Solving Bilevel Optimal Bidding Problems Using Deep Convolutional Neural Networks

D. Vlah, K. Šepetanc, *Student Member*, *IEEE* and H. Pandžić, *Senior Member*, *IEEE*

***Abstract*—Current state-of-the-art solution techniques for solving bilevel optimization problems either assume strong problem regularity criteria or are computationally intractable. In this paper we address power system problems of bilevel structure, commonly arising after the deregulation of the power industry. Such problems are predominantly solved by converting the lower-level problem into a set of equivalent constraints using the Karush-Kuhn-Tucker optimality conditions at an expense of binary variables. Furthermore, in case the lower-level problem is nonconvex, the strong duality does not hold rendering the single-level reduction techniques inapplicable. To overcome this, we propose an effective numerical scheme based on bypassing the lower level completely using an approximation function that replicates the relevant lower level effect on the upper level. The approximation function is constructed by training a deep convolutional neural network. The numerical procedure is run iteratively to enhance the accuracy.**

As a case study, the proposed method is applied to a price-maker energy storage optimal bidding problem that considers an AC power flow-based market clearing in the lower level. The results indicate that greater actual profits are achieved as compared to the less accurate DC market representation.

***Index Terms*—Bilevel optimization, deep convolutional neural network, optimal power flow.**

I. Introduction

A. Background and paper scope

Deregulation and liberalization of the power sector worldwide dislodged large monopolistic power utilities, allowing for private companies to become important players in the sector. However, each of the newly created entities have their own goal, e.g. generating companies want to maximize their profit, system operators maximize the security of supply, while market operators maximize social welfare. Because of the increased number of players with conflicting objectives, they need to consider each other's goals and objective functions when optimizing their own utility. To accommodate the interaction between own and other player's actions, the researchers commonly resort to bilevel models, where own optimization problem (the upper-level (UL) problem) is constrained by another optimization problem (the lower-level (LL) problem). This setting assumes that the lower-level behavior is known, which is the case when considering the market operator conducting its market-clearing procedure or any other regulated entity that behaves according to some widely-known rules.

As of October 5, 2021, the IEEE Xplore database [1] indicates that three flagship IEEE Power and Energy Society journals published 182 journals with word *bilevel* in the title (115 such papers in IEEE Transactions on Power Systems, 45 in IEEE Transactions on Smart Grid and 22 in IEEE Transactions on Sustainable Energy). These papers cover a wide range of bilevel problems. Some of the most common topics include protection of a power system against a terrorist attack, e.g. [2], pricing schemes, e.g. [3], maintenance scheduling, e.g. [4], expansion planning, e.g. [5], or optimal bidding in one or more energy [6] or financial markets [7].

Although bilevel models have been used extensively in the literature, they often suffer from two drawbacks. The first one is related to linearization, as commonly one or more variables from the lower-level problem appear multiply an upper-level variable in the upper-level objective function. Although in many cases this can be linearized using the strong duality theorem and some of the Karush-Kuhn-Tucker (KKT) optimality conditions [8], see e.g. [9], in some cases this is not possible. In such cases the authors commonly resort to the binary expansion method (see appendix B in [6]). However, besides being an approximation, this method can result in intolerable computational times, bringing us to the second drawback, i.e. computational (in)tractability. Issues with tractability often arise when the lower-level problem is stochastic or has many inequality constraints, resulting in a large number of binary variables. Some authors thus resort to an iterative procedure that considers the complicating dual variables in the problematic bilinear terms as parameters, and updating their values in the following iteration [10].

The aim of this paper is to present a numerical scheme based on deep convolutional neural networks paired with state-of-the-art training procedure for solving complex bilevel problems arising in the power systems community. As a representative of such problems, we solve a bilevel problem of optimal participation of an energy storage in the day-ahead energy market. We assume an AC-optimal power flow (OPF)-based market clearing algorithm in the lower-level. AC OPF is a challenging problem with numerous simplification attempts, e.g. by convexification [11]. However, to this date there is still no known exact finite convex AC OPF formulation that

The authors are with the Innovation Centre Nikola Tesla (ICENT) and the University of Zagreb Faculty of Electrical Engineering and Computing – UNIZG-FER (emails: domagoj.vlah@fer.hr; karlo.sepetanc@fer.hr; hrvoje.pandzic@fer.hr).

Employment of Karlo Šepetanc at UNIZG-FER is fully funded by the Croatian Science Foundation under programme DOK-2018-09. The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No. 863876 (project FLEXGRID) and No. 864298 (project ATTEST). Work of D. Vlah is supported by the Croatian Science Foundation (HRZZ) grant PZS-2019-02-3055 from the "Research Cooperability" program funded by the European Social Fund. The sole responsibility for the content of this document lies with the authors. It does not necessarily reflect the opinion of the Innovation and Networks Executive Agency (INEA) or the European Commission (EC). INEA or the EC are not responsible for any use that may be made of the information contained therein.

classical approaches could solve to optimality.

Optimal bidding problems consist of two interlinked optimizations. The first problem, also called the leader or the upper-level problem, represents the market participant that maximizes the agent's profit due to arbitrage, while the second problem, also called the follower or the lower-level problem, is the market clearing that maximizes the social welfare and determines the electricity prices that depend on the bids from the upper-level problem. The described bilevel optimization can not be directly solved using commercial off-the-shelf solvers, thus they are usually converted into a single-level equivalent optimization problem. However, such conversion is difficult since the exact AC OPF models, that appear in the lower level, are nonconvex and thus render many existing techniques inapplicable. The authors of this paper have already explored the single-level reduction approach in a two-part paper [12] and [13], where AC OPF is modeled using a convex quadratically constrained quadratic approximation [11], but even the most computationally efficient solution techniques start diverging for large systems, i.e. systems with over 70 buses. Here we explore a neural network (NN) metamodeling approach that is known to require significant computational time and resources, but can compute for large systems and even allows for discrete variables in the lower level (at even greater computational cost). On the other hand, the KKT-based single-level reduction techniques solve the upper and the lower level simultaneously so the solution process can diverge for large systems.

Proving the global optimality of solution is not within the scope of this paper. Generally, numerical optimization methods that can find global optimal solutions require much stronger conditions on the optimizing goal function and domain of optimization, for instance, in the case of optimizing a linear function on a convex domain. Our method is general in a sense that it imposes no mathematical conditions on the class of the lower level problem except that, using reasonable time and resources, the lower level can be evaluated a number of times to create a dataset for the NN training.

B. Literature review

As depicted in Fig. 1, the existing literature on bilevel solution techniques branches out in two main directions: classical and evolutionary approaches. Due to computational difficulty of bilevel problems, the classical approaches can only tackle well-behaved problems with strong assumptions, such as linearity or convex quadraticity and continuity of the lower-level problem, as strong duality generally does not hold for other types of problems. By far the most common classical approach is a single-level reduction based on the KKT conditions and the duality theory. It has been widely used to solve bilevel problems with linear constraints and either linear [14] or convex quadratic [15] objective functions, as well as problems with convex quadratic constraints when the interaction between the two levels is discrete [16]. The resulting formulations contain complementarity constraints which are combinatorial in their nature and thus can be modeled using binary variables making the final problems mixed-integer linear (MILP) or mixed-integer quadratic (MIQP). The existing state-of-the-art solvers generally handle well these types of optimizations using the branch-and-bound method for binary search tree and simplex for search-tree node subproblems, despite the exponential complexity in the worst case. The single-level reduction technique has also been successfully applied to a case where the lower level is a convex quadratically-constrained quadratic problem (QCQP), as in [17]. Other classical approaches are the descent method, the penalty function method, the trust-region method and the parametric programming method. The descent method determines the most favorable variable change for the objective function, as demonstrated in [18], so the model stays feasible. However, since the model is feasible only when the lower level is optimal, finding the descent direction is very difficult. The penalty function method replaces the lower-level [19] or both-level [20] constraints with penalty terms for constraint violations in the objective function. The trust region algorithms iteratively approximate the lower level around the operating point with linear problem (LP) or quadratic problem (QP) [21]. Both the penalty and the trust-region methods as the next step apply a KKT-based single-level reduction to the lower level and thus inherit the same applicability limits. A recent research thrust in parametric programming has resulted in an alternative approach to solving bilevel programs to global optimality by exploiting the notion of *critical regions*. To this point, solution approaches based on parametric programming have been proposed to handle bilevel programs with LP [22], QP [23], MILP and MIQP [24] lower levels.

As opposed to the classical ones, the evolutionary approaches are inspired by the biological evolution principle where candidate solutions are evaluated using a fitness function, e.g. objective function, to form the next generation of candidate solutions by reproducing, mutating, recombining and selecting processes. Evolutionary approaches are very effective at finding good approximate solutions of numerically very difficult problems with fewer regularity assumptions than the classical approaches. For bilevel problems, the evolution is commonly applied in a nested form where the lower level needs to be solved separately for every upper-level solution candidate, as explained and analyzed in [25]. The upper-level solution candidates are obtained using an evolution, e.g. particle swarm optimization [26] or differential evolution [27], while the lower level can be solved using classical approaches such as interior point method [28] or as well using an evolution, as in [27]. Despite applicability to nonconvex problems, where classical approaches do not hold, the nested evolutionary method does not scale well with the number of upper-level variables as they exponentially increase the

Classical approaches	**Evolutionary approaches**
• Single-level reduction • Descent method • Penalty function method • Trust region • Parametric programming	• Nested methods • Single-level reduction • Metamodeling-based methods: a) reaction set mapping b) optimal lower-level value function c) bypassing lower-level problem d) auxiliary bilevel meta-model

Fig. 1. Classification of the bilevel problem solution techniques.

number of lower-level optimizations that need to be performed. A single-level reduction technique can also be utilized in the context of evolutionary approaches where numerical evolution concept is applied to the reduced formulation. Due to KKT conditions, this technique inherits regularity assumptions of the classical approaches for the lower-level problem, but allows a more irregular upper level. Paper [29] is one of the first works where this technique was employed. Evolutionary approaches can also be applied in tandem with the meta-modeling method. Meta-model, or surrogate model, is an easy to evaluate, typically iteratively enhanceable, approximation of the original model. For bilevel modeling, the lower level can be meta-modeled using the reaction set mapping, optimal lower-level value function, by bypassing the lower-level problem completely and by using an auxiliary bilevel meta-model. The reaction set method maps the lower-level variable values as a response to the upper-level variables, as demonstrated in [30]. On the other hand, the optimal lower-level function method replaces the lower objective statement of minimization or maximization with a constraint requiring that the objective is at least as good as the optimal lower-level function [31]. Both the reaction set map and the optimal lower-level function are generally difficult to obtain even in an approximated form. Bypassing the lower-level completely is based on the principle that the lower-level variables are basically functions of the upper-level variables, which allows for the upper level reformulation not to include the lower level. Similar to the trust-region method, bilevel problems can be replaced with auxiliary meta-models. As of current, we are not aware of any works based on bypassing the lower-level problem or the auxiliary meta-models methods. A broader bilevel solution techniques research field review, for both the classical and the evolutionary approaches, can be found in [32].

