\right) \tag{16}$$

where $z^{(1)}$ is the input of the first hidden layer neurons and $a^{(1)}$ is the output of the first hidden layer neurons. Considering the forward pass, the input and outputs of neurons located in layer l will be:

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)} \tag{17}$$

$$a^{(l)} = \sigma\left(z^{(l)}\right) = \max\left(0, z^{(l)}\right) \tag{18}$$

Continuing the forward pass yields the final network outputs as follows:

$$\hat{y} = W^{(\text{output})}a^{(\text{last hidden layer})} + b^{(\text{output})} \tag{19}$$

To calculate the loss function, considering a batch point in the dataset $(x_i, y_i), i = 1, 2, \ldots, m$ where m is the desired batch number, the squared error over all outputs is:

$$\text{Error} = \frac{1}{\text{number of outputs}} \sum_{j=1}^{\text{number of outputs}} \left(\hat{y}_j - y_j\right)^2 \tag{20}$$

where $\hat{y}_j$ is the j^{th} network output and y_j is the target output corresponding to the same output neuron. The total loss is calculated for all batch sizes as follows:

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{\text{number of outputs}} \sum_{j=1}^{\text{number of outputs}} \left(\hat{y}_{ij} - y_{ij}\right)^2 \tag{21}$$

where θ represents the general parameters of the network calculated during the training process, given by $\theta = [W, b]$.

The optimization algorithm used in this case study is the Adam optimizer, which combines momentum and adaptive learning rates. This optimizer updates the network's parameters to drive the gradient of the loss function toward zero. At step time t, the gradient of the loss is:

$$g_t = \nabla_\theta L(\theta_{t-1}) \tag{22}$$

Adam keeps two exponential moving averages per parameter: the first moment m_t (which is the mean of gradients) and the second moment v_t (which is the mean of squared gradients). Thus, we have:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t \tag{23}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2 \tag{24}$$

where g_t is the gradient at step t, β_1 is the first moment decay rate that is commonly chosen as 0.9, and β_2 is the second moment decay rate that is commonly chosen as 0.999. In practice, β_1 smooths the direction of movement, while β_2 smooths the magnitude of movement. At time $t = 0, m_0 = 0$ and $v_0 = 0$. Because m_t and v_t are initialized at zero, their expectations are biased low, especially for small t. Adam corrects this by defining bias-corrected estimates as:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{25}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{26}$$

During the training process, the parameters $\theta = [W, b]$ will be updated according to:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{27}$$

where α is the learning rate, and the typical default $\epsilon = 10^{-8}$.

In the calculation of backpropagation derivatives, the loss gradient will be calculated with respect to the weight and bias parameters from the output layer in a backward direction until the first layer. Thus, considering the loss gradient in the output layer, which has a linear activation function, we have $\delta^{(L)} = \frac{\partial L}{\partial z^{(L)}} = \frac{\partial L}{\partial a^{(L)}} f'^{(L)}(z^{(L)})$. For linear functions, $f'(z) = 1$, so $\delta^{(L)} = \frac{\partial L}{\partial a^{(L)}}$. Gradient with respect to weights is $\frac{\partial L}{\partial W^{(L)}} = \delta^{(L)}\left(a^{(L-1)}\right)^T$, and gradient with respect to bias is $\frac{\partial L}{\partial b^{(L)}} = \delta^{(L)}$, where L is the output layer and the loss function, a is the output of a neuron, and z is the input to a neuron.

Considering a hidden layer, the gradients will be calculated as $\delta^{(l)} = \left(W^{(l+1)}\right)^T \delta^{(l+1)} \odot f'^{(l)}(z^{(l)})$. For the ReLU activation function where $f'(z) = 1 \text{if } z > 0 \text{and} f'(z) = 0 \text{otherwise}$, gradient with respect to weights is $\frac{\partial L}{\partial W^{(l)}} = \delta^{(l)}\left(a^{(l-1)}\right)^T$ and gradient with respect to bias is $\frac{\partial L}{\partial b^{(l)}} = \delta^{(l)}$. Therefore, the procedure for updating parameters with Adam optimizer can be briefly shown for each mini batch point based on as follows: 1) Initialize at $t = 0$: $m_0 = 0, v_0 = 0$; 2) At time t, calculate $g_t = \nabla_\theta L(\theta_{t-1})$; 3) Calculate $m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t$; 4) Calculate $v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$; 5) Calculate bias-corrected first moment $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$; 6) Calculate $\hat{v} = \frac{v_t}{1-\beta_2^t}$; and 7) Parameter update by $\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}}{\sqrt{\hat{v}}+\epsilon}$. Finally, by calculating the gradient norm and parameter variation norm, converged solution from the DNN will be approved by $\|\nabla_\theta L(\theta_t)\| \to 0$ and $\|\theta_t - \theta_{t-1}\| \to 0$.

