AMERICAN UNIVERSITY OF BEIRUT

Project 49 - Spring:
Optimal Power Flow via Machine Learning

by

Mohamad F. El Hajj Chehade, Intelligent Power Systems
Mohamad Al Tawil, Power & Energy Systems
Karim Khalife, Power & Energy Systems

Advisor: Prof. Rabih Jabr

A final year project
submitted in partial fulfillment of the requirements
for the degree of Bachelor of Engineering
to the Department of Electrical and Computer Engineering
of the Maroun Semaan Faculty of Engineering and Architecture (MSFEA)
at the American University of Beirut
Beirut, Lebanon

Acknowledgments

The successful completion of this project would not have been possible without the invaluable guidance and constructive feedback provided by our esteemed advisor and professor, Dr. Rabih Jabr. His unwavering support and wisdom served as a beacon of hope, especially during moments of adversity. We extend our sincere gratitude to Dr. Jabr, whose mentorship has been instrumental to our success.

We also express our appreciation to our report grader, Dr. Riad Chedid, and committee member, Dr. Farid Chaaban, for their valuable feedback and insights throughout this project.

Executive Summary

The optimal power flow (OPF) is one of the most essential problems in the operation of a power system. This problem is used to determine the minimal cost of operating a power network while abiding by the rules of energy balance, line flows, and generator limits. Despite being formulated more than half a decade ago, the problem still lacks a fast solution that can solve it in real-time and produce near-optimal results. The need for a new, more rapid, and accurate solution is even more crucial nowadays with the integration of renewable energy sources.

Machine learning has emerged in the past decade as a very powerful subset of artificial intelligence. Deep learning is a field in machine learning, where the machine learns a certain function in a structure called a neural network (NN). The NN constitutes several layers that learn input-output relationships with a high level of abstraction.

In this project, deep learning is applied to the problem of optimal power flow, with the goal of reaching faster solutions. Although neural networks require large amounts of historical data to be trained on, a special scheme has been designed to generate the required dataset using a traditional solver.

The project first introduces the OPF problem and describes the relevant work done in solving it. Then, the proposed NN solution is explored, along with the technical requirements of the project, constraints, and applicable standards. After that, the implementation of the solution for a variation of the problem, the DC-OPF, is thoroughly discussed and includes the following points:

1. Generating the data set through a developed program
2. Pre-processing the data
3. Constructing, training and testing the neural network
4. Training the neural network
5. Testing the neural network
6. Developing a user-friendly platform to solve the OPF problem

The neural network has been tested with a wide range of power networks of different sizes. The results show a significant speed-up in the time needed to solve the OPF problem while maintaining a high level of accuracy and abiding by the feasibility limits of the problem. Special schemes have also been discussed for a theoretical guarantee of a feasible solution and an improvement in the performance of the network in response to certain constraints.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The heart of a reliable and efficient Independent System Operator (ISO) is the optimal power flow (OPF) problem [1]. Since its establishment in the 1960s, OPF has been formulated in various forms as a static non-linear problem that solves optimal states of operation for certain variables in the power network [2]. The most common form of the OPF problem aims to minimize the operating cost of thermal resources, such as the fuel cost, through the optimal tuning of the electrical quantities while satisfying certain equality and inequality constraints in the network [3] [4].

The overall state of the system is determined by the load bus voltages and line flows, and the aim would be to control the generator's active power, or sometimes the transformer tap settings or reactive power sources, to minimize a certain cost or loss in the system [3]. The problem is generally limited by equality constraints related to power flow and inequality constraints determined by bounds on the operation of controllable variables such as voltages in generators [5].

Due to the heavy role that OPF plays in the determination of the cost of operation of a power system, it has gained the interest of many utilities and has been established as one of the most necessary operational demands [3] [6].

1.1 Project Needs and Motivation

Even though OPF has benefited the most from the advances in numerical computing and computer evolution [2, 7], current software tools still use lots of approximations and human intervention to address some real-time issues, which alter market prices and lead to enormous economic losses. An estimate of the losses due to the currently used software is in the order of tens of billions of dollars per year [8]. Moreover, an increase of just 5% in the optimality of the solvers would lead to a saving of about $12 billion in the United States and $87 billion

worldwide [1].

The demand for faster, more accurate solvers for OPF becomes more critical with the current developments in the power grid. Grids are welcoming more distributed energy resources (DERs) such as solar and wind energy systems and electric vehicle charging stations. These sources introduce stochasticity and rapid fluctuations to the network and require rapid responses in real-time from the network control centers [9].

As a result, traditional iterative OPF solvers must be replaced by new solvers that are faster and more robust. The new solvers must be capable of handling a new instance of the OPF problem in a fraction of a second and achieve a certain degree of optimality.

1.2 Requirements Specifications

In alignment with the desired needs of the designed OPF solver, optimal results, and fast computations, the fully trained deep learning model must satisfy the following requirements:

- The trained model should be able to solve the OPF problem in less than 1 second.
- The neural network model should produce should show a speed-up of at least 10 times from the traditional solver.
- The model must not violate any of the constraints of the problem.
- The increase in the total objective cost function from the traditional lengthy OPF solvers must not exceed 0.1%.

1.3 Deliverables

OPF is solved in energy control centers via special software programs [1]. The aim of this work is to design a similar program in a common programming language that would solve the OPF problem. At the heart of the program would be the trained model that should handle new instances of the OPF problem in real time. The associated deliverables to this project are thus:

- Software implementation of a user-friendly OPF solver based on machine learning. A user-friendly platform is developed that solves the OPF problem using a neural network and does not require any knowledge from the user in programming or machine learning.

- Validation of the OPF performance under different operational scenarios. The performance of the agent will be verified for different power network systems and will be compared to that of traditional solvers in terms of optimality and convergence time.
- Confirmation of the designated IEEE standards.

1.4 Related Work

Several techniques to solving the OPF problem have been proposed in recent decades. These methods could be grouped into deterministic, probabilistic and learning-based.

1.4.1 Determinstic Methods

Deterministic methods require exact values for load demand, energy generation and power network states. Mathematical methods are proposed in [10] and swarm intelligence methods are found in [11] [12]. The primal-dual interior-point approach is used in [13]. The convex relaxation approach is described in [14] for addressing the AC OPF problem for radial networks using second-order cone programming (SOCP). Semi-definite programming (SDP) for meshed networks is presented in [15]. Nonlinear programming [16], quadratic programming [17], linear programming [18], Newton-based approaches [19] and sequential unconstrained reduction methodology [20] are also examined.

The techniques mentioned above are unstable in terms of convergence, particularly in large-scale systems, due to the non-linearities in the problem. Hence, their applicability are restricted in real-time operation. Furthermore, the performance of intelligence-based methods is drastically limited by the uncertainty in load demand and the intermittency of renewable energy sources [21].

Nonlinear programming-based procedures, in general, have several downsides, such as unstable convergence qualities and algorithmic complexity. The piecewise quadratic cost approximation is one of the drawbacks of quadratic programming-based approaches. Newton-based approaches have convergence characteristics that are sensitive to initial conditions. As for sequential unconstrained minimization, when the penalty factors are excessively big, these approaches are known to demonstrate numerical difficulties.

Although linear programming methods are quick and accurate, their disadvantages remain related to piece-wise linear cost approximation. Although computationally efficient, if the step size is not appropriately set, the sub-linear problem may have a solution that is infeasible in the original nonlinear domain [22].

1.4.2 Probabilistic Methods

Another approach is the probabilistic, or model-based, OPF (POPF), that tries to model the uncertainty in load and generation data. Numerous approaches have been proposed to cope with the uncertainty of the distribution network. Robust optimization and stochastic programming for distribution network (DN) optimization have been suggested in [23, 24] . POPF has been also tackled in the literature using meta-heuristic algorithms, such as the artificial bee colony algorithm [25], Harris Hawks optimization (HHO) [26], and grey wolf optimization (GWO) [27]. The stochastic programming-based approaches assume information of the distribution of uncertain variables, which is used to build scenarios of uncertainty realizations.