The approach presented in this paper can be classified as an evolutionary meta-modeling method that bypasses the lower-level problem completely, as given by [32]. This bypassing of the lower-level problem is achieved by approximating the solution of the lower-level, which depends only on the upper-level variables, using a carefully designed NN, see [33] and [34]. As a NN is simply a function composed of elementary functions, it can be substituted directly into the upper-level objective function. This way, the original bilevel optimization problem is reduced into a single-level optimization problem, which approximates the solution of the original problem. The main difficulty of our framework is the design and training of a NN that efficiently and accurately approximates the lower-level problem.

C. Contribution

In this work we develop a general numerical solution technique for bilevel problems and apply it to the energy storage (ES) bidding problem on an AC-OPF-constrained energy market. The technique is applicable to any other upper-level subject, but we chose the ES due to modeling simplicity and clarity of presentation. The numerical and mathematical difficulty of solving the considered bilevel optimization arises from insufficient problem regularity due to nonconvexity of the exact AC OPF formulations. Current modeling practice is to avoid the difficulties by using a simpler linear DC OPF [35] network representation as in [36]. To the authors knowledge, there are very few attempts to solve bilevel problems with an AC OPF in the lower level. We have not found any with the exact AC OPF, thus we single out two papers with AC OPF relaxations, [37] and [38]. Scalability and tractability issues are not discussed in these papers which is also one of the important points of this work.

The contribution of this paper consists of the following:

1) We introduce a novel numerical scheme for solving bilevel optimization problems based on deep convolutional neural networks. It is an evolutionary meta-modeling method that completely bypasses the lower-level problem. Our method successfully works with previously intractable, i.e. nonconvex, classes of the lower-level problems. As opposed to the existing techniques, solution times are basically independent on the upper-level problem size and scale well with the lower-level problem size.
2) We demonstrate the solution technique effectiveness by solving a price-maker energy storage AC-OPF-constrained market bidding problem. The results demonstrate higher achieved profits than with the DC market representation.

The paper is organized as follows. Section II provides mathematical foundation of the work and is divided in six subsections. Subsection II-A states the optimization problem, Subsection II-B explains how to approximate the lower level using a neural network, Subsection II-C describes the concept of fully connected neural networks, Subsection II-D explains the advantages, concept and our choice of hyperparameters of the used convolutional neural network, Subsection II-E describes the neural network training algorithm and Subsection II-F explains our iterative numerical scheme to solve the optimization problem at hand. The case study is presented in Section III with implementation details stated in Subsection III-A and results in Subsection III-B. The final Section IV concludes the paper.

II. Mathematical Modeling

A. Optimization model

In the following model we solve optimal ES bidding problem in the AC OPF network-constrained electricity market. The problem is of bilevel structure, i.e. the upper level maximizes the ES profit while the lower level maximizes social welfare due to supply and demand market bids. In the lower level we consider an exact nonconvex quadratic AC OPF formulation based on rectangular coordinates [39] notation written out in the Appendix, however, other notations such as polar [39] or current-voltage [40] are also applicable.

The upper-level problem consists of objective function (1.1), where λ_t is the electricity price in each hour indexed by t, and p_t^{ES} is the average ES power during one hour, i.e. energy, at the interface. Constraint (1.2) models the ES (dis)charging process, i.e. change in its state-of-energy SoE_t considering charging and discharging efficiencies $\boldsymbol{\eta}^{\mathbf{ch}}$ and $\boldsymbol{\eta}^{\mathbf{dis}}$. Constraint (1.3) sets limits to the state-of-energy (SoE), with $\overline{\boldsymbol{SoE}}$ being the maximum value. Constraints (1.4) and (1.5) limit the ES (dis)charged energy to $\overline{\boldsymbol{q}}^{\mathbf{ch}}$ for charging and $\overline{\boldsymbol{q}}^{\mathbf{dis}}$ for discharging. Binary variable x_t^{ch} disables simultaneous charging and discharging. Finally, equation (1.6) combines charging and discharging into a single variable p_t^{ES}. Optimization variables are written in formulas in normal font and contained in the variables set Ξ, while the parameters are written in bold font.

$$\underset{\Xi}{\mathrm{Max}} \ -\sum_t p_t^{\mathrm{ES}} \cdot \lambda_t \tag{1.1}$$

$$SoE_t = SoE_{t-1} + p_t^{\mathrm{ch}} \cdot \boldsymbol{\eta}^{\mathbf{ch}} - p_t^{\mathrm{dis}} / \boldsymbol{\eta}^{\mathbf{dis}}, \quad \forall t \tag{1.2}$$

$$0 \leqslant SoE_t \leqslant \overline{\boldsymbol{SoE}}, \quad \forall t \tag{1.3}$$

$$0 \leqslant p_t^{\mathrm{ch}} \leqslant \overline{\boldsymbol{q}}^{\mathbf{ch}} \cdot x_t^{\mathrm{ch}}, \quad \forall t \tag{1.4}$$

$$0 \leqslant p_t^{\mathrm{dis}} \leqslant \overline{\boldsymbol{q}}^{\mathbf{dis}} \cdot (1 - x_t^{\mathrm{ch}}), \quad \forall t \tag{1.5}$$

$$p_t^{\mathrm{ES}} = p_t^{\mathrm{ch}} - p_t^{\mathrm{dis}}, \quad \forall t \tag{1.6}$$

The lower level is only textually explained and not written here since we are bypassing it completely. The rectangular AC OPF consists of the objective function, the bus power balance constraints, the power flow equations, the line apparent power limits, the bus voltage limits, the generator production limits and the reference bus constraints. Mathematically challenging are the power flow equations and the lower bus voltage limit constraints, which do not conform to the traditional single-level reduction technique as they are nonconvex. Moreover, there are two additional convex, but nonlinear parts of the formulation. The considered objective function has quadratic cost coefficients and line apparent power limit constraints are of second-order cone form. Broader insights of different AC OPF formulations can be found in tutorial works such as [39].

Bypassing the lower level is based on the fact that the locational marginal prices λ_t are essentially a function of the upper-level p_t^{ES} variables, i.e. the objective can be expressed as $F\left(p_1^{\mathrm{ES}}, p_2^{\mathrm{ES}}, \ldots, p_{|\tau|}^{\mathrm{ES}}\right)$, where $|\tau|$ is cardinality of the time steps set τ. However, F can not be expressed explicitly, so we replace the objective function F with the approximation $\hat{F}$ from (1.7). The approximating function $\hat{F}$ is given as the feed-forward neural network, so it can be expressed explicitly in terms of elementary mathematical functions. Essentially, we are solving a single-level optimization meta-model that maximizes (1.7) subject to constraints (1.2)–(1.6).

$$\underset{\Xi}{\mathrm{Max}} \ \hat{F}\left(p_1^{\mathrm{ES}}, p_2^{\mathrm{ES}}, \ldots, p_{|\tau|}^{\mathrm{ES}}\right) \tag{1.7}$$

The problem belongs to the mixed-integer nonlinear optimization class due to the nonlinear neural network function $\hat{F}$ and due to x_t^{ch} being binary variables.

B. Lower level approximation using neural networks

For any function $f : U \subseteq \mathbb{R}^n \to \mathbb{R}^m$ there exists a NN that uniformly approximates the given function, see [41] and [42]. Typically, it is unknown how exactly to construct a specific NN, approximating the function f to the desired accuracy and using the smallest possible number of neurons. The first problem we encountered is the limited size of dataset used to train such NN. More precisely, each element in the dataset must be constructed by solving a single instance of the lower-level optimization problem for chosen values of the upper-level variables. Solving too many instances of the lower-level problem would take too long. On the other hand, the size of the dataset limits the maximum network size (the number of neurons), by limiting the number of parameters that define that particular network. NNs trained on a dataset that is small compared to the number of network parameters tend to overfit the training data and are poor in generalization on unseen data. In our case that would lead to lower accuracy of approximation of the lower-level problem solutions. Basically, the size of the dataset limits the accuracy of the NN approximation.

The second issue is in determining an optimal topology of a NN for a given network size, in order to achieve the greatest possible approximation accuracy. The optimal topology is dependent on an unknown function, which we are trying to approximate. Our first approach, using fully connected neural network with only few hidden layers, led to poor approximation accuracy. By carefully analyzing the properties of the lower-level optimization problem, the choice of a network topology was settled on a convolutional neural network (CNN) [43]. As the CNN architecture shares the same values of parameters between different parts of the network, the cumulative number of parameters is much smaller for the network of the same size, so the CNN architecture can be trained to approximate the original optimization problem to a higher accuracy. The first big success of the CNN architecture was in the area of computer vision, in the image classification problems [44].

The third obstacle we encountered was the generation of a dataset for the CNN training. Our first idea was to generate the dataset by uniform random sampling of the independent upper-level variables, only in intervals of their permissible values. Then for each sample, we solved the associated lower-level optimization problem. This strategy proved to be inefficient as the near-optimal values of the upper-level variables, which solve our bilevel optimization problem, are poorly represented by sampling these variables independently from the uniform distribution. It resulted in much higher approximation error of the CNN on the optimal solution than on the generated dataset. The solution proved to be in iterative refining of the generated dataset. In the first iteration we generate a uniform dataset on the whole permissible domain and find the solution of the approximation for the bilevel optimization problem. In each additional iteration we restrict the domain to an even smaller neighborhood around the approximated solution from the previous iteration. Then we generate a new uniform dataset on this smaller domain, train a new instance of the CNN, and using this new trained network, we again find a solution of the approximating problem. In each iteration we verify the quality of the current solution by computing the upper-level objective function exactly on optimal variables approximate problem values. We stop iterating when the actual value of the upper-level objective function stops improving.