5. Case studies and simulation results

5.1. A two-generator microgrid model

The case study considers a two-generator, notional microgrid model having two windings fed by liquefied natural gas. Because of the expectation for

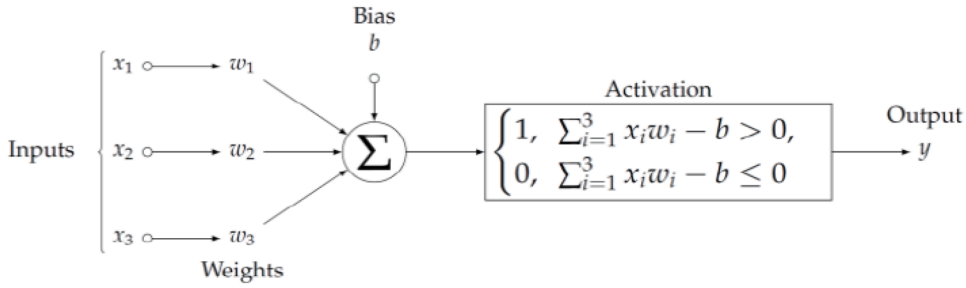

Fig. 5. Single perceptron.

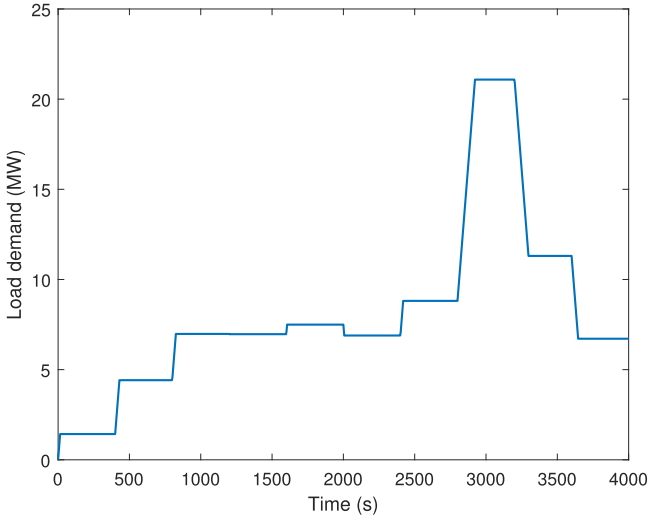

Fig. 6. Load demand profile.

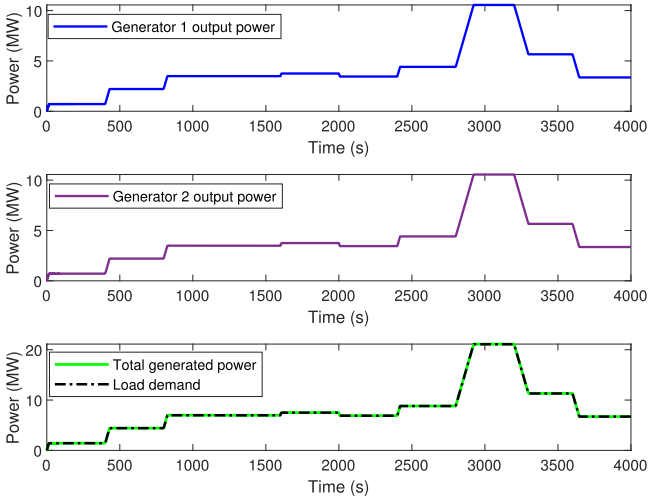

Fig. 7. Results from CSA in centralized mode.

dual winding generators, each winding is modeled as a separate voltage source to achieve independent operation. This requires that each generator has two control inputs from the external control algorithm, one for each winding. It is assumed that both generators have identical dynamic characteristics with minimum generation limit as zero and maximum generation limit as 34.48 MW, and their cost function is as follows [25]:

$$C_i(P_i) = \left(87.52 - 1.8689P_i + 0.0597P_i^2 - 0.0006845P_i^3\right)P_i$$

5.2. Centralized scenario

First, this paper investigates the performance of the proposed approach on the centralized control scenario. The optimal solution for the centralized EM control is obtained via the CSA algorithm capable of solving the nonlinear and nonconvex total cost functions. After that, the numerical, optimized data is used for training and testing the DNN consisting of one network with one input as $\sum_{i=1}^{N} P_{L,i}$ and N outputs as P_i, $i = 1, 2, \ldots, N$. Fig. 6 shows the total load profile provided to the system. The simulation runtime is 4000 s and the powers are in MW. The DNN training dataset is provided by constant sampling of the numerical results over its whole range.

CSA is used to solve the problem with the given load demand profile, and the results are provided in Fig. 7, which shows the optimal output power of each generator and the total generated power, respectively. It can be seen from these figures that CSA is efficient to converge to the optimal output power for each generator such that their total generation meets the whole load demand shown in Fig. 6. These obtained optimal solutions are used to generate sample data for DNN training and comparison.

According to the development in the previous section, offline training is performed to train and well tune the DNN and its weights. The trained DNN consists of one input, two outputs, and four hidden layers containing 15, 40, 40, and 30 neurons, respectively, as shown in Fig. 8. The batch size is chosen to be 1000, and the learning rate is 0.001 based on their least MSE values shown in Table 1 obtained via cross-validation studies. According to this table, enough large batch size is selected to yield faster training and gradient convergence, while ensuring it is not too large to avoid overfitting and poor generalization performance. Conversely, the learning rate is chosen small enough to ensure more accurate modeling and better convergence, while preventing the risk of getting stuck in local minima oscillations or divergence. After training, the centralized DNN control is provided to the two-generator model. Fig. 9 shows each generator output power an acceptable performance of the trained DNN in meeting the load demand and the power limitations of the generators.