Unlike stochastic programming, robust optimization methods deal with uncertainties by establishing an uncertainty set and searching for solutions that are resilient to all realizations inside the set. In [28,29], robust optimization-based approaches for DN management are suggested. [28] presents defines the uncertainty set using a convex hull tool. [29] presents a robust quadratic method for smart DN operation. Chance-constrained [30] and model predictive control (MPC) [31] approaches are also applied.

The stochastic models tend to have more realistic models of the power network, capturing the random variations in load demand, as opposed to the deterministic approach. However, since a huge number of situations must be examined, these approaches have high computational load. Furthermore, in reality, determining the probability distribution of unknown variables is challenging [32].

Similarly, the MPC algorithm's success is determined by the accuracy with which renewable energy generation and load demand are predicted. Moreover, when a new circumstance is encountered, the aforementioned approaches must partially or entirely address a stochastic nonlinear problem, which may be a lengthy process [33]. As a result, these strategies may be inapplicable to real-time control challenges.

1.4.3 Learning-based Methods

Learning-based methods use machine learning and data-driven insights to solve the OPF problem. Many types of machine learning models of various complexities have been used in OPF. This section summarizes them into reinforcement learning and deep learning methods.

Reinforcement Learning Methods

Reinforcement Learning (RL) is a strong learning model that learns optimal decisions through having an intelligent agent interact and learn in a certain environment [34]. Various reinforcement learning algorithms have been used to solve different forms of the OPF problem, such as the Proximal Policy Optimization (PPO) in [21], Deep Determinstic Policy Gradients (DDPG) [35], Twin-delayed Deterministic Policy Gradient (TD3) [36], Lagrangian-based DRL [37] and multi-agent systems (MAS) [38].

The mentioned RL algorithms suffer from a common drawback: in order to reach a final model ready for deployment, the RL model requires extensive training. RL models, especially deep models, are known to be very costly in terms of training time. Training a simple RL may require several hours or even days. The model depends on a set of hyperparameters that require precise, manual tuning. The performance of an RL algorithm is mostly determined by the selection of these hyperparameters, but no optimal, or at least efficient, hyperparameter modification mechanism exists to guide this process. Even if the training process is successful, it may require many training iterations that could last weeks or even months to reach the final model.

Deep Learning Methods

Several supervised-learning-based algorithms are present in the literature, which aim to mimic OPF solutions while greatly increasing solving speed. Supervised learning is the field of machine learning that necessitates the availability of training data and output samples. The "supervisor" dictates the machine to learn labels through input-output relationships in the training data [39]. A subset of supervised learning is deep learning, which allows the models to learn the data accurately through multiple processing layers that deal with representations of data with high levels of abstraction [40]. Deep learning in general relies on structures called deep neural networks (DNN). Neural networks (NN) have been widely devised in power systems [41].

The different forms of deep learning models have one thing in common: they all rely on supervised learning techniques to train neural networks. These techniques employ huge simulations that must be performed ahead of time to find approximate values of the optimal results. In order to solve this problem, several works such as [42, 43] rely on generating this dataset via well-known OPF solvers rather than getting real datasets. Although such models rely on data from traditional solvers, once the neural network completes the training phase, it will be able to replicate the results, and with proper architecture, improve on them in terms of optimality and in a much-reduced duration.

Training a neural network relies on solving an optimization problem that minimizes a certain loss function such as a mean-squared error. However, due to the ambiguous nature of the operation that imitates a "black box", a minimal loss objective may be achieved but does not guarantee a practical solution to the OPF problem, which satisfies the constraints of the problem. In order to address that issue, [44] includes a penalty term for any constraint violation in their loss function. The penalty terms may work in the case of inequality constraints, but they fail in equality constraints that address the power balance in the system. For that reason, [44] uses a NN framework that only predicts a set of the control variables. The other constraints, related to power balance, are then deduced from solving the power flow problem.

Consequently, due to the speed-up that learning-based methods offer, such techniques are explored in this work to solve the OPF problem. With regards to the deep learning method, a special program, similar to the one used in [42], is used to generate the needed data. Furthermore, as advised in [44], to avoid the inconvenience of inequality constraints, only control variables, i.e. power generation values, are predicted.

Chapter 2

Design Process

This section begins by stating the technical and non-technical limitations imposed by the problem. Various design alternatives are then examined, leading to a design decision. The multiple design iterations are subsequently explored, and the relevant standards are briefly discussed.

2.1 Engineering Constraints

2.1.1 Technical

1. **Voltage Constraint**: The power network mandates that the voltage at each node remain within specific limits, defined by a minimum and maximum value. The upper bounds are determined by the capabilities of circuit breakers, whereas the lower bounds are dictated by the operating criteria of motors or generators.

2. **Thermal Constraint**: Thermal limits apply to transmission lines and restrict the maximum current that can flow through them. The sag region may overheat and affect nearby equipment if the current is allowed to increase beyond these limits.

3. **Voltage Angle Constraint**: The power flow over an AC transmission line is directly proportional to the sine of the voltage angle difference between the receiving and transmitting ends. To ensure system stability, voltage angle differences must be maintained within specific limits. Exceeding these limits may result in the loss of synchrony between synchronous machines at either end and could potentially cause outages.

4. **Power Constraint**: For each generator, power constraints encompass both the output active and reactive power, with limits on both the upper and lower bounds depending on the specific generator utilized. It is imperative

Figure 2.1: The operation of a traditional programmed machine.

that these limits are not exceeded. Similarly, transmission lines also have an upper bound for both real and reactive power.

5. **Performance Constraint**: The real-time usage of the program necessitates that the trained model solve the OPF problem within a time frame of one second or less.

2.1.2 Non-Technical

1. **Economic**: An OPF solution must achieve the most economical operation of generation units, with the primary objective of minimizing the associated operating costs.

2. **Environmental**: The OPF solution must enable the seamless integration of renewable energy resources, which should be accurately modeled as negative loads.

3. **Sustainability**: In accordance with the sustainable development goals, the OPF should prioritize the integration of affordable clean energy to promote sustainable development.

2.2 Design Alternatives

2.2.1 Supervised Machine Learning

Supervised machine learning is a technique in which the machine learns from past experience to take decisions on future events. Suppose the problem is defined as shown in figure 2.1. The general aim of a computer or model is to produce the output based on a certain input and a certain function that governs the input-output relationship.

As in the case of a traditional computer program, the computer is fed with a certain function f and an input vector $\mathbf{X}$. In this case, depicted in Fig. 2.2, f is well-defined. For example, f could be a law of physics such as the rule for the elastic potential energy in a spring, where the input is the compression/elongation of the spring. The relationship is well-established and is illustrated below:

$$f : x \mapsto E \tag{2.1}$$

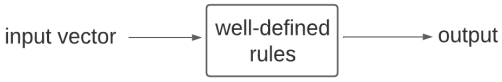


Figure 2.2: The operation of a traditional programmed machine.

Figure 2.3: A supervised learning (SL) model.

$$E = f(x) = \frac{1}{2}kx^2 \tag{2.2}$$

where E is the elastic potential energy, x is the spring compression/elongation and k is a constant. In this case, f is well-known.

More generally defined, given a certain function f and an input vector $\mathbf{X}$, the computer program is able to determine an estimate $\hat{\mathbf{y}}$ of the output as follows:

$$\hat{\mathbf{y}} = f(\mathbf{X}) \tag{2.3}$$

In this case, both $\mathbf{X}$ and f are given and are used to estimate $\hat{\mathbf{y}}$. The process involves solving the above equation and requires a single iteration only.

In some cases, f may not be fully-known, or even if known, may be very expensive to compute. For example, f could involve a set of equations that include expensive operations such as matrix inversions, dynamic programming, differential equations, etc. In these cases, the traditional computer program may fail to find an accurate estimate $\hat{\mathbf{y}}$ in a given time frame. Therefore, other techniques such as supervised learning may be used.