C. Feed-forward fully connected neural networks

A feed-forward fully connected NN (see Figure 2) consists of K layers, where each layer consists of a number of neurons [45]. The first layer is referred to as the input layer, the last layer as the output layer, while the intermediate layers are called hidden layers. Neurons in each layer are connected only to the neurons in the neighboring layers. Feed-forward means that the data flows from the input layer to the output layer, strictly from one layer to the next one and in only one direction. Fully connected means that each neuron is connected to every neuron in the neighboring layers. Finally, each neuron in every hidden layer performs a nonlinear transformation on the data by applying the so-called activation function. More precisely, a feed-forward fully connected NN is a function $\mathcal{F} : \mathbb{R}^{N_1} \to \mathbb{R}^{N_K}$, where N_1 is the number of neurons in the input layer, and N_K is the number of neurons in the output layer. Function $\mathcal{F}$ is a composition of the alternating affine maps $\mathcal{A}_k : \mathbb{R}^{N_{k-1}} \to \mathbb{R}^{N_k}$, $k = 2, \dots, K$ and the element-wise nonlinear activation functions $\mathcal{N}_k : \mathbb{R}^{N_k} \to \mathbb{R}^{N_k}$, $k = 2, \dots, K-1$, such that

$$\mathcal{F} = \mathcal{A}_K \circ \mathcal{N}_{K-1} \circ \mathcal{A}_{K-1} \circ \dots \circ \mathcal{N}_3 \circ \mathcal{A}_3 \circ \mathcal{N}_2 \circ \mathcal{A}_2,$$

where N_k is the number of neurons in k-th layer.

Affine map $\mathcal{A}_k$ can be written in a matrix form as

$$\hat{\mathbf{z}}_k := \mathcal{A}_k(\mathbf{z}_{k-1}) = \mathbf{W}_k \mathbf{z}_{k-1} + \mathbf{b}_k, \quad \forall k = 1, \dots, K,$$

where weight matrix $\mathbf{W}_k$ has dimension $N_k \times N_{k-1}$ and bias vector $\mathbf{b}_k$ has dimension N_k.

For the activation functions we element-wise use the Softplus function,

$$z_k^i := \mathcal{N}_k^i(\hat{\mathbf{z}}_k) = \frac{\ln(1 + \exp(\beta \cdot \hat{z}_k^i))}{\beta}, \quad \begin{array}{l} \forall k = 1, \dots, K, \\ \forall i = 1, \dots, N_k, \end{array}$$

where $\hat{z}_k^i$ is the i-th component of vector $\hat{\mathbf{z}}_k$ and β is the hyperparameter of the Softplus function. Notice that for large values of β, Softplus uniformly converges to a rectified linear unit (ReLU) activation function (see Figure 3), which is given element-wise by

$$z_k^i = \max(\hat{z}_k^i, 0), \quad \forall k = 1, \dots, K, \quad \forall i = 1, \dots, N_k.$$

ReLU activation function is commonly used in recent NN applications. The reason why we decided to use Softplus will be become clear in Section II-F.

D. Convolutional neural networks

A CNN can be regarded as a sub-type of a feed forward NN. It is generally not fully connected, and a large number of weight and bias elements of matrices $\mathbf{W}_k$ and vectors $\mathbf{b}_k$ share the same values, as affine maps $\mathcal{A}_k$ are defined using the operation of matrix convolution [43].

Figure 4 depicts a deep CNN, describing the exact NN topology used in approximating function $\hat{F}$ of our problem. The structure of the NN is determined by an educated guess of the authors and by experimentation. Besides the input and output layers, we have six additional hidden layers. Unlike in a general NN, each layer in our CNN is described using the

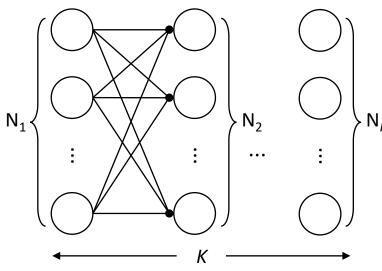


Fig. 2. Example of a fully connected neural network.

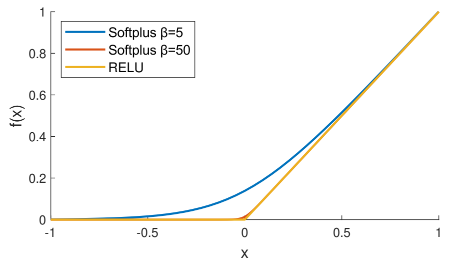


Fig. 3. Softplus and RELU plot.

layer length L_k and the number of channels C_k. The number of neurons in each layer is given by $N_k = L_k \cdot C_k$ and neurons are grouped in L_k groups of size C_k. In Figure 4, a single square depicts one group of neurons. The exact number of neurons within each group (the number of channels C_k) is written inside each square. The number of groups in each layer is given by a number written just below each layer. For instance, the total number of neurons in the second layer is equal to $24 \cdot 32 = 768$.

To define affine map $\mathcal{A}_k$, each layer in a CNN has an additional integer hyperparameter called the kernel size S_k. In Figure 4, only the first (input) layer and the last of the hidden layers have kernel sizes $S_1 = 1$ and $S_7 = 1$. All the other hidden layers have kernel size equal to 3. Notice that the kernel size is not applicable to the output layer, as the output layer only collects the output of the last hidden layer and is not applying any further affine maps. All kernels are depicted by a number of empty squares equal to the kernel size S_k. A downward arrow indicates that a kernel window is sliding over the layer in steps, performing a computation of the affine map. This means that the convolution operation can be in each step regarded as a smaller affine map $\hat{\mathcal{A}}_k$ that is defined only between the neurons in the groups covered by the kernel window in layer $k-1$ and the neurons in the single output group in layer k. In each step of the convolution operation on layer $k-1$ we use the same map $\hat{\mathcal{A}}_k$.

Each convolution layer has two additional integer hyperparameters called a stride and a padding size. The stride is the number of groups by which each kernel window moves in every step of the computing convolution operation. The first and second layers have the stride equal to 1 and the third to fifth layers have the stride equal to 2. This is the reason why the lengths L_k of the fourth to sixth layer are decreasing by a factor of 2. For the sixth and seventh layers, the stride is not applicable as the convolution operation is trivially performed only in the single possible position. The padding controls whether the kernel window can slide over the side of the layer or not. If we let the kernel windows slide over the side of the layer, as for the second to fifth layer, the padding is equal to 1 and we substitute zeros for the input in the convolution operation in place of the non-existing data. For the first, sixth and seventh layer, the padding is equal to 0, which means we do not let the kernel window slide over the side of the layer.

In matrix representation $\mathbf{W}_k$ of affine map $\mathcal{A}_k$, lot of matrix components are equal to zero and lot of other non-zero matrix components share the same values. We actually have, for the kernel size equal to 1 the block diagonal matrix $\mathbf{W}_k$, and for

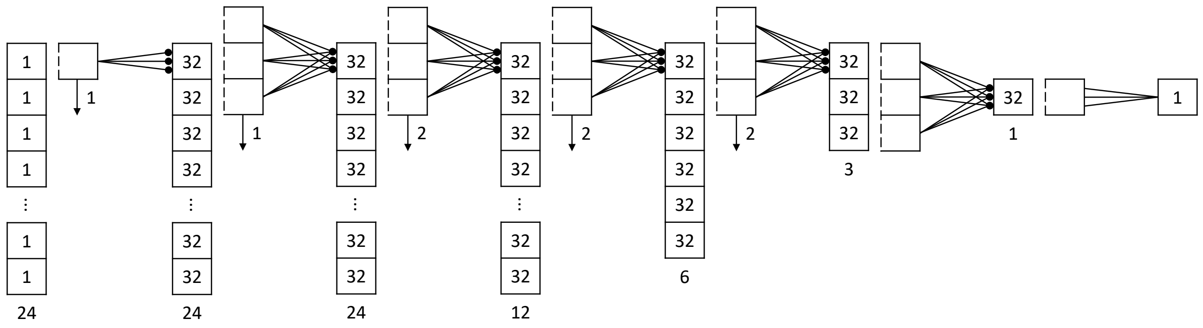


Fig. 4. Deep convolutional neural network structure.

$$\begin{pmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_n \end{pmatrix} \quad \begin{pmatrix} B_1 & C_1 & 0 & \cdots & 0 \\ A_2 & B_2 & C_2 & & \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ & & A_{n-1} & B_{n-1} & C_{n-1} \\ 0 & \cdots & 0 & A_n & B_n \end{pmatrix}$$

(a) (b)

Fig. 5. Block matrices: a) diagonal; b) tridiagonal

the kernel size equal to 3 the block tridiagonal matrix $\mathbf{W}_k$, see Figure 5. Every block is of size $C_k \times C_{k-1}$. For the kernel size equal to 1, every block in the block diagonal matrix $\mathbf{W}_k$ representing affine map $\mathcal{A}_k$ is exactly the same block. For the kernel size equal to 3, every 3 vertical blocks in the block tridiagonal matrix representation are exactly the same blocks. The bias vector $\mathbf{b}_k$ also has repeating components. Regardless on the kernel size, components of $\mathbf{b}_k$ repeat every C_k entries, which means each neuron group in a single layer shares the same biases.

Notice that a CNN, for the same number of neurons, typically has much lower number of parameters defining affine maps $\mathcal{A}_k$, than a fully connected NN. For instance, in our case the number of parameters defining map $\mathcal{A}_3$ is $C_2S_2C_3 + C_3 = 3104$, whether the number of parameters defining map $\mathcal{A}_3$ in a fully connected neural network with the same number of neurons would be $N_2N_3 + N_3 = 590592$.

E. Training feed-forward neural networks

To train a neural network simply means to optimize matrices $\mathbf{W}_k$ and vectors $\mathbf{b}_k$ in order to minimize a chosen loss function over a dataset. Since we use CNNs, we must respect the block diagonal or tridiagonal structure of matrices $\mathbf{W}_k$ having many shared values, as discussed in Section II-D.

The dataset is generated by solving a number of instances of the lower-level optimization problem, for as many random samples of vector $\left(p_1^{\mathrm{ES}}, p_2^{\mathrm{ES}}, \ldots, p_{|\tau|}^{\mathrm{ES}}\right)$, and subsequently evaluating value of the target objective function (1.1), i.e. $F\left(p_1^{\mathrm{ES}}, p_2^{\mathrm{ES}}, \ldots, p_{|\tau|}^{\mathrm{ES}}\right)$.

The loss function is the mean squared error between the NN computed value $\hat{F}\left(p_1^{\mathrm{ES}}, p_2^{\mathrm{ES}}, \ldots, p_{|\tau|}^{\mathrm{ES}}\right)$ and the lower-level exact solution $F\left(p_1^{\mathrm{ES}}, p_2^{\mathrm{ES}}, \ldots, p_{|\tau|}^{\mathrm{ES}}\right)$.