In the next step, two new load profiles are introduced to the centralized DNN to further evaluate its generalization performance. As shown in Fig. 10, the first new load contains different values from the original training load in Fig. 6 but shares similar minimum and maximum ranges. The DNN demonstrates strong performance by providing an acceptable power estimation that could accurately meet the desired demand with the average error calculated on the entire prediction horizon of only $P_{\text{new load 1}} - P_{\text{DNN generation}} = 0.0015$ MW, which is practically negligible. Fig. 11 presents an evaluation of the generalization capability of the DNN under the second new load demand profile, which includes values that exceed the maximum of the original training data. Given that the maximum generation capacity of each generator is approximately 34.8 MW, the total load should not exceed 69.6 MW, as there are no additional generation units available. For load intervals not exceeding the training data range, the DNN maintains negligible estimation error. However, for load values exceeding the original dataset's maximum, the estimation error is analyzed in critical regions labeled operational conditions 1 through 4. According to Table 2 indicating results of Fig. 11 , in operational condition 1 where $P_L = 24.95$ MW, the total power estimation met the desired demand very closely with just the estimation error of 0.05 MW (about 0.2 % of main load) and in operational condition 2 the estimation error is about 0.187 MW (0.4 % of main load), while in operational condition 3 this error is 0.29 MW (almost 0.55 % of main load)

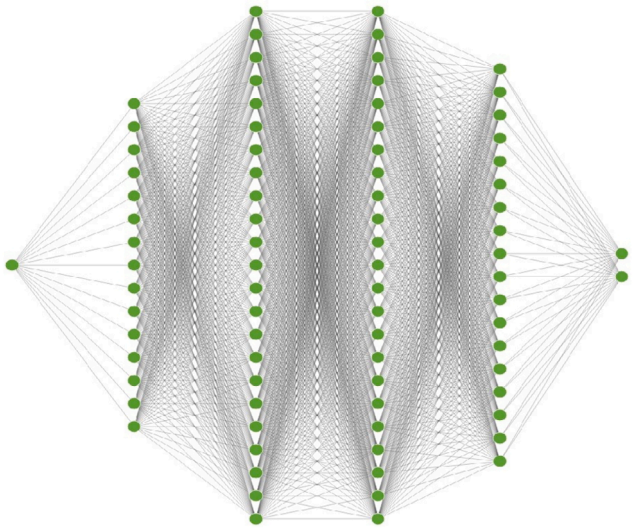

Fig. 8. DNN architecture in centralized mode.

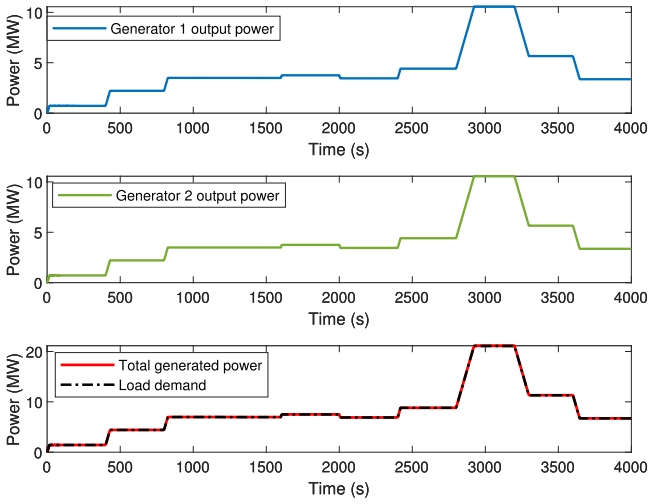

Fig. 9. Results from DNN in centralized mode.

and finally in operating condition 4 the estimation error reaches about 0.5 MW, which is just about 0.7 % of the load demand. These results indicate that the proposed DNN effectively handles unseen load profiles with higher ranges.

To further evaluate the performance of our proposed method, a comparative analysis is conducted between our DNN approach, CSA, and the sequential quadratic programming (SQP) algorithm as a representative gradient-based method through the MATLAB fmincon function. Comparison benchmarks include convergence stability, success rate, objective value achieved, and computational efficiency. To perform a statistical analysis, the three methods are evaluated in the entire original load data set using a consistent set of initial conditions, randomly selected from the interval $[P_{\min}, P_{\max}]$. A run is successful if both the P_1 and P_2 solutions converge with a solution accuracy within a threshold 10^{-4} of the true solution for each approach, and the success rate is computed as:

$$\text{Success Rate} = \frac{\text{Number of Successful Runs}}{\text{Total Number of Runs}} \tag{28}$$

After each run, the objective function value and computational time are recorded, and their averages are computed for comparison as shown in Table 3. Statistical results express that the proposed DNN approach is more successful than the gradient-based method in convergence stability with a higher success rate, lower average objective value, and less computational time.

Table 1
Batch size and learning rate selection through cross validation for the centralized case.