Instead of being supplied with the function f, the supervised learning machine tries to learn it from experience. More precisely, the machine is fed with the input vector $\mathbf{X}$ and the corresponding real output vector $\mathbf{y}$. This output vector could be obtained through experimentation or through traditional solvers that have been previously used to solve for $\mathbf{y}$. This set of historical data is compiled in a dataset containing a certain number N of instances, where for each $\mathbf{X}$, there exists a $\mathbf{y}$. The aim of the supervised machine learning process is to infer the relationship between the input $\mathbf{X}$ and output $\mathbf{y}$. This is illustrated in figure 2.3.

The aim of the training process is to approximate the function $\hat{f}$ that relates $\mathbf{y}$ to $\mathbf{X}$. The supervised learning model precisely learns a set of parameters, and

uses them to form an estimate $\hat{\mathbf{y}}$ of the real $\mathbf{y}$ from the vector $\mathbf{X}$. This is depicted below:

$$\hat{\mathbf{y}} = \hat{f}(w, b|\mathbf{X}) \tag{2.4}$$

where $\hat{\mathbf{y}}$ is the estimate of the real output $\mathbf{y}$, $\mathbf{X}$ is the input vector, $\hat{f}$ is the estimate of the function relating the input to the output, and w and b are known as the weights and biases and are the parameters that define the supervised learning model.

In order to reach a good-enough estimate $\hat{f}$, the model enters a training process formed of several iterations, where in each iteration, w and b are adjusted in a way to minimize the error of the prediction. In other words, the training process involves an optimization problem, where optimal values for w and b are obtained from minimizing the following objective function:

$$min_{w,b}\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) \tag{2.5}$$

where w and b are the parameters of the machine learning model, $\mathbf{y}$ is the real value of the output, which is found in the dataset and obtained through experimentation or traditional solvers and $\hat{y}$ is the estimated value for the output given an input $\mathbf{X}$ shown in equation 2.4. The loss function $\mathcal{L}$ depends on the nature of the problem and output value. Examples of error functions include the mean-squared error, mean-absolute error and binary cross-entropy error. The choice of error is strongly related to the type of output variable $\mathbf{y}$. In this discussion, two branches of supervised learning emerge: '

Classification:

In this case, the output variable may only take discrete values. These values may be binary, such as in the case of predicting whether a person has breast cancer (output) based on an input vector of pixels forming an X-Ray image (input). It can also be formed of a larger set of discrete values, like in the case of predicting a person's emotions (anger, happiness, nervousness, etc) from his/her speech (input). Many classification problems use a binary cross-entropy loss function to optimize the parameters of the model. This work does not focus on classification problems.

Regression:

In regression problems, the output variable can take continuous values. For example, the output variable could be the price of a house, whereas the input vector may include the number of rooms, the location of the house, etc. Linear regression is a very common, yet simple, family of regression problems. Most regression

Table 2.1: Different types of supervised learning.

Problem	Output Variable	Example
Classification	Discrete	Breast Cancer Diagnosis
Regression	Continuous	House Price Estimation

problems rely on minimizing the mean-squared error of the estimated output. In this project, the OPF problem is converted into a regression problem, where the output target variable includes a continuous value for control variables such as the active and reactive power in the generating units.

Hence, the nature of the problem, whether a classification or regression, is strictly based on the type of the output variable, whether discrete or continuous. The difference between the two methods is better illustrated in table 2.1.

In order to have a clearer picture of the training process, if we take the example of the house price, suppose that a customer would like to estimate the price based on the number of rooms and location. A supervised learning model is constructed and uses a dataset that contains the number of rooms in the house and the house location as input variables, and the price of the house as output variable. This dataset can be acquired from real-life examples.

The supervised learning model takes these input values, and through optimally tuning its set of parameters, finds an estimate of the output variable. As long as the error in estimation value is larger than a certain stopping criteria, the parameters continue to be updated based on a well-established update scheme.

Once the stopping criterion is met, the model finishes training and enters the testing phase, where its performance is evaluated on a set of new data points that it has never seen before. The metric based on which it is tested could be the mean-squared error, percentage error, etc. At this stage, the parameters of the model can no longer be updated, as in the training stage. If this testing process yields satisfactory results, the model can be deployed and used to estimate prices of houses using a new instance of input data, shown in figure 2.4.

In a nutshell, supervised learning involves three different stages:

1. Training stage
2. Testing stage
3. Deployment stage

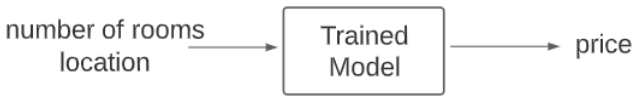


Figure 2.4: The trained model is used on a new instance of input data.

Figure 2.5: A NN-based OPF solver.

Deep Models

Deep supervised learning models typically utilize neural networks, which consist of multiple layers. The input layer receives a set of inputs, and the output layer returns a vector of outputs. During training, the neural network is optimized by minimizing the mean squared error between predicted and actual output values. The network's parameters, namely its weights and biases, are updated at each iteration of the training process using gradient descent or a variant thereof.

For a NN realization of the thermal resource dispatch version of the OPF problem, the input to the NN may be the current state of the power network that is represented by the active power P_D and reactive power Q_D demands at the load nodes. The input network is then forward propagated across the layers of the NN until the output vector is found in the final layer. In this case, the output vector includes the control variables of the OPF problem, namely the active power P_G and Q_G produced by the generating nodes. This is depicted in figure 2.5. After the control variables are found from the NN, the associated state variables such as the voltages V and phase angles θ are deduced from solving the load flow equations. In this manner, the NN does not have to deal with equality constraints, such as those related to power balance, which may cause problems for the NN.

2.2.2 Deep Reinforcement Learning

In this section, an overview of reinforcement learning (RL) is first presented. After that, the two main RL alternatives, deep Q-learning and actor-critic methods, are touched.

Markov Decision Process Modeling

The agent in an RL model takes actions that are governed by a Markov Decision Process (MDP). An MDP is a mathematical formulation of a sequential deci-

sion making process [34]. The agent, or computer, learns the solution through solving an MDP. A very noticeable feature in MDPs is that current states in the process are independent of previous states. The MDP has four main components:

State $s_t \in S$: the observables that the agent encounters in the environment. In our case, the states are P_{D_i} and Q_{D_i} $\forall i \in N$, the active and reactive power demand at each bus. These values are randomly assigned at the beginning of every training episode and are left constant throughout.

Action $a_t \in A$: the action that the agent can take when in state S. It represents the control variable that the agent uses to maximize the reward at each time step. In our case, the actions the agent can take are P_{G_i} and Q_{G_i} $\forall i \in G$, the respective active and reactive power generated at each generating node.

State Transition Probability $P(s_{t+1} = s' | s_t = s, a_t = a)$: probability of the agent reaching state s' when taking the action a at state s. Whenever the agent decides to take an action a that would lead it to a new state s', due to the stochasticity in the environment, there exists a certain probability that the agent may reach another state using a different action. This stochasticity in real-time operation is accounted for by errors in load forecasts and is reflected in our model in the randomness of the assigned values for power demand.

Reward r_t: the information that the agent receives for every state-action transition, and which it tries to maximize. A proper representation for the reward in our case would be to appreciate more the lower the cost of the objective function while punishing the agent by a penalty value for every constraint violation. Hence, it can be formulated as follows:

$$r_t = \begin{cases} -5000 & PF \quad solver \quad diverges \\ R_{br,v} & LF \quad constraint \quad violation \\ wC_{generation} + z & normal \quad operation \end{cases} \tag{2.6}$$

where a large penalty is imposed when the power flow solver fails due to certain actions taken by the agent, and penalty $R_{br,v}$ is used when the branch limits are violated and is equal to the absolute difference of the violation. When the system is operating normally and no constraints are violated, the lower the cost of generation, the more reward the agent must receive. As a result, the convex objective function $C_{generation}$ is made concave by applying a linear transformation, where w is a small negative number and z is a positive number, as established in [45].