As customary in NN training, the optimization of $\mathbf{W}_k$ and $\mathbf{b}_k$ is performed using a variant of a gradient descent optimizer. The dataset is split into training and validation datasets, using $80 : 20$ percent split ratio. The optimization is done in multiple epochs over the training dataset, where gradients are computed using backpropagation [46] and automatic differentiation [47] of NN. The validation dataset is used only for evaluating the loss function after every epoch of training and not for a gradient computation. Computed values of the loss function on the validation dataset are used to asses numerical viability of the training process and to select the best trained network, having the lowest value of the validation loss. Before the start of the first epoch of training, $\mathbf{W}_k$ and $\mathbf{b}_k$ are initialized to random values.

More details about our exact training procedure, together with all optimizer and training hyperparameter values are given in Section III-A.

F. Meta-optimization numerical scheme

We devised an iterative numerical scheme for computing a sequence of CNN approximations $\hat{F}_i$ of the otherwise intractable objective function F. In each iteration i we first generate dataset $\mathcal{D}_i$ of random sampled vectors $\left(p_{1,i}^{\mathrm{ES}}, p_{2,i}^{\mathrm{ES}}, \ldots, p_{|\tau|,i}^{\mathrm{ES}}\right)$. For each $t \in \tau$, values of $p_{t,i}^{\mathrm{ES}}$ are independently and uniformly random sampled from the interval centered around $p_{t,i}^{\mathrm{ES,cnt}}$ of length at most $2p_{t,i}^{\mathrm{ES,rad}}$, respecting the condition $-p_t^{\mathrm{dis}} \leqslant p_{t,i}^{\mathrm{ES}} \leqslant p_t^{\mathrm{ch}}$. In the first iteration we set $p_{t,i}^{\mathrm{ES,cnt}}$ equal to zero and $p_{t,i}^{\mathrm{ES,rad}}$ such that it allows all permissible values of p_t^{ES}, i.e. $-p_t^{\mathrm{dis}} \leqslant p_{t,i}^{\mathrm{ES}} \leqslant p_t^{\mathrm{ch}}$.

In practice, we noticed that our numerical scheme runs better if we introduce an additional small relative tolerance $\epsilon > 0$ on the dataset creation. Thus, for the maximum length of the sampling interval we actually use $2p_{t,i}^{\mathrm{ES,rad}}(1+\epsilon)$ and allow all permissible values of p_t^{ES} to be from the interval $-p_t^{\mathrm{dis}}(1+\epsilon) \leqslant p_{t,i}^{\mathrm{ES}} \leqslant p_t^{\mathrm{ch}}(1+\epsilon)$. The intuition behind introducing the tolerance is that our CNN would better approximate objective function F for parameter values $p_{t,i}^{\mathrm{ES}}$ near the boundary values $-p_t^{\mathrm{dis}}$ and p_t^{ch}, if the dataset is allowed to include values $p_{t,i}^{\mathrm{ES}}$ a bit outside the interval $[-p_t^{\mathrm{dis}}, p_t^{\mathrm{ch}}]$.

Each element in dataset $\mathcal{D}_i$ is now a pair of random vector $\left(p_{1,i}^{\mathrm{ES}}, p_{2,i}^{\mathrm{ES}}, \ldots, p_{|\tau|,i}^{\mathrm{ES}}\right)$ and a value $F\left(p_{1,i}^{\mathrm{ES}}, p_{2,i}^{\mathrm{ES}}, \ldots, p_{|\tau|,i}^{\mathrm{ES}}\right)$ of the target objective function (1.1), computed by solving a single instance of the lower-level problem.

Next, in each iteration we separately train a number of CNNs, $\hat{F}_{i,n}$ indexed by $n \in \{1, \ldots, M\}$, to approximate function F. For every additional training on the same dataset $\mathcal{D}_i$ we obtain a subtly different CNN, as the training process is intrinsically stochastic (random initialized network weights and random sampled stochastic gradient descent mini batches).

Now, as every trained CNN is a function $\hat{F}_{i,n}$, we symbolically insert function $\hat{F}_{i,n}$ into the upper-level problem objective function. For this we use our modeling environment's, which is AMPL, defining variables feature. Defining variables are a type of variables that are substituted out, potentially in a nested way, by their declaration expression before reaching the solver. This results in solving a single-level optimization meta-model which maximizes

$$\underset{\Xi}{\mathrm{Max}}\ \hat{F}_{i,n}\left(p^{\mathrm{ES}}_{1,i}, p^{\mathrm{ES}}_{2,i}, \ldots, p^{\mathrm{ES}}_{|\tau|,i}\right) \tag{1.8}$$

subject to constraints (1.2)–(1.3) and

$$0 \leqslant p^{\mathrm{ch}}_{t,i} \leqslant \overline{\boldsymbol{q}}^{\mathbf{ch}} \cdot x^{\mathrm{ch}}_{t,i}, \quad \forall t \tag{1.9}$$

$$0 \leqslant p^{\mathrm{dis}}_{t,i} \leqslant \overline{\boldsymbol{q}}^{\mathbf{dis}} \cdot (1 - x^{\mathrm{ch}}_{t,i}), \quad \forall t \tag{1.10}$$

$$p^{\mathrm{ES}}_{t,i} = p^{\mathrm{ch}}_{t,i} - p^{\mathrm{dis}}_{t,i}, \quad \forall t \tag{1.11}$$

$$p^{\mathrm{ES,cnt}}_{t,i} - p^{\mathrm{ES,rad}}_{t,i} \leqslant p^{\mathrm{ES}}_{t,i} \leqslant p^{\mathrm{ES,cnt}}_{t,i} + p^{\mathrm{ES,rad}}_{t,i}, \quad \forall t, \tag{1.12}$$

for every trained CNN indexed by n. The additional constraint (1.12) is used to respect that dataset $\mathcal{D}_i$ is created centered around $p^{\mathrm{ES,cnt}}_{t,i}$ using an interval length at most $2p^{\mathrm{ES,rad}}_{t,i}$.

Notice, as we used the differentiable Softplus activation function in our CNN, and as a CNN is just a composition of affine maps and element-wise activation functions, that $\hat{F}_{i,n}$ are differentiable functions. In case of using a ReLU activation function, the resulting $\hat{F}_{i,n}$ would not be differentiable. We experimentally established that lower values of the Softplus hyperparameter β result in lower overall neural network approximation accuracy, and higher values give rise to solver instabilities in the single-level meta-model optimization step. We also tried to use ReLU instead of Softplus, but we were plagued with solver instabilities and slowdown. Using Softplus showed to be much more efficient.

For every trained CNN we now have the computed profit $\mathcal{C}_{i,n} = \underset{\Xi}{\mathrm{Max}}\ \hat{F}_{i,n}(p^{\mathrm{ES}}_{t,i})$ and the computed optimal ES (dis)charged energy $\left(p^{\mathrm{ES}}_{1,i,n}, p^{\mathrm{ES}}_{2,i,n}, \ldots, p^{\mathrm{ES}}_{|\tau|,i,n}\right)$. Now we can verify what are the actual profits obtained for the computed optimal ES (dis)charging schedule. This is done by optimizing the lower-level problem independently of the upper level with fixed ES (dis)charging schedule. The actual profit is determined as $\mathcal{V}_{i,n} = \underset{\Xi}{\mathrm{Max}} - \sum_t p^{\mathrm{ES}}_{t,i,n} \cdot \lambda_t$, where λ_t is in this case the bus balance constraint marginal, computed by default by many interior point solvers.

Additionally, we compute mean optimal ES energy exchange quantities

$$\overline{p^{\mathrm{ES}}}_{t,i} = \frac{1}{M} \sum_{n=1}^{M} p^{\mathrm{ES}}_{t,i,n} \quad \forall t \in \tau$$

and for the computed mean vector $\left(\overline{p^{\mathrm{ES}}_{1,i}}, \overline{p^{\mathrm{ES}}_{2,i}}, \ldots, \overline{p^{\mathrm{ES}}_{|\tau|,i}}\right)$ we again optimize the lower-level problem with fixed ES charging values to the obtained mean optimal vector, arriving at the mean actual profit $\overline{\mathcal{V}_i} = -\sum_t \overline{p^{\mathrm{ES}}}_{t,i} \cdot \lambda_t$. Considering the mean actual profit is justified, because averaging over optimal solutions of many different CNN models $\hat{F}_{i,n}$, each one approximating an intractable objective function F, it can result in a better mean solution. By looking at the actual test results (i.a. Tables III and IV) we see that this approach is justified in practice, i.e., in some iterations the mean actual profit $\overline{\mathcal{V}_i}$ can be higher than any of the actual profits $\mathcal{V}_{i,n}$.

Finally, we have to choose values of $p^{\mathrm{ES,cnt}}_{t,i+1}$ and $p^{\mathrm{ES,rad}}_{t,i+1}$ for a next iteration of the meta-optimization scheme. For $p^{\mathrm{ES,cnt}}_{t,i+1}$ we either chose the optimal ES (dis)charging quantities $\left(p^{\mathrm{ES}}_{1,i,n}, p^{\mathrm{ES}}_{2,i,n}, \ldots, p^{\mathrm{ES}}_{|\tau|,i,n}\right)$ from CNN that achieved the highest actual profit $\mathcal{V}_{i,n}$, or in the case $\overline{\mathcal{V}_i}$ is the highest profit, we chose the mean optimal ES (dis)charging quantities $\left(\overline{p^{\mathrm{ES}}_{1,i}}, \overline{p^{\mathrm{ES}}_{2,i}}, \ldots, \overline{p^{\mathrm{ES}}_{|\tau|,i}}\right)$.

To choose $p^{\mathrm{ES,rad}}_{t,i+1}$, we first compute the maximum over all $t \in \tau$ of the standard deviations of samples $\left\{p^{\mathrm{ES}}_{t,i,n} : 1 \leqslant n \leqslant M\right\}$. More precisely, we compute

$$\sigma_i := \underset{t\in\tau}{\mathrm{Max}}\ \mathrm{std}\left\{p^{\mathrm{ES}}_{t,i,n} : 1 \leqslant n \leqslant M\right\}.$$

For the next iteration we take a smaller value between the current $p^{\mathrm{ES,rad}}_{t,i}$ and a new estimate,

$$p^{\mathrm{ES,rad}}_{t,i+1} := \mathrm{Min}\{p^{\mathrm{ES,rad}}_{t,i},\ \gamma \cdot \sigma_i\},$$

for every $t \in \tau$, where γ is a hyperparameter. Note that $p^{\mathrm{ES,rad}}_{t,i+1}$ does not depend on t. It has the same value for every t.