Batch Size	Learning Rate	Final Validation MSE	Average Validation MSE
50	0.01	2.6915e-15	0.3690
50	0.001	4.8040e-11	2.3313
50	0.0005	2.8128e-10	1.6313
500	0.01	1.0067e-11	0.3133
500	0.001	2.7575e-12	0.0681
500	0.0005	4.5421e-10	0.5731
1000	0.01	1.8946e-09	1.6681
1000	0.001	4.7177e-23	0.0353
1000	0.0005	4.0933e-09	0.5270
10000	0.01	9.7238e-08	1.8028
10000	0.001	1.0570e-11	0.1073
10000	0.0005	2.9432e-11	0.1175

Table 2
Generalization evaluation for centralized DNN with new load demand 2.

Operational Condition	1	2	3	4
New Load Demand 2 (MW)	24.95	43.74	52.71	66.78
Total Generation (MW)	25.00	43.92	53.00	67.28
Error (MW) = $P_{\text{new load 2}}$ - $P_{\text{DNN generation}}$	0.05	0.187	0.29	0.50

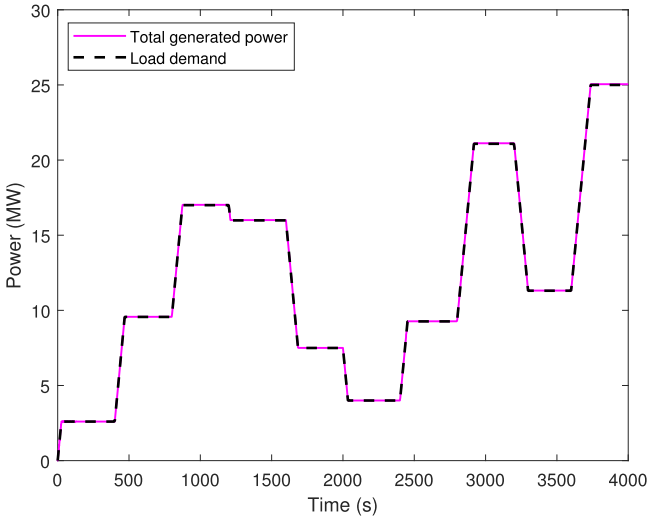

Fig. 10. New load demand profile 1 and total generated power by DNN.

5.3. Distributed scenario

In addition to the centralized case, this study also demonstrates the DNN-based approach on the distributed control scenario using the same two-generator model. It is assumed that both generators have the same load profile, and the total load demand is the same as illustrated in Fig. 6. The results of the numerical optimization approach using ADMM and CSA for distributed EM are shown in Fig. 12. Same as the centralized scenario shown in the previous subsection, the training data set is selected by constant sampling over the entire main data to capture about 20 % of it consisting all value levels.

Table 3
Statistical analysis and comparison in centralized mode.

Method	Total Number of Runs	Total Number of Successful Runs	Success Rate (%)	Average Objective Function Value	Average Computational time
CSA	800,001	800,001	100 %	646.0973	626 micro sec
fmincon	800,001	578,973	72 %	650.0254	30.36 micro sec
DNN	800,001	799,868	99.98 %	646.0973	10.94 micro sec

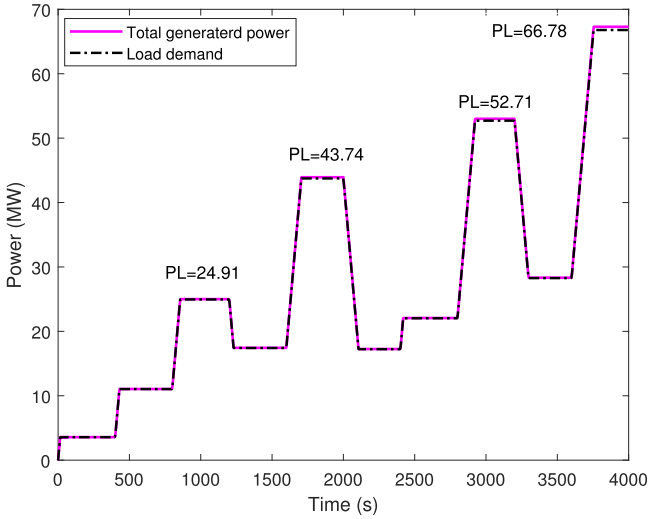

Fig. 11. New load demand profile 2 and total generated power by DNN.

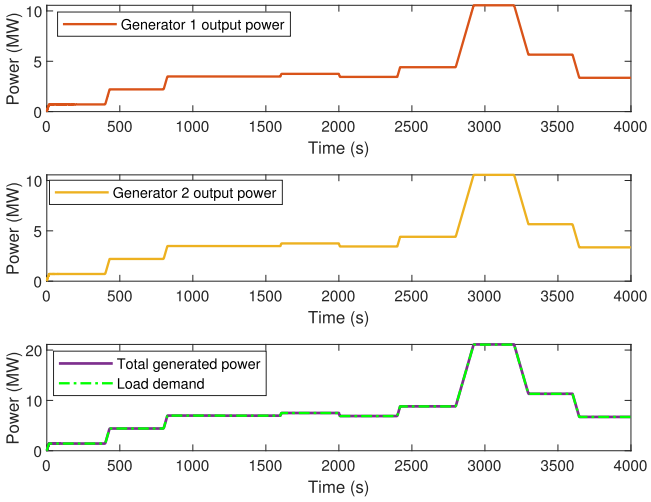

Fig. 12. Results from CSA in distributed mode.