Associated with the reward is the discount factor γ that ensures that the reward decreases with time, encouraging the agent to reach the optimal policy as

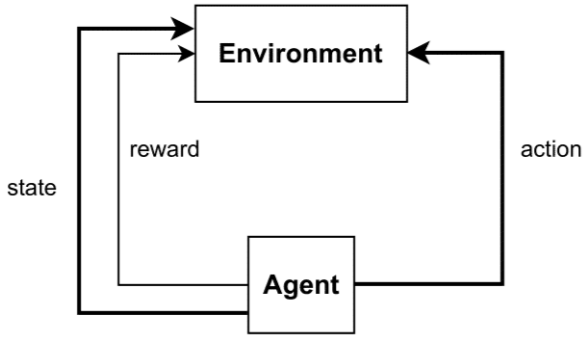


Figure 2.6: MDP components in RL

soon as possible. γ is generally a positive number ≤ 1 and decays with time. For simplicity, γ is assumed to be 1.

The relationships between these four components is illustrated in Fig. 2.6. The agent is set in a power network environment that runs the power flow solver and sends the agent active and reactive power data (states) and reacts by adjusting the power generated by each generator (action). The actions it takes are shaped by the resulting total cost they achieve and the amount of constraints they violate (reward).

Deep Q-learning

The agent in deep Q-learning learns the optimal policy through evaluating the actions it takes while interacting with the environment. The evaluation of each action is based on the action-value function, a discretized look-up matrix of dimensions, equal to the possible states in the environment and actions that the agent may take [46]. Although deep Q-learning has enjoyed success in applications related to the smart grid, its use in OPF is very limited due to the failure of the algorithm in continuous-action spaces. Therefore, this design alternative shall not be implemented.

Actor-Critic Models

Actor-crtic models combine actor, or policy-based, and critic, or value-based, models in one agent.

Actor Network: this network maps an action a to a state s. Since the actor network is a policy-based model, the transition from a state s to s' is dictated by a policy function $\pi_\theta(a|s)$ where θ is a vector of special parameters for the policy function. $\pi_\theta \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$, where μ_θ is obtained from the output of the actor NN, and Σ_θ is a diagonal matrix updated via back-propagation. As a result,

when the agent is at state s, each action taken is assigned a certain probability value $\pi(s, a|\theta)$, and the agent may choose to follow the action with the highest probability. The training of the actor network is based on the gradient ascent update of the parameters θ and makes use of the expected reward values provided by the critic. The update scheme for the actor network is as follows:

$$\theta \leftarrow \theta + \alpha_\theta Q_w(s, w) \nabla_\theta log \pi_\theta(a|s) \tag{2.7}$$

where α_θ is the learning rate associated with the actor network and is a hyperparameter that must be tuned, $Q_w(s, w)$ is the expected reward of the value function estimated by the critic network, and $\nabla_\theta log \pi_\theta(a|s)$ is the normal update scheme in policy-based methods.

Critic Network: this network maps the current state s to a certain value. By this way, it assesses the quality of the current state. Critic networks use value-based methods and value functions, similar to Q-learning, to evaluate the states. The value function is also a deep network with its own parameter vector w, just as in deep Q-learning. The critic network makes use of the temporal difference function $\delta(t)$. This function is given by:

$$\delta(t) = r_t(s, a) + \gamma Q_w(s', a') - Q_w(s, a) \tag{2.8}$$

where $r_t(s, a)$ is the reward for taking action a from state s, $Q_w(s, a)$ is the expected cumulative reward starting from state s and action a, γ is the discount factor and $Q_w(s', a')$ is the expected cumulative reward one step later. In other words, the temporal difference measures the advantage of taking action a from step s as opposed to the average of taking all actions. The more the temporal difference, the more reward is taken from this particular action, and naturally, the more this action is favored. Therefore, the larger the temporal difference, the larger the update of the weights w at that time step. Hence, the critic network is updated according to the following scheme:

$$w \leftarrow w + \alpha_w \nabla_w Q_w(s, w) \tag{2.9}$$

where α_θ is the learning rate associated with the actor network and is a hyperparameter that must be tuned and $Q_w(s, w)$ is the expected reward of the value function.

Fig. 2.7 is a high-level depiction of an actor-critic RL model.

he software chosen for the design of the RL agent is the MATLAB environment for two main reasons: (1) MATLAB has a very powerful power flow solver known as MATPOWER [47]. MATPOWER is an open-source simulation package that provides solvers for the power flow and optimal power flow problems [48] .

The deep reinforcement learning model is based on an actor-critic algorithm.

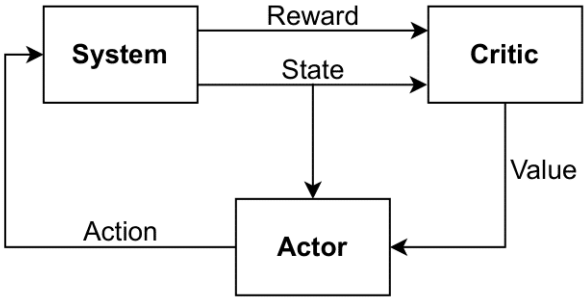


Figure 2.7: An A2C MDP.

The RL environment contains the states that the RL agent interacts with. In this case, the states are the power demands in the network The reward function is based on minimizing the cost function and contains penalty terms for any violated constraints. Each of the actor and the critic has its own NN. A couple of variations of the actor-critic algorithms have been used in this study, namely the deep deterministic policy gradient (DDPG) and the twin-delayed DDPG (TD3). These algorithms are readily available in the Reinforcement Learning toolbox of MATLAB.

The MATLAB RL model necessitates the presence of two files:

- An "Environment.m" file that defines a class object responsible for updating the states in the environment.
- A "trainAgent.m" file that designs the actor and critic NNs, selects the A2C algorithm and sets its hyperparameters, and trains the agent.

The environment is set in a power network formed of certain bus configurations. The actor-network decides on the set of power generation values $P_{G_i}\ \forall i \in G$ and sends the values to the environment, where MATPOWER performs power flow analysis and computes the cost of the system. The cost is used to find the reward and is fed into the critic network.

The environment file consists of 3 main methods:

- Environment Constructor
  1. set the observation information
  2. set the action information
  3. call the built-in function
- Step
  1. get the action performed by the agent
  2. create a new state by adopting previous demand state

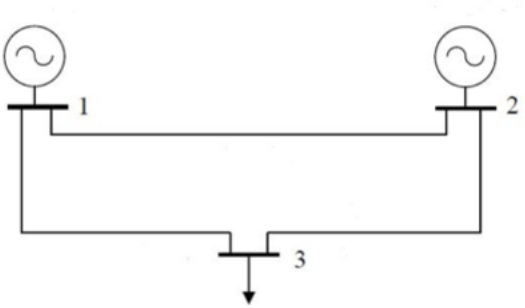


Figure 2.8: A simple 3-bus network

3. perform DC load flow on the network with the new value of states and adopted action
4. update the reward

- Reset
  1. Randomly set the states at the start of each episode

A simple 3-bus network shown in Fig 2.8 is considered at the beginning with 2 generators and 1 load. The agent has control over the power production of one generator while the other generator compensates the mismatch in power.

The training of such a simple network by an RL agent requires the manual tuning of several hyperparameters listed in table 2.2. Due to the large number of hyperparameters, finding the optimal set is a very complex procedure.

After building the RL environment and agent in MATLAB and conducting some attempts of hyperparameter tuning, the training procedure is depicted in figure

Table 2.2: RL model hyperparameters.