For the next iteration of our meta-optimization scheme, dataset D_{i+1} is created around $p^{\mathrm{ES,cnt}}_{t,i+1}$, which is the best optimal ES (dis)charging schedule computed in the current iteration, i.e., ES charging and discharging energy quantities that produce the highest profit. The dataset width, which is decided by $p^{\mathrm{ES,rad}}_{t,i+1}$, is influenced by how close together are optimal ES schedules predicted by different CNNs trained in the current iteration. In case different CNNs produce relatively close optimums, the computed standard deviation σ_i is relatively small, and a next iteration dataset is going to be concentrated around a smaller neighborhood of $p^{\mathrm{ES,cnt}}_{t,i+1}$. On the contrary, in case different CNNs produce optimums that are more apart, the computed standard deviation σ_i is relatively large, and a next iteration dataset is going to span over a bigger neighborhood of $p^{\mathrm{ES,cnt}}_{t,i+1}$. Notice that $p^{\mathrm{ES,rad}}_{t,i}$ is non-increasing between iterations.

In the end, we have to prescribe a stopping criterion for our meta-optimization scheme. We choose to stop further iterations if there is no improvement in the actual profit of the current iteration compared to the previous one. We empirically conclude (see Section III-B) that the convergence of our meta-optimization scheme is achieved in few iterations (see Tables III, IV and V). An overview of the numerical optimization scheme is provided in Algorithm 1.

III. Case Study

A. Implementation details

For the dataset creation, each instance of the lower-level problem, one for every dataset entry, was solved using AMPL running KNITRO 12.3 solver. A single dataset entry consists of a 24-dimensional floating point vector $\left(p^{\mathrm{ES}}_1, p^{\mathrm{ES}}_2, \ldots, p^{\mathrm{ES}}_{|\tau|}\right)$ as an input and a single floating point value as an output, which corresponds to the computed upper-level profit for given $\left(p^{\mathrm{ES}}_1, p^{\mathrm{ES}}_2, \ldots, p^{\mathrm{ES}}_{|\tau|}\right)$ values (computed as $-\sum_t p^{\mathrm{ES}}_t \cdot \lambda_t$, where

λ_t is the marginal of an active power bus balance constraint at the ES location). In total, we used 10^5 dataset entries and thus the same number of independent lower levels to be solved. To reduce the computation time, computations were carried out in parallel running on a dual Intel Xeon CPU computer system over a total of 40 physical cores.

We implemented the CNN depicted in Figure 4 in Python using PyTorch library [48]. CNN hyperparameters are described in in Section II-D and the hyperparameter β of the Softplus activation function is set to 50.

Training of the CNN was implemented using fast.ai library [49] using training procedure similar as in [50]. We used the Ranger algorithm [51], employing the RAdam optimizer [52], the parameter lookahead [53], and the flat-cosine one-cycle policy [54]. After experimenting with training thousands of models we set RAdam hyperparameters to the following values: the number of training epochs to 500, the maximum learning rate to 0.003, the training batch size to 128, the weight decay factor to 0.01 and the exponential decay rates of the first and second moments to values 0.95 and 0.85.

In every iteration of the meta-optimization scheme we trained a total of $M = 60$ CNNs on the same dataset. Our computer system was equipped with 6 Nvidia Quadro RTX 6000 GPUs, each having 16 Gb of RAM, so we could train all CNNs in parallel, training 10 CNNs per GPU.

In the meta-optimization scheme there are two hyperparameters to consider. After experimenting with different values, for the relative tolerance of the dataset creation we take $\epsilon = 0.1$, and for the other hyperparameter we take $\gamma = 5$. A value of hyperparameter γ influences the decreasing rate of $p_{t,i}^{\mathrm{ES,rad}}$ through subsequent iterations. Using lower values of γ produces lower values of $p_{t,i}^{\mathrm{ES,rad}}$ in later iterations, which can lead to a sub-optimal optimization result in the end, and a higher value tends to slow down the speed of convergence. Table I shows the $p_{t,i}^{\mathrm{ES,rad}}$ decrease throughout all iterations from the case study. Notice the difference in speed and intensity of $p_{t,i}^{\mathrm{ES,rad}}$ decrease in different transmission networks. More significant and faster decrease, as seen in e.g. 3_lmbd, suggests that all 60 trained CNNs yield closer optimums, which can be explained by inherently tamer underlying optimization landscape. On the contrary, much less significant decrease in $p_{t,i}^{\mathrm{ES,rad}}$, as seen in e.g. 73_ieee_rts, suggests that CNNs are struggling more to approximate the optimums, which is probably induced by more demanding optimization landscape.

Algorithm 1 Numerical optimization scheme

```
1: repeat
2:     Generate a new random dataset ($10^5$ entries)
3:     Evaluate LL response for the dataset
4:     Train 60 NNs to approximate LL response
5:     Optimize the ULs with inserted NNs into objective
       function
6:     Determine actual profits by optimizing LL with fixed
       ES (dis)charging schedule
7:     Select the best actual solution out of:
         • the best direct result;
         • the result obtained averaging decisions from all
           optimized NNs;
8:     For the next iteration, reduce and concentrate the
       dataset spatial size in the neighborhood of the best
       solution found from this iteration
9: until The best solution is worse than in the preceding
   iteration
```

B. Results

We tested our method on four separate transmission system meshed networks from PGLib-OPF [55] library: 3_lmbd, 57_ieee, 73_ieee_rts and 300_ieee. Topologies of the three large networks are provided in Fig. 11 in the Appendix. The three-bus network is of typical triangle topology. A time dimension was added to the data by applying the load scaling factors for winter workdays available from IEEE RTS-96 [56]. Set of time steps τ has 24 elements for different hours in a single day. In case of 73_ieee_rts network, we also applied 0.85 scaling factor to the transmission lines capacities to induce congestion. The networks were otherwise unmodified. For an ES to have an impact on the energy market prices, a feature for which the bilevel modeling is used for, it has to be very large. Thus, we model the ES with 100 MWh (1 p.u.) capacity. Charging and discharging efficiencies were both set to 90% and maximum ES (dis)charging power to 60 MW.

Table II presents an average wall time per iteration for each step of the proposed meta-optimization scheme. Dataset creation is a cumulative time for three sub-steps: generation of 10^5 random vectors of p_t^{ES}, solving lower-level problems for every dataset entry, and data format post-processing of the generated dataset, which mostly include disk input-output (IO) operations. Most of the time is consumed for solving 10^5 lower-level problems. Notice that larger transmission system networks require more solver time. NN training is the time consumed for parallel CNN models training. This time does not depend on the transmission system network size, as it depends solely on the CNN and training hyperparameters. The last step is solving the meta-models, which is performed sequentially for all 60 trained CNNs. A possible speedup of using parallel computations in this step would not be significant compared to the total time used per iteration.

TABLE I
VALUES OF $p_{t,i}^{\mathrm{ES,rad}}$ FOR DIFFERENT TRANSMISSION NETWORKS AND ITERATIONS

Iter	3_lmbd	57_ieee	73_ieee_rts	300_ieee
1	0.6	0.6	0.6	0.6
2	0.0618	0.3265	0.4894	0.2302
3	0.0319	0.2123	0.3465	
4	0.0129	0.1876	0.3465	
5	0.0129			
6	0.0129			

TABLE II
AVERAGE PROCESSING TIME IN SECONDS PER SINGLE ITERATION OF THE META-OPTIMIZATION SCHEME.

	Dataset creation [s]	NN training [s]	Solving meta-models (mean ± std) [s]	Component total time [s]
3_lmbd	471	5118	$60\times(7.7 \pm 10.0)$	6051
57_ieee	1774	5129	$60\times(11.7 \pm 10.6)$	7605
73_ieee_rts	3644	5125	$60\times(4.0 \pm 2.6)$	9009
300_ieee	28256	5245	$60\times(2.0 \pm 1.9)$	33621

Time for solving the meta-models can vary greatly between different CNNs and different iterations. We consider this to be a normal solver behavior due to binary variables x_t^{ch}. For the dataset creation and the NN training the average wall time is pretty much unchanged between different iterations, so we supply only the mean times without the standard deviation. Component total time is a sum of the dataset creation time, the NN training time, and the mean time for solving meta-models. Our model is highly scalable as long as the lower-level problem can be evaluated a number of times under reasonable time and resources. An alternative method from our two-part paper [12] and [13] has scalability issues when using lower levels with larger networks since it computes the lower and the upper level simultaneously so the solution process can diverge. Table II indicates that our method scales reasonably even for 73 and 300 bus systems.

Tables III–VI present optimization results in terms of the ES computed and actual profits acquired in four different transmission systems. Actual profits are obtained in the verification Step 6 of Algorithm 1 by optimizing the lower level with fixed ES charging decisions as explained in Section II-F. We also compare actual profits achieved by our method to actual profits achieved by solving a bilevel ES market optimal bidding using i) the AC OPF model and single-level reduction approach from the two-part paper [12] and [13] and ii) a standard DC OPF [35] model in the lower level. AC OPF single-level reduction approach results in slightly higher actual profits compared to the NN approach, but its solution process fails to converge for 73- and 300-bus networks. Actual DC OPF profits are profits that would occur in the AC OPF market, but by using bidding decisions from the DC OPF bilevel model. In our tables, the best NN computed profit is the maximum value of $\mathcal{C}_{i,n}$ over all 60 NNs, the best NN actual profit is the maximum value of $\mathcal{V}_{i,n}$ over all 60 NNs, and the mean p_t^{ES} actual profit is $\overline{\mathcal{V}_i}$, where i is the iteration number and NNs are indexed by n. By design, the best profit is always achieved in the penultimate iteration of our method, as a worse profit in the last iteration actually triggers the stopping criterion. The number of required iterations differs between the transmission systems and ranges from 2 to 6. Tables III–VI demonstrate we also achieved a high first iteration accuracy, since the greatest second iteration improvement of the actual profit is only 0.03%. Note that the total time per iteration presented is somewhat higher than a component total time in Table II, as it includes an additional overhead for some data reformatting and IO disk operations. Also, note that we decided to present profits using up to four decimal places, so that small improvements between subsequent iterations in Table III become visible, which also reaffirms the optimality of the first iteration result. Relative differences in profits using all three approaches (single-level reduction, NN and bilevel DC OPF) are clearly presented in Figure 6 where the highest profits are normalized to 100%. On 3- and 57-bus networks, the NN approach achieved 99.9987% and 99.9848% of ES profits of the single-level reduction. Profit increase over the DC OPF model was 1.5% for 3_lmbd, 1.5% for 57_ieee, 10.5% for 73_ieee network, and 16.4% for 300_ieee network.

Figures 7–10 present ES (dis)charging profiles for all four test cases. The 3-bus network is characterized with relatively small charging and discharging powers (up to 10 MW). In such small network, the ES (dis)charging would change the marginal producer, thus significantly affecting the market prices. In the 57-bus network the DC model discharges at over 40 MW in hour 12. Since it is a lossless model, it does not predict any price change due to ES arbitrage. On the other hand, the AC model captures price changes incurred by the ES and charges more evenly throughout the day. The 73-bus network features high price volatility, thus the ES makes the most cycles. In the 300-bus network, the DC

TABLE III
PROFITS FOR 3_LMBD NETWORK (ES AT BUS 3).