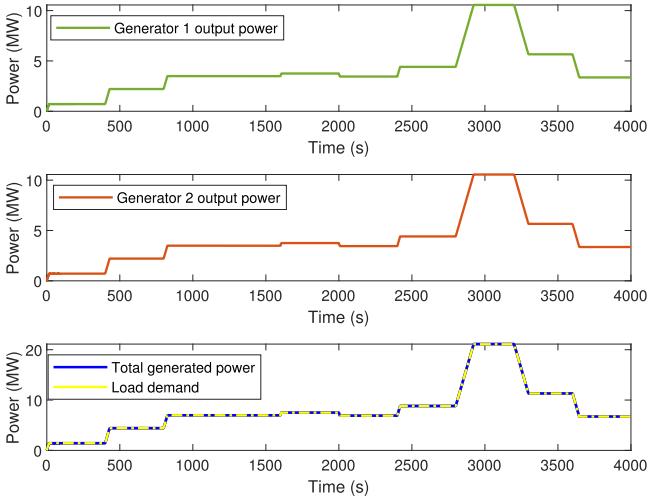

Fig. 13. Results from DNNs in distributed mode.

According to the considered two-generator model, two distributed DNNs are designed, each consisting of one input layer, one output layer, and two hidden layers with 10 and 20 neurons respectively for both distributed DNNs. To find the optimal parameters for our DNNs, cross-validation is performed as in the previous subsection. The effects of batch size and learning rate on the MSE are evaluated for generator 1. According to Table 4, it can be seen that a larger batch size causes slower convergence, while a smaller batch size may lead to non-convergent behavior. Also, a small learning rate can result in a long training process, while a large learning rate can cause the model to converge too quickly to a sub-optimal solution. As a result, for the distributed case, the batch size is chosen to be 1000 and the learning rate is 0.001 according to the least MSE. The well-trained distributed DNN-based control units are then implemented online, and the predicted results for both generators are shown in Fig. 13, which are nearly the same as those from ADMM and CSA. According to the provided figure, it is shown that the total power generation of both generators in the distributed EM approach is very close, even identical, to the total load demand, demonstrating the ability of the proposed DNN-based method to generate accurate and optimal EM solutions for microgrids.

To evaluate the generalizability performance of the distributed DNNs, the new load demand 1, shown in Fig. 14, is equally distributed to the generators, with each handling half of the total load. It is seen that both distributed DNNs perform effectively to adjust power such that their summation can fulfill the whole new load as unseen data. In the next step, same as the previous subsection, the new load profile 2, illustrated in Fig. 15, consisting of varied values higher than the maximum ranges of the original data set, is provided to the distributed DNNs as an unseen input. According to the results shown in Table 5, both distributed DNNs demonstrate successful performance on higher ranges of unseen data and in the highest possible value shown in operational condition 4, the difference between total generation and new demand is just about 0.49 MW, which is about 0.7 % of the load and can be neglected in practice.

Finally, statistical analysis is performed to compare the three methods (DNN, CSA, and SQP) for the distributed mode based on the same evaluation procedure and the benchmarks in the previous subsection. The results are shown in Table 6, demonstrating that the DNN-based approach outperforms the SQP solver with a higher success rate, a lower objective function value, and less computational time. The server computer used in this research is equipped with a 11th Gen Intel 4-Core i7 processor and 1.0 TB RAM.

5.4. Discussion and future work

Communication Challenges: Communication networks are essential to distributed microgrids. Although this study assumes ideal communication, real-world challenges, such as latency, packet loss, and synchronization, can affect how well a trained DNN generalizes and executes decisions. Latency or packet loss may delay the exchange of information among distributed energy resources, leading to outdated or incomplete data for local optimization. This may degrade decision quality and prediction accuracy, as DNNs depend on timely, accurate inputs. Such delays can shift operations away from the global optimum, even if the DNN was trained under perfect conditions. In extreme cases, excessive packet loss may cause the EM algorithm to diverge, with model parameters drifting from optimal values. The impact of communication delay

Table 4
Batch size and learning rate selection through cross validation for the distributed case.

Batch Size	Learning Rate	Final Validation MSE	Average Validation MSE
50	0.01	2.6915e-15	0.3690
50	0.001	4.8040e-11	2.3313
50	0.0005	2.8128e-10	1.6313
500	0.01	1.0067e-11	0.3133
500	0.001	2.7575e-12	0.0681
500	0.0005	4.5421e-10	0.5731
1000	0.01	1.8946e-09	1.6681
1000	0.001	4.7177e-23	0.0353
1000	0.0005	4.0933e-09	0.5270
10000	0.01	9.7238e-08	1.8028
10000	0.001	1.0570e-11	0.1073
10000	0.0005	2.9432e-11	0.1175

Table 5
Generalization evaluation for distributed DNN with new load demand 2.