Parameter	related to
Number of layers	Actor NN
Number of neurons per layer	Actor NN
Activation function	Actor NN
Learning rate	Actor NN
Gradient Threshold	Actor NN
Number of layers	Critic NN
Number of neurons per layer	Critic NN
Activation function	Critic NN
Learning rate	Critic NN
Gradient Threshold	Critic NN
Sample Time	Agent Options
Batch Size	Agent Options
Number of Episodes	Training Options
Number of Steps per Episode	Training Options
Score Averaging Window Length	Training Options

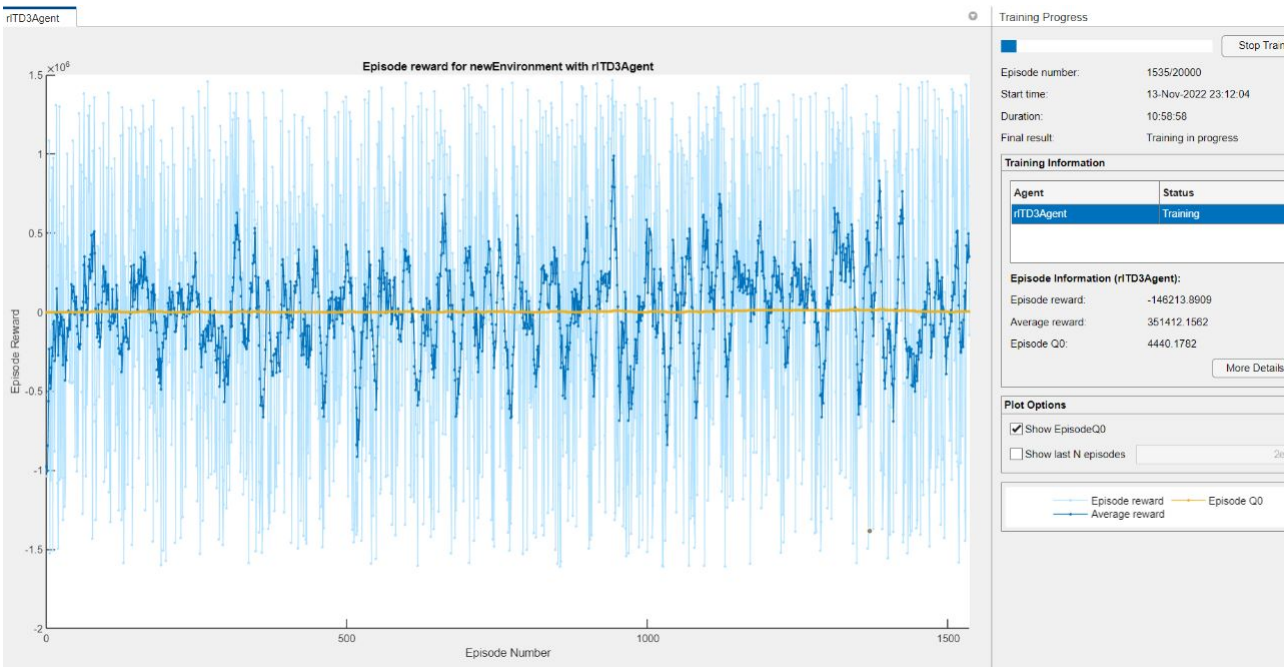


Figure 2.9: RL model training.

2.9. The training was carried out using a 16GB RAM computer with processor Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz. It can be seen that the reward is relatively stable, even after over 1,500 episodes of the training process. This means that the agent is failing to learn the optimal policy with the current set of hyperparameters. Consequently, the set of parameters requires several more training iterations that could last weeks or even months. Due to the time frame of the current project, such a process is deemed infeasible and a different design alternative is explored.

2.3 Design Decisions

The neural network model has demonstrated its capacity for accurate results and speed enhancement. Additionally, a program can be coded to generate the necessary dataset from a conventional solver like MATPOWER. On the other hand, the deep reinforcement learning model is constrained by its lengthy training process and the necessity to fine-tune hyperparameters. Consequently, the decision has been made to proceed with the neural network model to address the OPF problem.

2.4 Design Iterations

Several design iterations were executed to optimize the neural network architecture:

- **Iteration 1**: The input data was not normalized, leading to model failure and equivalent performance to that of a dummy model. A dummy model returns a predetermined value for output irrespective of input data.
- **Iteration 2**: The final activation function was selected as a ReLu function, consistent with prior iterations. However, convergence was unsuccessful 20% of the time. Consequently, the ReLu function was substituted with a linear function having no impact (output equals input) on model performance.

2.5 Standards

1. **IEEE Standard for Electric Power Systems Communications - Distributed Network Protocol** [49] relates to the use of low-cost distribution feeder devices which in our case may involve the use of cost-efficient generators and transmission lines.
2. **IEEE Standard for Exchanging Information Between Networks Implementing IEC 61850 and IEEE Std 1815** [50] relates to how the data is retrieved from the network and sent to the energy control center.
3. **Smart Grid Research: Power - IEEE Grid Vision 2050 Roadmap** [51] relates to the need to improve and develop the grid to satisfy the required energy technologies in the future.

Chapter 3

Implementation

3.1 OPF Formulation

3.1.1 AC-OPF Formulation

The OPF problem has been modeled in various forms, depending on the desired objective. In this work, the problem will focus on minimizing the total cost of operation of the thermal sources in the power network.

Objective Function:

The cost of power production is determined from the cost function. This cost function maps the amount of power generated to the associated cost. This function is derived empirically from a heat rate curve that outputs the generator power from the input thermal energy and fuel cost per thermal unit [5]. In most cases, the cost function is a quadratic polynomial [3], and the objective function can be modeled as follows:

$$min \sum_{i \in G} (c_{2i} P_{gi}^2 + c_{1i} P_{gi} + c_{0i}) \tag{3.1}$$

where G is the set of generators in the system and c_{2i},c_{1i} and c_{0i} are non-negative coefficients of the cost function.

The constraints in the OPF optimization problem could be grouped into inequality and equality constraints. The former group deals with limits imposed on various electrical quantities while the latter ensures power balance in the network.

Inequality Constraints:

These constraints are related to limits on voltage values in all nodes and active and reactive powers in generators. Upper bounds on voltage values are exerted by limits on circuit breaker capabilities while lower bounds are due to generator and motor ratings [1]. The power generation is bounded from above by the rated capacity of the generator and from below by the flame limits required to ensure fuel combustion. Line flow constraints are represented in the form of apparent powers and depict thermal limits imposed on the line [1].

$$V_{G_i}^{min} \leq V_{G_i} \leq V_{G_i}^{max} \qquad \forall i \in N \tag{3.2}$$

$$P_{G_i}^{min} \leq P_{G_i} \leq P_{G_i}^{max} \qquad \forall i \in G \tag{3.3}$$

$$Q_{G_i}^{min} \leq Q_{G_i} \leq Q_{G_i}^{max} \qquad \forall i \in G \tag{3.4}$$

$$|S_{flow_{i,l}}| \leq S_{flow_{i,l}}^{max} \qquad \forall (i,l) \in L \tag{3.5}$$

where N and L are the respective sets of nodes and branches in the power network.

Equality Constraints:

These constraints deal with power balance in the network:

$$P_{G_i} - P_{D_i} = V_i \sum_{j \in N} V_j (G_{ij} \cos \delta_{ij} + B_{ij} \sin \delta_{ij}) \qquad \forall i \in N \tag{3.6}$$

$$Q_{G_i} - Q_{D_i} = V_i \sum_{j \in N} V_j (G_{ij} \sin \delta_{ij} - B_{ij} \cos \delta_{ij}) \qquad \forall i \in N \tag{3.7}$$

where P_{G_i}, P_{D_i}, Q_{G_i} and Q_{D_i} are the real and reactive power generated and demanded in node i respectively, G_{ij} and B_{ij} are the conductance and susceptance on the line (i, j) and δ_{ij} is the voltage phase angle difference between nodes i and j.

3.1.2 DC-OPF Formulation

A linearized version of OPF known as the DC Optimal Power Flow (DC-OPF) is considered [5]. DC-OPF is a convex optimization problem that finds is usage in market clearing and transmission management [44]. The main assumptions that DC OPF takes into consideration are:

- All the node voltages are assumed to be 1 p.u.
- Phase angles are assumed to be small enough to use small angle approximations

- The active power losses in transmission lines are neglected.
- Reactive power is neglected in this study

Consequently, the optimization problem for DC-OPF can be modeled as follows:

$$min \sum_{i \in G} (c_{2i} P_{gi}^2 + c_{1i} P_{gi} + c_{0i}) \tag{3.8}$$

subject to:

$$P_{G_i}^{min} \leq P_{G_i} \leq P_{G_i}^{max} \qquad \forall i \in G \tag{3.9}$$

$$|\frac{1}{x_{i,l}}(\theta_i - \theta_l)| \leq P_{flow_{i,l}}^{max} \qquad \forall (i,l) \in L \tag{3.10}$$

$$P_{G_i} - P_{D_i} = \sum_{j=1}^{N} B_{ij} \theta_j \qquad \forall i \in N \tag{3.11}$$

where Eq. 3.10 reflects the active power line flow limits and Eq. 3.11 represents the active power balance.