Iter	Total time [s]	Mean p_t^{ES} actual profit	Best NN actual profit	Best NN computed profit	Single-level reduction actual profit [12], [13]	DC OPF actual profit
1	8386	2016.0583	**2016.2954**	2011.5968	2016.8762	1986.4979
2	7158	**2016.8395**	2016.7695	2018.1744		
3	6965	**2016.8414**	2016.8271	2017.5707		
4	6910	**2016.8416**	2016.8339	2016.9533		
5	6903	**2016.8505**	2016.8015	2017.2495		
6	6882	2016.8452	2016.8096	2017.1350		

TABLE IV
PROFITS FOR 57_IEEE NETWORK (ES AT BUS 1).

Iter	Total time [s]	Mean p_t^{ES} actual profit	Best NN actual profit	Best NN computed profit	Single-level reduction actual profit [12], [13]	DC OPF actual profit
1	8305	1564.0254	**1564.1351**	1564.3315	1565.2053	1542.5985
2	8724	1564.5929	**1564.6218**	1572.8308		
3	8886	**1564.9676**	1564.8713	1568.1618		
4	9088	1564.9433	1564.9370	1570.1936		

TABLE V
PROFITS FOR 73_IEEE_RTS NETWORK (ES AT BUS 101).

Iter	Total time [s]	Mean p_t^{ES} actual profit	Best NN actual profit	Best NN computed profit	Single-level reduction actual profit [12], [13]	DC OPF actual profit
1	10234	5503.2526	**5512.4143**	5525.5078	No solution	4990.1637
2	10186	5503.2608	**5512.6547**	5520.6102		
3	10159	5503.8406	**5512.7558**	5519.5750		
4	10192	5503.7870	5512.1032	5513.6247		

TABLE VI
PROFITS FOR 300_IEEE NETWORK (ES AT BUS 1).

Iter	Total time [s]	Mean p_t^{ES} actual profit	Best NN actual profit	Best NN computed profit	Single-level reduction actual profit [12], [13]	DC OPF actual profit
1	33482	1396.5797	**1397.0175**	1377.6627	No solution	1199.9745
2	36481	1396.7220	1397.0172	1399.9574		

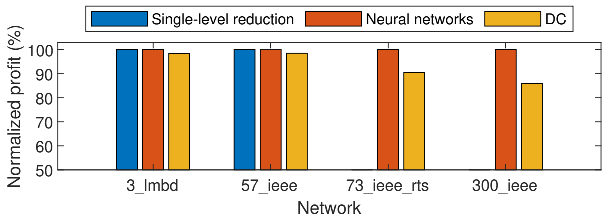


Fig. 6. Normalized profits with different approaches over four different transmission system meshed networks.

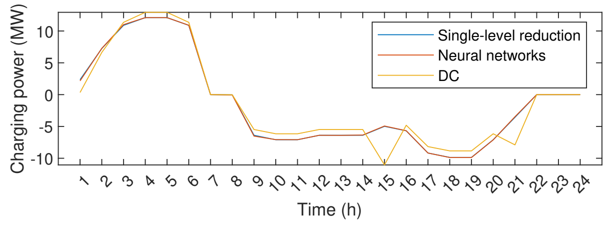


Fig. 7. Charging profile for 3_lmbd network at bus 3.

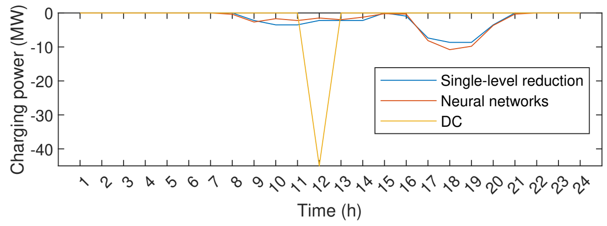


Fig. 8. Charging profile for 57_ieee network at bus 1.

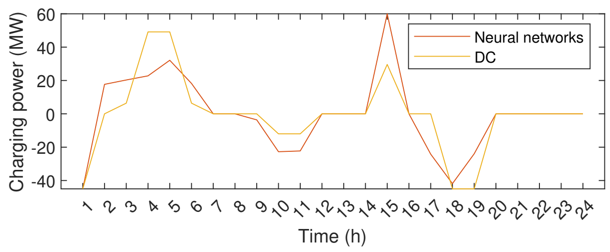


Fig. 9. Charging profile for 73_ieee_rts network at bus 101.

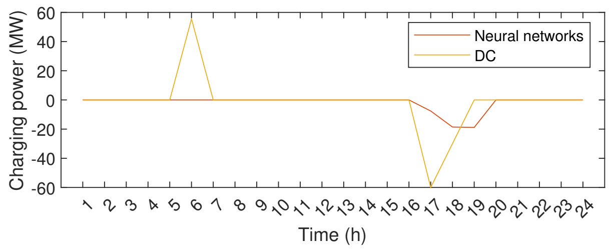


Fig. 10. Charging profile for 300_ieee network at bus 1.

model also assumes no price impact by its actions, which is incorrect, while the NN approach, which better captures the price changes due to the ES (dis)charging, is more conservative and discharges at low power in ours 17-19. Figures7–8 also show that the NN approach produces almost identical ES charging profile as the single-level reduction approach from [12] and [13].

IV. Conclusion

This paper presents a novel numerical method which utilizes deep convolutional neural networks to efficiently solve a wide class of bilevel optimization problems arising in deregulated power systems. The method uses evolutionary meta-modeling to bypass the lower-level problem, thus it is insensitive to the lower-level complexity, which is the main culprit in rendering bilevel optimization problems intractable. the model successfully deals with nonlinear nonconvex lower-levels that include binary variables, as long as the lower-level can be efficiently solved as a single-level problem by treating all upper-level variables as parameters. We demonstrate the application of the method to solve the ES market participation problem using AC OPF in the lower level, which enables electricity market operators to perform highly accurate market clearing procedure. However, the proposed framework is generally applicable for any other bilevel optimization problem in the power systems domain.

Additionally, our method is scalable in terms of the required precision versus the run time. Using larger training datasets we could train an NN having more parameters than we used here. We would obtain more accurate optimums, but would also require longer run time. On the other hand, if we are satisfied with lower precision, we could use a smaller dataset for training a smaller neural network, and our method would run faster.

Finally, we note that this procedure can be easily implemented to trilevel models, as they are generally solved by first merging the middle- and the lower-level problems into a mixed-integer problem with equilibrium constraints, see e.g. [57], which is a direct application of our proposed procedure. The obtained single-level problem then acts as a lower-level problem to the original upper-level problem. The resulting bilevel structure is commonly iteratively solved using a cutting plane algorithm. However, a direct implementation of our procedure to trilevel problems will be explored in future.

V. Appendix: Lower Level

This section includes the formulation of the lower level, i.e. the exact AC OPF in the rectangular coordinates. Objective function (A.1) minimizes production costs, (A.2) and (A.3) are bus balances, (A.4)–(A.7) are power flow equations, (A.8) and (A.9) are generator production limits, (A.10) is line thermal limit, (A.11) is voltage limit and (A.12) is reference bus constraint. $V^{\boldsymbol{r}}_{t,i}$ and $V^{\boldsymbol{i}}_{t,i}$, $\boldsymbol{\tau}^{\mathbf{r}}_{e}$ and $\boldsymbol{\tau}^{\mathbf{i}}_{e}$ are real and imaginary parts of voltage magnitude and transformer tap ratio respectively. All other notations are the same as in our previous paper [11]. The simulated bidding process assumes the strategic market participant is the one in the upper level, while bids of all other participants are deterministic. Such modeling is a common practice and is described in detail in [58]. In reality, sufficient historical offering data to derive other participants' offering curves are available in some markets, e.g. see [59] for the Alberta market.

$$\underset{\Xi^{\mathrm{ll}}}{\mathrm{Min}} \sum_{t,k} (\ddot{\boldsymbol{c}}_k \cdot (P^{\mathrm{g}}_{t,k})^2 + \dot{\boldsymbol{c}}_k \cdot P^{\mathrm{g}}_{t,k} + \boldsymbol{c}_k) \tag{A.1}$$

$$\begin{aligned} &\sum_{k\in G_i} P^{\mathrm{g}}_{t,k} - \sum_{l\in L_i} \boldsymbol{P}^{\mathbf{d}}_{t,l} - \sum_{(e,i,j)\in E\cup E^{\mathrm{R}}} P_{t,e,i,j} - \underset{:i\in\beta}{p^{\mathrm{ES}}_{t}} \\ &- ((V^{\boldsymbol{r}}_{t,i})^2 + (V^{\boldsymbol{i}}_{t,i})^2) \cdot \sum_{s\in S_i} \boldsymbol{g}^{\mathbf{sh}}_{s} = 0, \quad \forall t,i \quad : \lambda_{t,i} \end{aligned} \tag{A.2}$$

$$\begin{aligned} &\sum_{k\in G_i} Q^{\mathrm{g}}_{t,k} - \sum_{l\in L_i} \boldsymbol{Q}^{\mathbf{d}}_{t,l} - \sum_{(e,i,j)\in E\cup E^{\mathrm{R}}} Q_{t,e,i,j} \\ &+ ((V^{\boldsymbol{r}}_{t,i})^2 + (V^{\boldsymbol{i}}_{t,i})^2) \cdot \sum_{s\in S_i} \boldsymbol{b}^{\mathbf{sh}}_{s} = 0, \quad \forall t,i \end{aligned} \tag{A.3}$$

$$\begin{aligned} &P_{t,e,i,j} = ((V^{\boldsymbol{r}}_{t,i})^2 + (V^{\boldsymbol{i}}_{t,i})^2) \cdot (\boldsymbol{g}_e + \boldsymbol{g}^{\mathbf{fr}}_{e}) / \boldsymbol{\tau}^2_e + \\ &((-\boldsymbol{g}_e \cdot \boldsymbol{\tau}^{\mathbf{r}}_{e} + \boldsymbol{b}_e \cdot \boldsymbol{\tau}^{\mathbf{i}}_{e}) \cdot (V^{\boldsymbol{r}}_{t,i} \cdot V^{\boldsymbol{r}}_{t,j} + V^{\boldsymbol{i}}_{t,i} \cdot V^{\boldsymbol{i}}_{t,j}) + \\ &(\boldsymbol{b}_e \cdot \boldsymbol{\tau}^{\mathbf{r}}_{e} + \boldsymbol{g}_e \cdot \boldsymbol{\tau}^{\mathbf{i}}_{e}) \cdot (V^{\boldsymbol{r}}_{t,i} \cdot V^{\boldsymbol{i}}_{t,j} - V^{\boldsymbol{i}}_{t,i} \cdot V^{\boldsymbol{r}}_{t,j})) / \boldsymbol{\tau}^2_e, \\ &\forall t, (e,i,j) \in E \end{aligned} \tag{A.4}$$