Operational Condition	1	2	3	4
New Load Demand 2 (MW)	24.95	43.74	52.71	66.78
Total Generation (MW)	25.00	43.9	52.97	67.27
Error (MW) = $P_{\text{new load 2}}$ - $P_{\text{DNN generation}}$	0.05	0.16	0.26	0.49

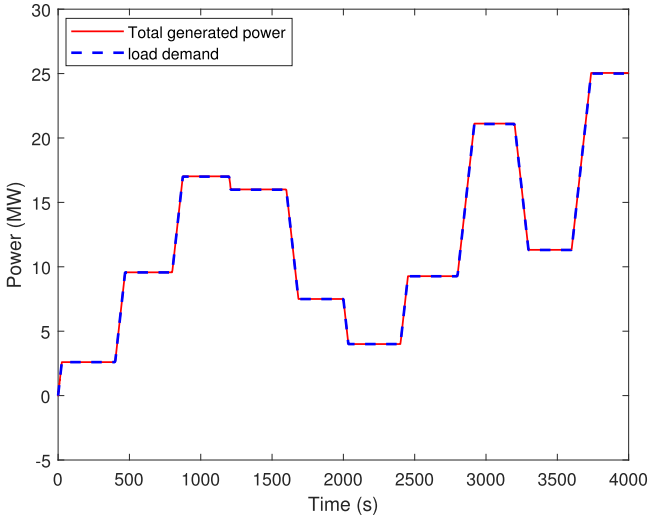

Fig. 14. New load demand profile 1 and total generated power with distributed DNNs.

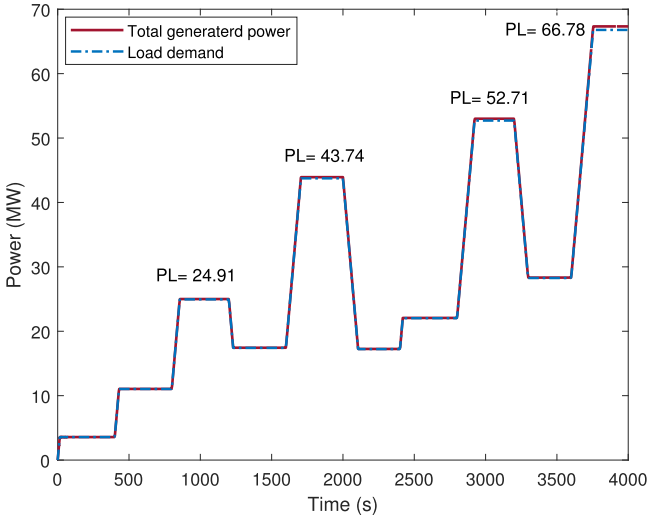

Fig. 15. New load demand profile 2 and total generated power with distributed DNNs.

can be examined through its influence on control horizons and the number of communication rounds. In our simulation, the total duration is 4000 s with a sampling time of 0.005 s, resulting in 800,000 discrete steps. Assuming a control horizon of 0.5 s, this translates to 100 discrete steps per horizon. For low latency scenarios, such as 5 milliseconds per hop, the system can accommodate approximately 100 communication rounds within each control horizon, enabling rapid convergence. However, under high latency conditions, such as 50 milliseconds per hop, only about 10 communication rounds are feasible, which significantly slows convergence. If consensus error decreases exponentially per round, modeled as $e_k = e_0 \cdot \gamma^k$ with $0 < \gamma < 1$, then in low latency cases, $\gamma^{100} \approx 0$, indicating near-perfect convergence within a single horizon. However, in high latency cases, $\gamma^{10} > 0$, leaving a non-negligible residual error that requires multiple horizons to resolve. As a result, low latency enables smooth and efficient operation with convergence achieved in nearly every horizon, while high latency leads to delayed convergence, sustained suboptimal dispatch, and gradual error accumulation across horizons. Therefore, designing latency-aware control and communication strategies is important to ensure reliable and optimal performance in microgrid environments. Additionally, poor scheduling of message exchanges can desynchronize nodes, forcing them to wait for updates and reducing overall efficiency and convergence speed. To address these issues, this study recommends: (a) robust DNN training that accounts for noisy and delayed inputs; (b) edge intelligence, enabling autonomous operation via local fallback DNNs during communication disruptions; and (c) communication-aware control, which co-designs DNN models and protocols to dynamically adjust control frequency based on network quality.

Scalability to Larger Systems: Although the current validation is limited to a two-generator system for proof-of-concept purposes, the proposed CSA-DNN framework is inherently scalable to larger power systems. From a computational perspective, CSA is a population-based metaheuristic whose computational complexity scales approximately linearly with the number of decision variables (e.g., generators, storage units) and the number of iterations as $\phi(N_p \times N_d \times N_{iter})$, where N_p is the population size, N_d the number of decision variables, and N_{iter} the number of iterations. This implies that increasing the number of generators tenfold results in a roughly linear increase in offline computation cost, which does not affect real-time performance. Once the dataset is generated, the DNN surrogate handles online decision-making, with forward inference cost given by $O(\sum_{l=1}^{L} n_{l-1} \times n_l)$, where n_l is the number of neurons in layer l. This cost grows approximately quadratically with the size of the largest hidden layer but remains independent of the number of training scenarios. As system size increases, the input and output dimensions of the DNN expand, yet modern architectures can efficiently accommodate such growth. In addition, scalability also depends on the communication and control architecture: in centralized implementations, the DNN processes higher-dimensional input vectors, but fast inference mitigates this cost; in distributed implementations, multiple smaller DNNs can be trained on local CSA-derived solutions, with global coordination achieved via lightweight consensus protocols, reducing data explosion risks and communication overhead. In sum, our CSA-DNN framework remains scalable because CSA complexity grows linearly with system size and is offline and parallelizable, DNN inference time is essentially constant with respect to system size and bounded by architecture, and distributed control strategies ensure communication scales with local connectivity rather than total system size. However, scaling to very large systems introduces challenges such as increased training data requirements and communication overhead. Each additional generator expands the decision space, requiring exponentially more samples for effective DNN generalization, which risks overfitting and longer training times. On the communication side, a fully

Table 6
Statistical analysis and comparison in distributed mode.