3.2 Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs), or neural networks (NNs), are structures formed of several layers, each having a certain number of neurons. A simple NN is shown in Fig. 3.1. Each color represents a layer, and each circle represents a neuron. The first layer of an NN is the input layer, where a vector of input features is fed. The input then propagates through a certain number of hidden layers, the result of which is the final output layer. This layer consists of one or several neurons depending on the problem. NNs can be used for both classification and regression problems.

The propagation of data through the NN is a function of the network parameters, a set of weights $W = \{ w_1, w_2, ..., w_i, ..., w_{m-1}\}$ and biases $B = \{ b_1, b_2, ...,b_i,... , b_{m-1} \}$, where m is the number of layers in the network. Each weight w_i is a matrix of dimensions $l \times k$ and each bias b_i represents a scalar, where k is the number of neurons in the layer $i-1$ and l the number of neurons in layer i.

The value of each neuron in layer i is a linear combination of the neurons of the previous layer followed by a non-linear activation function. This function can be a hyperbolic tangent, a rectifier (ReLu), or a sigmoid. The parameters used for the linear combination are the weights and biased and are the ones optimized. This relation is known as forward propagation and is illustrated in eq. (3.12).

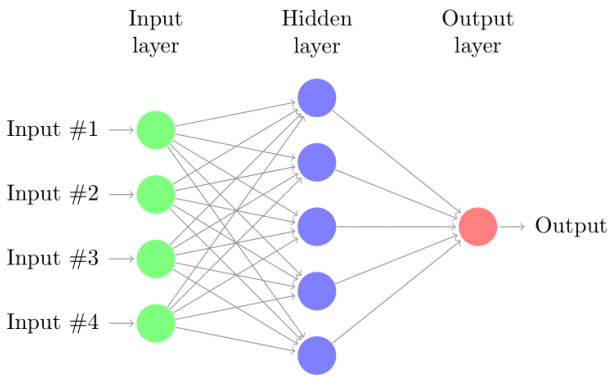


Figure 3.1: A simple neural network.

$$a_i = \sigma(w_{i-1}a_{i-1} + b_{i-1}) \tag{3.12}$$

where a_i is known as the activation vector that represents the values of the neurons at layer i, σ is the activation function, and w_{i-1} and b_{i-1} are the weights and biases associated with layer

3.2.1 Neural Network Training

The supervised learning problem can be formulated as an unconstrained optimization problem. The input values in the case of OPF are the states of the system, such as the demand values and bus voltages while the output values are those corresponding to the control variables, such as the active and reactive power in the generating units. The objective function in Eq. (3.13) is a mean-squared error (MSE) function, where the average of the squared difference between the actual and estimated values is minimized. The predicted value is a continuous output from the neural network found in its output layer. Since layers of a neural network are related to each other through Eq. (3.12), this error propagates back through the layers. As a result, the parameters of the NN, namely the weights and biases, are tuned using a reverse-propagating mechanism known as back-propagation and discussed in the next section.

$$\min_{W,B} \frac{1}{N} \sum_{i=1}^{N} (\hat{\mathbf{y}}_i - \mathbf{y}_i)^2 \tag{3.13}$$

where the $\hat{\mathbf{y}}$ and $\mathbf{y}$ are respectively the forecasted and actual values of output, n the number of samples in the dataset, and W and B are the sets of weights and biases respectively.

3.2.2 Back-propagation

As mentioned in the previous section, the error propagates back from the output to the input. The weights and biases in each layer are updated in a recursive manner through a method known as back-propagation. In our case, due to the convexity of the loss function, back-propagation may use a first-order method such as gradient descent (GD) to update the values of the NN parameters. GD aims to minimize the second-order Taylor series approximation of the loss function, which acts as an upper bound to the original function. The convergence of such a method is well-established. An alternative method would be to find the closed form solution, but computing such a solution is very expensive due to the large amount of data points.

If we let the loss function $\mathcal{L}(w_i, b_i)$ be a mean-squared error function, the gradient of $\mathcal{L}$ is defined as shown in Eq. (3.14).

$$\nabla\mathcal{L} = (\ldots, \frac{\partial\mathcal{L}}{\partial w_i}, \ldots, \frac{\partial\mathcal{L}}{\partial b_i}, \ldots) \tag{3.14}$$

The partial derivative of the loss function with respect to the weight w_i linking layers i and $i+1$ is:

$$\frac{\partial\mathcal{L}}{\partial w_i} = \frac{\partial f}{\partial\hat{y}}\frac{\partial\hat{y}}{\partial a_{m-1}}\cdots\frac{\partial a_{i+2}}{\partial a_{i+1}}\frac{\partial a_{i+1}}{\partial w_i} \tag{3.15}$$

where

$$\frac{\partial\mathcal{L}}{\partial\hat{y}} = \frac{2}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \tag{3.16}$$

$$z_m = w_{m-1}a_{m-1} + b_{m-1} \tag{3.17}$$

$$a_m = \sigma(z_m) \tag{3.18}$$

$$\frac{\partial\hat{y}}{\partial a_{m-1}} = \frac{\partial\hat{y}}{\partial z_m}\frac{\partial z_m}{\partial a_{m-1}} = \sigma'(z_m)w_{m-1} \tag{3.19}$$

$$\frac{\partial a_{i+2}}{\partial a_{i+1}} = \sigma'(z_{i+2})w_{i+1} \tag{3.20}$$

$$\frac{\partial a_{i+1}}{\partial w_i} = \frac{\partial a_{i+1}}{\partial z_{i+1}}\frac{\partial z_{i+1}}{\partial w_i} = \sigma'(z_{i+1})a_i \tag{3.21}$$

3.2.3 Iterative Descent Methods

After performing back-propagation, a gradient descent scheme is adopted as shown in Eq. (3.22). Due to the huge size of the dataset used, a mini-batch stochastic gradient descent (SGD) is used instead of the normal GD method. In each iteration of the training process, the mini-batch SGD evaluates the gradient

vector only for a small subset of the larger dataset, chosen at random, drastically reducing the training complexity. An adapted mechanism to the SGD method, known as adaptive moment estimation (ADAM) is also explored in the testing chapter. The parameter update scheme is based on choosing a direction and a constant known as the learning, which determines the size of the step in that direction. The direction is that of the negative gradient of the loss function while the learning rate is a hyperparameter that is tuned during the training process. The pseudo-code for the descent method is shown in Algorithm 1.

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla \mathcal{L} \tag{3.22}$$

where $\mathbf{w}$ is the vector of parameters and α is the learning rate. In other words, $\mathbf{w}$ is updated in the direction of the negative of the gradient $\nabla \mathcal{L}$ by a size of α.

Algorithm 1 Iterative Descent Scheme

```
for iterations do
    for batch size do
        randomly select a sample
        perform forward propagation using the sample
        perform back propagation using the sample
    end for
end for
```

3.2.4 NNs - Universal Function Approximators

A crucial advantage of NNs is that they are universal function approximators. The universal approximation theorem, shown in eq. 3.23, states that any function, linear or non-linear, can be approximated by a NN with enough neurons and layers. This enables NNs to solve complex problems that involve iterative updates more efficiently. In our work, we examine the NN performance in solving the OPF problem formulated earlier.

$$f(\mathbf{x}) \approx \sum_{j=1}^{N_L} v_j \phi(\mathbf{w_j x} + b_j) \tag{3.23}$$

where f is the function to be approximated by the neural network, $\mathbf{x}$ is the input vector, N_L is the number of layers in the network, $v_j \; in \; \mathbb{R}$ is a constant and $\mathbf{w_j}$ and b_j are respectively the weights and biases associated with layer j.