$$\begin{aligned}&P_{t,e,i,j} = ((V^{\boldsymbol{r}}_{t,i})^2 + (V^{\boldsymbol{i}}_{t,i})^2)\cdot(\boldsymbol{g}_e + \boldsymbol{g}^{\mathbf{to}}_e)+\\&((-\boldsymbol{g}_e\cdot\boldsymbol{\tau}^{\mathbf{r}}_e - \boldsymbol{b}_e\cdot\boldsymbol{\tau}^{\mathbf{i}}_e)\cdot(V^{\boldsymbol{r}}_{t,i}\cdot V^{\boldsymbol{r}}_{t,j} + V^{\boldsymbol{i}}_{t,i}\cdot V^{\boldsymbol{i}}_{t,j})+\\&(\boldsymbol{b}_e\cdot\boldsymbol{\tau}^{\mathbf{r}}_e - \boldsymbol{g}_e\cdot\boldsymbol{\tau}^{\mathbf{i}}_e)\cdot(V^{\boldsymbol{r}}_{t,i}\cdot V^{\boldsymbol{i}}_{t,j} - V^{\boldsymbol{i}}_{t,i}\cdot V^{\boldsymbol{r}}_{t,j}))/\boldsymbol{\tau}^2_e,\\&\forall t,(e,i,j)\in E^{\mathrm{R}}\end{aligned}\tag{A.5}$$

$$\begin{aligned}&Q_{t,e,i,j} = -((V^{\boldsymbol{r}}_{t,i})^2 + (V^{\boldsymbol{i}}_{t,i})^2)\cdot(\boldsymbol{b}_e + \boldsymbol{b}^{\mathbf{fr}}_e)/\boldsymbol{\tau}^2_e+\\&((\boldsymbol{b}_e\cdot\boldsymbol{\tau}^{\mathbf{r}}_e + \boldsymbol{g}_e\cdot\boldsymbol{\tau}^{\mathbf{i}}_e)\cdot(V^{\boldsymbol{r}}_{t,i}\cdot V^{\boldsymbol{r}}_{t,j} + V^{\boldsymbol{i}}_{t,i}\cdot V^{\boldsymbol{i}}_{t,j})+\\&(\boldsymbol{g}_e\cdot\boldsymbol{\tau}^{\mathbf{r}}_e - \boldsymbol{b}_e\cdot\boldsymbol{\tau}^{\mathbf{i}}_e)\cdot(V^{\boldsymbol{r}}_{t,i}\cdot V^{\boldsymbol{i}}_{t,j} - V^{\boldsymbol{i}}_{t,i}\cdot V^{\boldsymbol{r}}_{t,j}))/\boldsymbol{\tau}^2_e,\\&\forall t,(e,i,j)\in E\end{aligned}\tag{A.6}$$

$$\begin{aligned}&Q_{t,e,i,j} = -((V^{\boldsymbol{r}}_{t,i})^2 + (V^{\boldsymbol{i}}_{t,i})^2)\cdot(\boldsymbol{b}_e + \boldsymbol{b}^{\mathbf{fr}}_e)/\boldsymbol{\tau}^2_e+\\&((\boldsymbol{b}_e\cdot\boldsymbol{\tau}^{\mathbf{r}}_e + \boldsymbol{g}_e\cdot\boldsymbol{\tau}^{\mathbf{i}}_e)\cdot(V^{\boldsymbol{r}}_{t,i}\cdot V^{\boldsymbol{r}}_{t,j} + V^{\boldsymbol{i}}_{t,i}\cdot V^{\boldsymbol{i}}_{t,j})+\\&(\boldsymbol{g}_e\cdot\boldsymbol{\tau}^{\mathbf{r}}_e - \boldsymbol{b}_e\cdot\boldsymbol{\tau}^{\mathbf{i}}_e)\cdot(V^{\boldsymbol{r}}_{t,i}\cdot V^{\boldsymbol{i}}_{t,j} - V^{\boldsymbol{i}}_{t,i}\cdot V^{\boldsymbol{r}}_{t,j}))/\boldsymbol{\tau}^2_e,\\&\forall t,(e,i,j)\in E\end{aligned}\tag{A.7}$$

$$\underline{\boldsymbol{P}}^{\mathbf{g}}_k \leqslant P^{\mathrm{g}}_{t,k} \leqslant \overline{\boldsymbol{P}}^{\mathbf{g}}_k, \quad \forall t,k \tag{A.8}$$

$$\underline{\boldsymbol{Q}}^{\mathbf{g}}_k \leqslant Q^{\mathrm{g}}_{t,k} \leqslant \overline{\boldsymbol{Q}}^{\mathbf{g}}_k, \quad \forall t,k \tag{A.9}$$

$$P^2_{t,e,i,j} + Q^2_{t,e,i,j} \leqslant \overline{\boldsymbol{S}}^2_e, \quad \forall t,(e,i,j)\in E\cup E^{\mathrm{R}} \tag{A.10}$$

$$\underline{\boldsymbol{V}}^2_i \leqslant (V^{\boldsymbol{r}}_{t,i})^2 + (V^{\boldsymbol{i}}_{t,i})^2 \leqslant \overline{\boldsymbol{V}}^2_i, \quad \forall t,i \tag{A.11}$$

$$V^{\boldsymbol{r}}_{t,i} \geqslant 0, \quad V^{\boldsymbol{i}}_{t,i} = 0 \quad \forall t, i\in R \tag{A.12}$$

References

[1] IEEE Xplore. Accessed: Oct. 4 2021. [Online]. Available: ieeexplore.ieee.org/Xplore/home.jsp.

[2] J. M. Arroyo and F. D. Galiana, "On the solution of the bilevel programming formulation of the terrorist threat problem," *IEEE Trans. Power Syst.*, vol. 20, no. 2, pp. 789–797, May 2005.

[3] I. Momber, S. Wogrin and T. G. San Román, "Retail Pricing: A Bilevel Program for PEV Aggregator Decisions Using Indirect Load Control," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 464–473, Jan. 2016.

[4] C. Wang *et al.*, "Dynamic Game-Based Maintenance Scheduling of Integrated Electric and Natural Gas Grids With a Bilevel Approach," *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 4958–4971, Sept. 2018.

[5] L. P. Garces *et al.*, "A Bilevel Approach to Transmission Expansion Planning Within a Market Environment," *IEEE Trans. Power Syst.*, vol. 24, no. 3, pp. 1513–1522, Aug. 2009.

[6] E. Nasrolahpour *et al.*, "A Bilevel Model for Participation of a Storage System in Energy and Reserve Markets," *IEEE Trans. Sustainable Energy*, vol. 9, no. 2, pp. 582–598, April 2018.

[7] M. Carrion, J. M. Arroyo and A. J. Conejo, "A Bilevel Stochastic Programming Approach for Retailer Futures Market Trading," *IEEE Trans. Power Syst.*, vol. 24, no. 3, pp. 1446–1456, Aug. 2009.

[8] S. A. Gabriel, A. J. Conejo, J. D. Fuller, B. F. Hobbs and C. Ruiz, "Complementarity Modeling in Energy Markets," *Springer*, 2013.

[9] H. Pandžić, A. J. Conejo, I. Kuzle and E. Caro, "Yearly Maintenance Scheduling of Transmission Lines Within a Market Environment," *IEEE Trans. Power Syst.*, vol. 27, no. 1, pp. 407–415, Feb. 2012.

[10] K. Pandžić, K. Bruninx and H. Pandžić, "Managing Risks Faced by Strategic Battery Storage in Joint Energy-Reserve Markets," *IEEE Trans. Power Syst.*, vol. 36, no. 5, pp. 4355–4365, Sept. 2021.

[11] K. Šepetanc and H. Pandžić, "Convex Polar Second-Order Taylor Approximation of AC Power Flows: A Unit Commitment Study," *IEEE Trans. Power Syst.*, vol. 36, no. 4, pp. 3585–3594, July 2021.

[12] K. Šepetanc, H. Pandžić and T. Capuder, "Solving Bilevel AC OPF Problems by Smoothing the Complementary Conditions – Part I: Model Description and the Algorithm," *IEEE Trans. Power Syst.*, 2022.

[13] K. Šepetanc, H. Pandžić and T. Capuder, "Solving Bilevel AC OPF Problems by Smoothing the Complementary Conditions – Part II: Solution Techniques and Case Study," *IEEE Trans. Power Syst.*, 2022.

[14] J. Fortuny-Amat and B. McCarl, "A Representation and Economic Interpretation of a Two-Level Programming Problem," *J. Oper. Res. Soc.*, vol. 32, no. 9, pp. 783–792 , Sept. 1981.

[15] T. A. Edmunds and J. F. Bard, "Algorithms for nonlinear bilevel mathematical programs," *IEEE Trans. Systems, Man, Cybern.*, vol. 21, no. 1, pp. 83–89, Jan.–Feb. 1991.

[16] S. Zolfaghari and T. Akbari, "Bilevel transmission expansion planning using second-order cone programming considering wind investment," *Energy*, vol. 154, pp. 455–465 , July 2018.

[17] B. Dandurand, K. Kim and S. Leyffer, "A Bilevel Approach for Identifying the Worst Contingencies for Nonconvex Alternating Current Power Systems," *SIAM J. Optim.*, vol. 31, no. 1, pp. 702-–726, Feb. 2021.

[18] C. D. Kolstad and L. S. Lasdon, "Derivative evaluation and computational experience with large bilevel mathematical programs," *J. Optim. Theory Appl.*, vol. 65, no. 3, pp. 485-–499, June 1990.

[19] E. Aiyoshi and K. Shimizu, "A solution method for the static constrained Stackelberg problem via penalty method," *IEEE Trans. Autom. Control*, vol. 29, no. 12, pp. 1111–1114, Dec. 1984.

[20] Y. Ishizuka and E. Aiyoshi, "Double penalty method for bilevel optimization problems," *Ann. Oper. Res.*, vol. 34, no. 1, pp. 73–88, Dec. 1992.

[21] B. Colson, P. Marcotte and G. Savard, "A Trust-Region Method for Nonlinear Bilevel Programming: Algorithm and Computational Experience," *Comput. Optim. Appl.*, vol. 30, no. 3, pp. 211–227, March 2005.