Method	Total Number of Runs	Total Number of Successful Runs	Success Rate (%)	Average Objective Function Value	Average Computational time
CSA	800,001	800,001	100 %	646.0973	810 micro sec
fmincon	800,001	724,001	90.5 %	648.708	47.27 micro sec
DNN	800,001	799,895	99.98 %	646.0974	13.21 micro sec

connected mesh leads to quadratic growth in links, increasing latency and synchronization issues. To address these challenges, several strategies may be pursued. For example, generators may be clustered into virtual power plants to reduce dimensionality. More advanced machine learning approaches such as federated learning may be used to share model updates trained locally instead of raw data. Consensus-based or event-triggered communication may be adopted to minimize unnecessary exchanges. Also, alternative architectures such as graph neural networks or multi-agent RL may be leveraged to efficiently capture system interactions.

Sensitivity, Transparency, and Dependency: DNNs in microgrid energy management face additional challenges, including vulnerability to unseen operating conditions, limited interpretability, and computational overhead from iterative learning. Enhancing robustness requires strategies such as data augmentation, hybrid control mechanisms, online learning, uncertainty quantification, and anomaly detection. Improving transparency involves explainability tools, interpretable model architectures, constraint-aware designs, and systematic auditing. While iterative learning enhances adaptability, it also increases computational demands and may reduce system responsiveness, highlighting the need for efficient retraining methods and reliable fallback mechanisms.

6. Conclusion

This study introduces a novel learning-based optimization algorithm for the optimal EM of microgrid systems under both centralized and distributed control scenarios. Our approach leverages DNNs to address the computational and convergence challenges associated with nonconvex and nonlinear EM problems. The proposed DNN-based method offers several significant advantages over traditional numerical optimization techniques. First, it enables real-time optimization by directly providing optimal power generation solutions based on load demand inputs, significantly reducing computational time and increasing convergence speed. This is particularly beneficial for real-time applications in microgrids where quick and reliable decision-making is crucial. Second, the distributed control framework ensures that each local controller can operate independently while coordinating with others to achieve global optimization goals. In addition, the DNN-based EM approach combined with iterative learning technique increases the model generalizability in varying operational conditions. The integration of DNNs into the EM of microgrids represents a significant advancement in the field, offering a practical and efficient solution for real-time EM in maritime, terrestrial, and other microgrids. However, although empirical validation on unseen load profiles shows consistently low deviation from optimal results, the proposed method has only been tested on a small, two-generator microgrid, leaving its performance on larger systems as an open area for future investigation. In addition, the optimality of DNN predictions is asserted under the assumption that the CSA-generated training data represent optimal solutions. CSA's results are considered reliable based on its convergence behavior and defined stopping criteria. As a global search algorithm, CSA is better at avoiding local minima than local methods like SQP, as evidenced by the comparisons in this paper. However, guaranteeing strict, particularly global, optimality remains a difficult task and presents an interesting direction for future research.

CRediT authorship contribution statement

Nafiseh Seifi Boghrabadi: Writing - original draft, Validation, Methodology, Investigation, Formal analysis; Zhenbo Wang: Writing - review & editing, Supervision, Project administration, Funding acquisition; Laxman Timilsina: Writing - review & editing, Investigation; Okan Ciftci: Writing - review & editing, Investigation; Asif Ahmed Khan: Writing - review & editing, Investigation; Behnaz Papari: Writing - review & editing, Supervision, Funding acquisition; Chris S. Edrington: Writing - review & editing, Supervision, Funding acquisition.

Data availability

No data was used for the research described in the article.

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.

Acknowledgment

The authors acknowledge the support to this work by the Office of Naval Research grant N00014-23-1-2607.