3.3 Proposed Methodology

NNs belong to the family of supervised learning, as the NN model infers the relationships between the inputs and outputs of a certain problem. Since the OPF

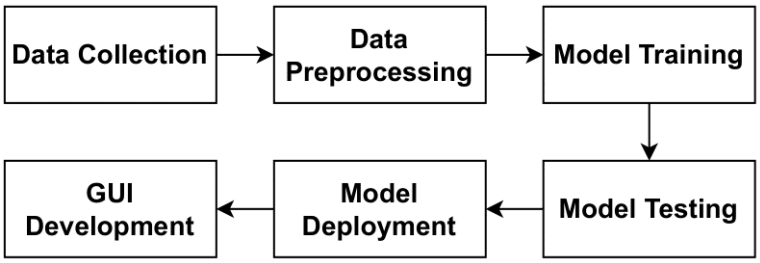


Figure 3.2: Proposed methodology.

problem involves continuous values for its state and control variables, the NN model used in this project is a regression model.

The first step of any supervised learning process is to gather the data. A program is developed to generate data from a conventional and well-known OPF solver that uses value-iteration methods. After that, input and output data are normalized before being fed to the neural network model. A neural network architecture is built for training the pre-processed data. When the training is done, the performance of the model is evaluated on a new test set using a set of performance metrics. Finally, the developed OPF solver is deployed in a proper graphical user interface (GUI). This plan is shown in figure 3.2.

3.4 Choice of Software

Since neural networks require a dataset of input-output relationships, the best way to generate this set is by solving the DC-OPF problem a certain number of times using traditional solvers. An important remark is that the traditional solvers are only used to generate the dataset that would enable the neural network model to learn. Once that is done, these solvers are no longer used and are replaced by the neural network model, which in the case of DC-OPF should show better results in terms of convergence time.

MATPOWER has been widely used by the power systems community to solve the OPF problem. One of its major advantages is the clear documentation of its different functions and data structures. As a result, MATPOWER was chosen to generate the dataset, the process which shall be explained in the next section.

After the dataset has been generated, the neural network model can be built using a certain library. Most programming languages contain special libraries or toolboxes for machine learning and deep learning applications. The aim is to find the programming language and library that offer the best computational power, interpretability of the code, and collaboration of the different members of the

Table 3.1: Choice of software tools.

Stage	Software	Library/Toolbox
Data Generation	MATLAB	MATPOWER 7.1
Data Pre-processing	Python	NumPy
Neural Network Construction	Python	Keras

group.

Most of the machine learning and data science community use Python as their preferred programming language. Python offers very useful libraries for machine learning applications such as Pytorch, TensorFlow, and Keras. The latter is chosen to build the neural network To enter data into a Keras NN model, a pre-processing stage must be done using the NumPy library of Python.

The conclusions regarding the choice of software tools are summarized in table 3.1

3.5 Data Acquisition

As mentioned in the previous parts, the dataset is generated by running the DC-OPF problem on the IEEE 9-busbar system for a certain specified number of samples. In each of these instances, the value of the active power demand at the load nodes is varied randomly within a certain range δ from its default value found in table **??**. δ is a percentage value that represents the deviation from the default value of demand. For example, the default value for the demand at node 5 is 90 MW. Consequently, in each sample in the data set, the demand at node 5 will be $P_{D_5} = x$ where x is a random uniform sample from the range $[90 - 90\delta, 90 + 90\delta]$. A typical value for δ is 10%.

An algorithm has been created in MATLAB to generate the dataset. The algorithm must receive, as input, the power network (eg: IEEE9), the number of training samples, N, needed, and the allowable variation δ. The algorithm first saves the default values for the demand nodes, which are found in table **??**. After that, for every training sample, the program generates a new demand vector $\widetilde{P_D}$ formed from varying each element of the default vector by a unique random variable generated uniformly from the range $[P_{D_i} - \delta P_{D_i}, P_D + \delta P_{D_i}]$. After receiving the new demand vector, the program runs the DC-OPF function provided by MATPOWER, which outputs the optimal values for generation $\widetilde{P_D}$ correspond-

ing to the specific case $\widetilde{P}_G$. The samples are saved in a matrix that is converted to a comma separated values ".csv" file, or another file extension depending on the user's preference. The pseudocode can be found in Algorithm 2.

Algorithm 2 Data Generation Scheme

```
Specify case file, N and δ
P_D ← default demand values
for N do
    for i ∈ demand nodes do
        P̃_{D_i} ~ U(P_{D_i} − δP_{D_i}, P_D + δP_{D_i})
    end for
    P̃_G ← run (DC-OPF | P̃_D)
    Save P̃_D, P̃_G
end for
```

3.6 Neural Network Architecture

The neural network is built in the Keras library and designed to have 1 input layer, 3 hidden layers and 1 output layer. The number of neurons in the input and output layers are equal to the size of the input and output vectors respectively. For the case of the IEEE9 network, the number of neurons is 3 in each case. The number of neurons in each of the three hidden layers is chosen by design to be 128, 64, and 32. The different stages of the neural network are found in fig. 3.3, where a dense layer is a simple feed-forward layer discussed earlier. A more thorough visualization of the network architecture is found in fig. 3.4.

The activation function is chosen to be the rectifier ReLu function, shown in fig. 3.5. The function basically clips any negative values. The ReLu activation function is widely used in regression problems, as it eliminates problems of exploding gradient. The activation function mapping the penultimate layer to the final layer is, however, a simple linear function for reasons explained in the next part of the section.

The neural network training aims to minimize a certain loss function by adjusting its parameters, namely the weights and biases, through an iterative descent scheme. Weights and biases are generally initialized randomly. A poor initialization, however, may lead in some cases to divergence. Neural network parameter initialization is an active area of research, and one of the most notable schemes developed so far is the He initialization [52]. The weights and biases are sampled from a uniform distribution described in (3.24).

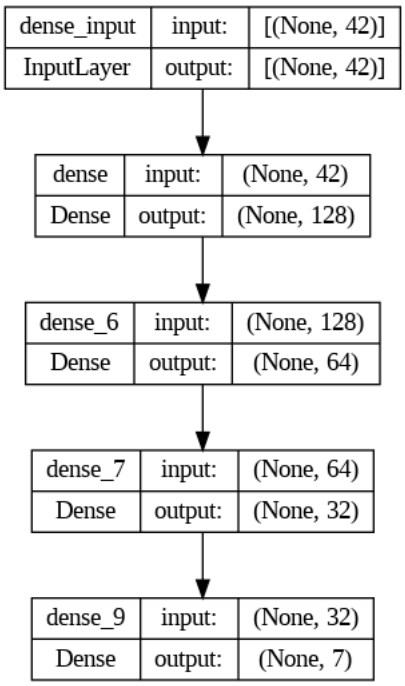


Figure 3.3: The layers of the neural network.

$$x_l \sim U(-\sqrt{\frac{6}{n_l}}, \sqrt{\frac{6}{n_l}}) \qquad \forall x_l \in \{W_l, b_l\} \qquad \forall l \in L \tag{3.24}$$

where x_l is a NN parameter, weight or bias, that relates layer l to layer $l+1$, L the set of layers, W_l and b_l the set of weights and biases of layer L and n_l the number of neurons in layer l.

3.7 Graphical User Interface

A Graphical User Interface (GUI) has been developed to aid provide an intuitive way of using the NN model.

3.7.1 Key Features:

The primary features of the platform include:

- **Training of neural network:** The GUI enables users to train the neural network on any input power network.
- **Solving of the DC-OPF problem:** The GUI can solve the Optimal Power Flow (OPF) problem using the trained network, resulting in a faster solution than that obtained using conventional solvers.