[22] H. C. Bylling, S. A. Gabriel and T. K. Boomsma, "A parametric programming approach to bilevel optimisation with lower-level variables in the upper level," *J. Oper. Res. Soc.*, vol. 71, no. 5, pp. 846–865, 2020.

[23] N. P. Faísca *et al.*, "Parametric global optimisation for bilevel programming," *J. Glob. Optim.*, vol. 38, pp. 609—623, Aug. 2007.

[24] S. Avraamidou and E. N. Pistikopoulos, "A Multi-Parametric optimization approach for bilevel mixed-integer linear and quadratic programming problems," *Comput. Chem. Eng.*, vol. 125, pp. 98—113, June 2019.

[25] A. Sinha, P. Malo, A. Frantsev and K. Deb, "Finding optimal strategies in a multi-period multi-leader–follower Stackelberg game using an evolutionary algorithm," *Comput. Oper. Res.*, vol. 41, pp. 374–385, Jan. 2014.

[26] X. Li, P. Tian, and X. Min, "A Hierarchical Particle Swarm Optimization for Solving Bilevel Programming Problems," *Artificial Intelligence and Soft Computing – ICAISC 2006* (LNCS 4029), L. Rutkowski et al., Eds. Heidelberg, Germany: Springer, 2006, pp. 1169–1178.

[27] J. S. Angelo, E. Krempser and H. J. C. Barbosa, "Differential evolution for bilevel programming," *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 470–477.

[28] X. Zhu, Q. Yu and X. Wang, "A Hybrid Differential Evolution Algorithm for Solving Nonlinear Bilevel Programming with Linear Constraints," *2006 5th IEEE Int. Conf. on Cognitive Informatics*, 2006, pp. 126–131.

[29] S. R. Hejazi, A. Memariani, G. Jahanshahloo and M. M. Sepehri, "Linear bilevel programming solution by genetic algorithm," *Comput. Oper. Res.*, vol. 29, no. 13, pp. 1913–1925, Nov. 2002.

[30] A. Sinha, P. Malo and K. Deb, "Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping," *Eur. J. Oper. Res.*, vol. 257, no. 2, pp. 395–411, March 2017.

[31] A. Sinha, P. Malo and K. Deb, "Solving optimistic bilevel programs by iteratively approximating lower level optimal value function," *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 1877–1884.

[32] A. Sinha, P. Malo and K. Deb, "A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 276–295, Apr. 2018.

[33] A. Ivakhnenko and V. Lapa, *Cybernetic predicting devices*, N.Y.: CCM Information Corp., 1973.

[34] A. Ivakhnenko and V. Lapa, *Cybernetics and forecasting techniques*, N.Y.: American Elsevier, 1967.

[35] B. Stott, J. Jardim, and O. Alsac, "DC Power Flow Revisited," *IEEE Trans. Power Syst.*, vol. 24, no. 3, pp. 1290–1300, Aug. 2009.

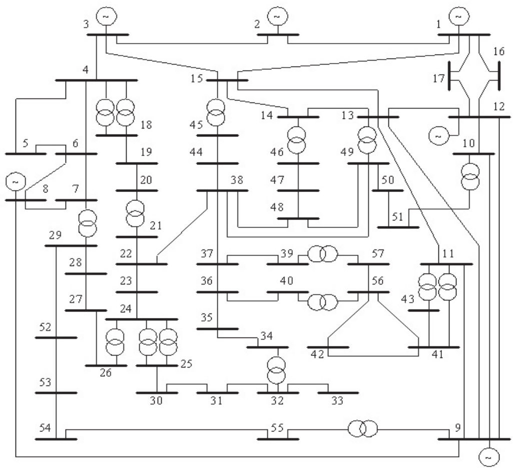
[36] H. Pandžić and I. Kuzle, "Energy storage operation in the day-ahead electricity market," *2015 12th International Conference on the European Energy Market (EEM)*, 2015, pp. 1–6.

[37] B. Dandurand, K. Kim and S. Leyffer, "A Bilevel Approach for Identifying the Worst Contingencies for Nonconvex Alternating Current Power Systems," *SIAM J. Optim.*, vol. 31, no. 1, pp. 702-–726, Feb. 2021.

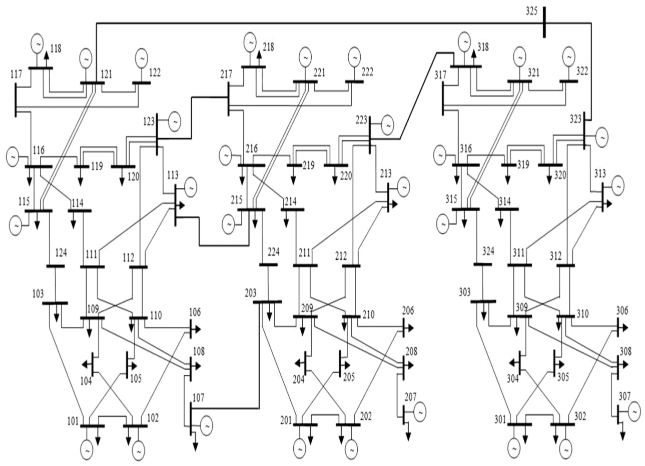
[38] S. Zolfaghari and T. Akbari, "Bilevel transmission expansion planning using second-order cone programming considering wind investment," *Energy*, vol. 154, pp. 455-–465, April 2018.

[39] S. Frank and S. Rebennack, "An introduction to optimal power flow: Theory, formulation, and examples," *IIE Trans.*, vol. 48, no. 12, pp. 1172–1197, May 2016.

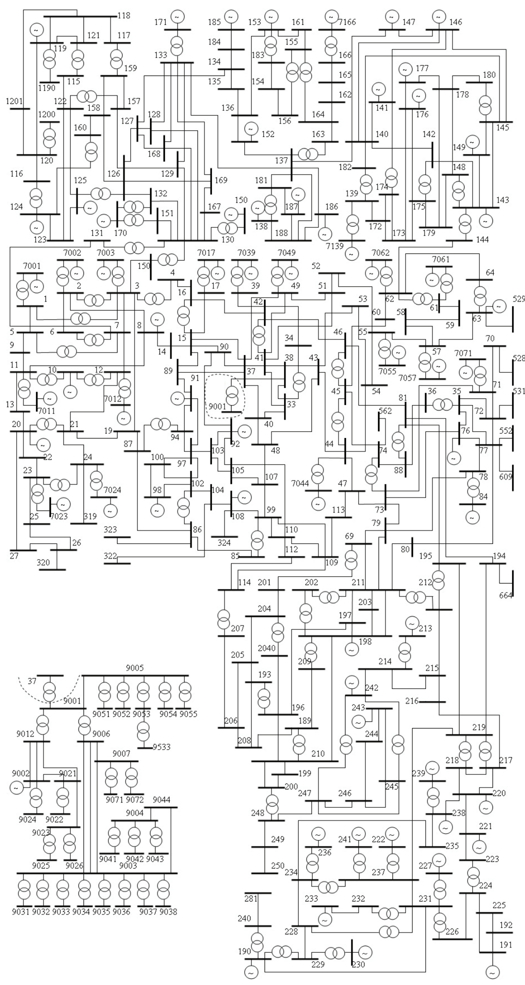
[40] R. P. O'Neill, A. Castillo and M. B. Cain, "The IV formulation and linear approximations of the ac optimal power flow problem," FERC, Washington, DC, USA, Tech. Rep., Dec. 2012. Accessed on: Jul. 10, 2021. [Online] Available at: cms.ferc.gov/sites/default/files/2020-05/acopf-2-iv-linearization.pdf



(a) 57 bus system [61]



(b) 73 bus system [60]



(c) 300 bus system [61]

Fig. 11. Network topologies

[41] K. Hornik, M. Stinchcombe and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Mar. 1989.

[42] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[43] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980.

[44] D. Ciregan, U. Meier and J. Schmidhuber, “Multi-column deep neural networks for image classification,” *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[45] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[46] D. Rumelhart, G. Hinton and R. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533-536, Oct. 1986.

[47] R. B. Rall, *Automatic Differentiation: Techniques and Applications*, N.Y: Springer-Verlag, 1981.

[48] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” *Advances in Neural Information Processing Systems 32*, 2019, pp. 1–12.

[49] J. Howard *et al. fastai*, GitHub, 2018. Accessed on: Aug. 17, 2021. [Online] Available at: github.com/fastai/fastai

[50] T. Ivek and D. Vlah, “BlackBox: Generalizable reconstruction of extremal values from incomplete spatio-temporal data,” *Extremes*, vol. 24, no. 1, pp. 145-–162, Oct. 2020.

[51] L. Wright *et al. Ranger Deep Learning Optimizer*, GitHub, 2019. Accessed on: Aug. 17, 2021. [Online] Available at: https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer

[52] L. Liu *et al.*, “On the Variance of the Adaptive Learning Rate and Beyond,” *2020 International Conference on Learning Representations*, 2020.

[53] M. R. Zhang, J. Lucas, G. Hinton and J. Ba, “Lookahead Optimizer: k steps forward, 1 step back,” *Neural Information Processing Systems 2019*, 2019.

[54] L. N. Smith, “A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay,” *arXiv*, 2018.

[55] C. Coffrin *et al. PGLib-OPF v20.07*, GitHub, Jul. 29, 2020. Accessed on: Sept. 15, 2021. [Online] Available at: github.com/power-grid-lib/pglib-opf/tree/v20.07

[56] C. Grigg *et al.*, “The IEEE Reliability Test System-1996,” *IEEE Trans. Power Syst.*, vol. 14, no. 3, pp. 1010–1020, Aug. 1999.

[57] K. Pandžić, H. Pandžić and I. Kuzle, “Coordination of Regulated and Merchant Energy Storage Investments,” *IEEE Trans. Sustainable Energy*, vol. 9, no. 3, pp. 1244–1254, June 2022.

[58] S. Wogrin, S. Pineda and D. A. Tejada-Arango, “Applications of Bilevel Optimization in Energy and Electricity Markets,“ in *Bilevel Optimization: Advances and next challenges*, Cham, Switzerland: Springer, 2020.

[59] Alberta Electric System Operator, “Market and System Reporting,“ Web page. [Online] Available at: www.aeso.ca/market/market-and-system-reporting

[60] R. A. González-Fernández *et al.*, “Composite Systems Reliability Evaluation Based on Monte Carlo Simulation and Cross-Entropy Methods,“ *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4598–4606, Nov. 2013.

[61] Al-Roomi, “Power Systems and evolutionary algorithms - load flow,“ Web page [Online] Available at: https://al-roomi.org/power-flow/