References

[1] C.S. Edrington, G. Ozkan, B. Papari, D.E. Gonsoulin, D. Perkins, T.V. Vu, H. Vahedi, Distributed energy management for ship power systems with distributed energy storage, J. Mar. Eng. Technol. 19 (1) (2020) 31–44.
[2] C.L. Peterson, Power Electronic Power Distribution Systems, 2020. ONR.
[3] J. Kuseian, Naval Power Systems Technology Development Roadmap, 2015. NAVSEA.
[4] R. Geertsma, R. Negenborn, K. Visser, J. Hopman, Design and control of hybrid power and propulsion systems for smart ships: a review of developments, Appl. Energy 194 (2017) 30–54.
[5] J. Peng, H. He, R. Xiong, Rule based energy management strategy for a series-parallel plug-in hybrid electric bus optimized by dynamic programming, Appl. Energy 185 (2017) 1633–1643.
[6] F. Millo, L. Rolando, L. Tresca, L. Pulvirenti, Development of a neural network-based energy management system for a plug-in hybrid electric vehicle, Transp. Eng. 11 (2023) 100156.
[7] H. Lee, C. Song, N. Kim, S.W. Cha, Comparative analysis of energy management strategies for HEV: dynamic programming and reinforcement learning, IEEE Access 8 (2020) 67112–67123.
[8] M.M. Mardani, M.H. Khooban, A. Masoudian, T. Dragičević, Model predictive control of DC-DC converters to mitigate the effects of pulsed power loads in naval DC microgrids, IEEE Trans. Ind. Electron. 66 (7) (2018) 5676–5685.
[9] J. Hou, Z. Song, H. Park, H. Hofmann, J. Sun, Implementation and evaluation of real-time model predictive control for load fluctuations mitigation in all-electric ship propulsion systems, Appl. Energy 230 (2018) 62–77.
[10] M. Banaei, J. Boudjadar, M.-H. Khooban, Stochastic model predictive energy management in hybrid emission-free modern maritime vessels, IEEE Trans. Ind. Inf. 17 (8) (2020) 5430–5440.
[11] J. Hu, Y. Shan, J.M. Guerrero, A. Ioinovici, K.W. Chan, J. Rodriguez, Model predictive control of microgrids-an overview, Renewable Sustainable Energy Rev. 136 (2021) 110422.
[12] U.R. Nair, R. Costa-Castelló, A model predictive control-based energy management scheme for hybrid storage system in islanded microgrids, IEEE Access 8 (2020) 97809–97822.
[13] E. Farrokhi, H. Ghoreishy, R.A. Ahangar, Optimization-based power management for battery/supercapacitor hybrid energy storage system with load estimation capability in a DC microgrid, Int. J. Electr. Power Energy Syst. 155 (2024) 109665.
[14] C.L. Bhattar, M.A. Chaudhari, Centralized energy management scheme for grid connected DC microgrid, IEEE Syst. J. 17 (3) (2023) 3741–3751.
[15] K. Baker, J. Guo, G. Hug, X. Li, Distributed MPC for efficient coordination of storage and renewable energy sources across control areas, IEEE Trans. Smart Grid 7 (2) (2016) 992–1001.
[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, 2011.
[17] G. Chen, Q. Yang, An ADMM-based distributed algorithm for economic dispatch in islanded microgrids, IEEE Trans. Ind. Inf. 14 (9) (2017) 3892–3903.
[18] P. Xie, S. Tan, N. Bazmohammadi, J.M. Guerrero, J.C. Vasquez, J.M. Alcala, J.E. M.C. no, A distributed real-time power management scheme for shipboard zonal multi-microgrid system, Appl. Energy 317 (2022) 119072.
[19] D.E. Olivares, C.A.C. nizares, M. Kazerani, A Centralized Optimal Energy Management System for Microgrids, in: 2011 IEEE Power and Energy Society General Meeting, IEEE, 2011.

[20] F. Li, J. Qin, W.X. Zheng, Distributed q-learning-based online optimization algorithm for unit commitment and dispatch in smart grid, IEEE Trans. Cybern. 50 (9) (2019) 4146–4156.
[21] S. Hasanvand, M. Rafiei, M. Gheisarnejad, M.-H. Khooban, Reliable power scheduling of an emission-free ship: multiobjective deep reinforcement learning, IEEE Trans. Transp. Electrif. 6 (2) (2020) 832–843.
[22] M. Khodayar, J. Regan, Deep neural networks in power systems: a review, Energies 16 (12) (2023).
[23] F. Guo, B. Xu, W.-A. Zhang, C. Wen, D. Zhang, L. Yu, Training deep neural network for optimal power allocation in islanded microgrid systems: a distributed learning-based approach, IEEE Trans. Neural Netw. Learn. Syst. 33 (5) (2021) 2057–2069.
[24] C. Shang, L. Fu, X. Bao, X. Xu, Y. Zhang, H. Xiao, Energy optimal dispatching of ship's integrated power system based on deep reinforcement learning, Electr. Power Syst. Res. 208 (2022) 107885.
[25] C.S. Edrington, T.V. Vu, H. Vahedi, D. Gonsoulin, D. Perkins, B. Papari, G. Ozkan, K. Schoder, H. Ravindra, M. Stanovich, M. Sloderbeck, M. Steurer, Demonstration and evaluation of large-scale distributed control for integrated power and energy mvdc systems on ships, in: Advanced Machinery Technology Symposium, 2018.
[26] T.V. Vu, C.S. Edrington, R. Hovsapian, Distributed Demand Response Considering Line Loss for Distributed Renewable Energy Systems, IEEE, 2017, pp. 1–5.
[27] A. Askarzadeh, A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm, Comput. Struct. 169 (2016) 1–12.
[28] S.S. Kia, B.V. Scoy, J. Cortes, R.A. Freeman, K.M. Lynch, S. Martinez, Tutorial on dynamic average consensus: the problem, its applications, and the algorithms, IEEE Control Syst. Mag. 39 (3) (2019) 40–72.
[29] Z. Lv, H. Huang, W. Sun, M. Jia, J.A. Benediktsson, F. Chen, Iterative Training Sample Augmentation for Enhancing Land Cover Change Detection Performance with Deep Learning Neural Network, 2023.
[30] Z. Lv, P. Zhang, L. Xie, J.A. Benediktsson, T. Lei, Iterative Sample Generation and Balance Approach for Improving Hyperspectral Remote Sensing Imagery Classification with Deep Learning Network, 2024.