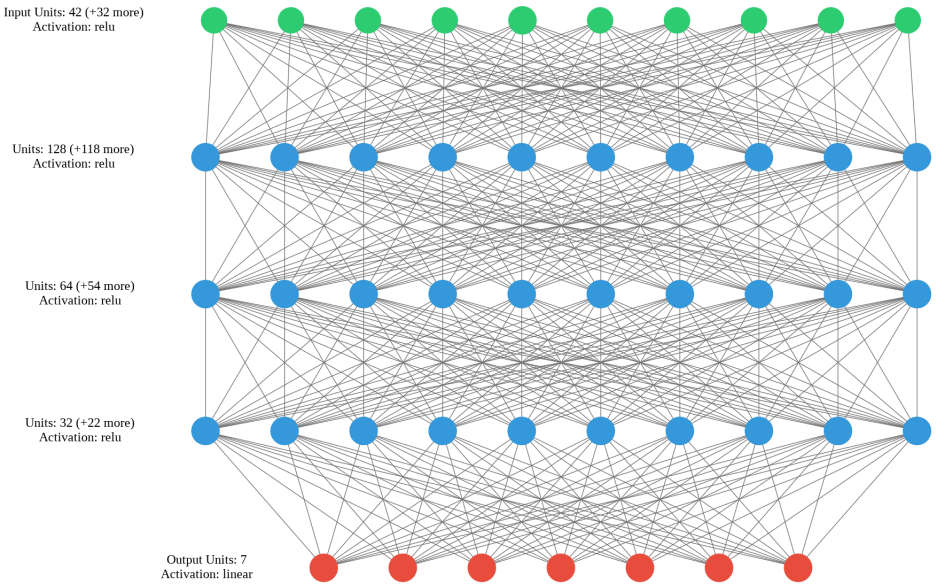


Figure 3.4: The neural network architecture.

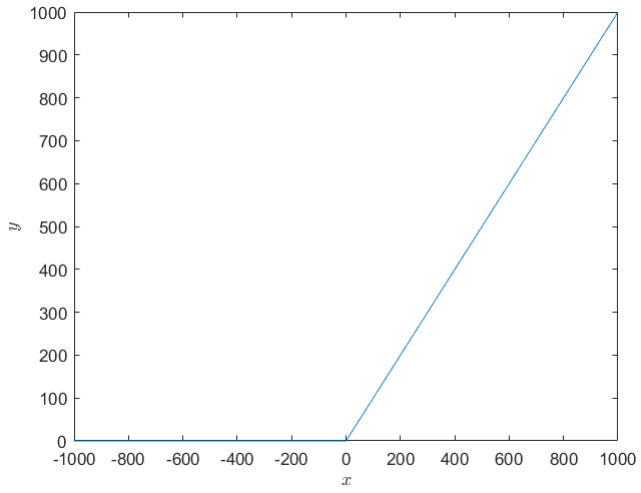


Figure 3.5: The ReLu activation function.

3.7.2 Primary Advantages:

The primary advantages of utilizing this interface are:

- **Ease of Use:** The GUI is user-friendly and does not necessitate prior knowledge of machine learning or programming. Users simply input power network information (either as a number or file) and train the neural network. Once training is complete, case files with node demand information can be entered, and the program will provide corresponding generation values.

- **Speed:** Upon completion of training, the neural network can solve the OPF problem for the designated power network multiple times with substantial speed-up compared to traditional solvers, as detailed in Table 4.3.

3.7.3 Input/Output:

Training and Testing

To commence training, users must first input the designated power network. This can be accomplished by entering the network number or a Python file (.py) that contains data about the power network in the same format as the PyPower library. The network employs this information to construct the required dataset using the methodology proposed in Algorithm 2. Following dataset generation, training occurs, and the associated mean-squared and percentage errors on the testing set are displayed.

Solving

Upon completion of training, the neural network's weights and biases are optimized for the designated power network. The NN can then be utilized numerous times to solve new instances of the OPF problem, negating the requirement for network retraining. Users must subsequently input an active power demand spreadsheet, which the neural network will utilize to produce optimal active power generation values at the generation nodes. The output will be displayed on the screen.

The GUI process is fully illustrated in Figure 3.6.

The GUI's front-end is developed with React and CSS, while its back-end is developed using Python's Flask library.

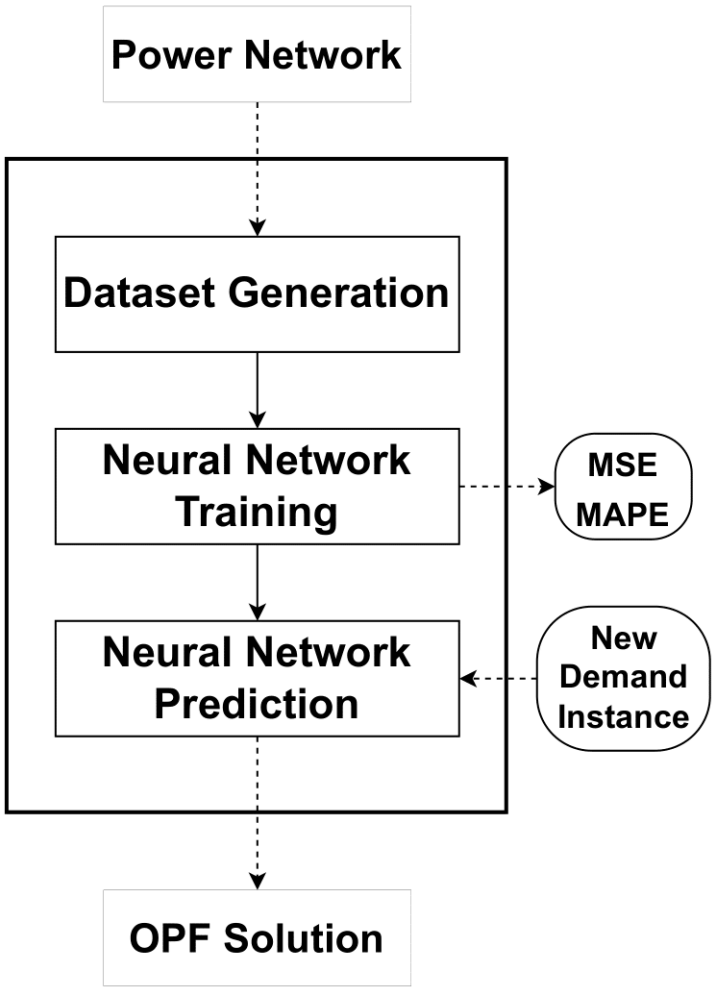


Figure 3.6: The GUI working principle.

Chapter 4

Experimental Setup and Results

4.1 Experimental Setup

Neural Network Training

This chapter evaluates the efficacy of the neural network model in addressing the DC-OPF problem on several power networks of different dimensions, namely the IEEE 30, IEEE 57, IEEE 118, and IEEE 300 systems. The neural network training process is analyzed initially, followed by an examination of performance metrics, such as the mean-squared error and percentage error, for the testing dataset. The acceleration achieved by the neural network solution compared to the conventional solver is emphasized. Lastly, a brief demonstration is presented to showcase the utilization of the solver through the developed graphical user interface.

The generated dataset will be partly used for training the neural network. The other part will only be used for testing the NN after the training has been completed. The set containing 10,000 samples is split into 80% data points for training and 20% for testing. This section will focus on the training aspect while the following section will discuss the testing.

The loss function that the neural network aims to minimize is the mean-squared error, which is the most used loss function in regression settings. The objective function, shown in eq. (4.1), tries to minimize the average squared difference between the actual value of power generation found in the dataset and the value estimated by the neural network.

$$min_{W,b} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|G|} \sum_{j=1}^{|G|} (\widehat{P_{Gij}} - P_{G_{ij}})^2 \tag{4.1}$$

where $w_{MSE} = 1$ and $w_{pen} = 5$

Table 4.1: The neural network hyperparameters.

Parameter	Value
No. iterations (Epochs)	300
Batch Size	32
Learning rate	10^{-3}

where W and b represent the weights and biases of the neural network, G is the set of generator node, N the number of training samples, $P_{G_{ij}}$ the actual optimal value for the generation at node j and sample i and $\widehat{P_{Gij}}$ the estimated value for that generation using the neural network.

To prevent overfitting, which occurs when a machine learning model performs exceptionally well on the training set but underperforms when tested, a validation set is incorporated into the training process. Overfitting occurs when the model possesses excessive variance due to overly complex model design. During training, the neural network utilizes the training set to update its parameters and evaluates the loss function for both the training and validation sets at each iteration. If the loss function of the training set is lower than that of the validation set, it typically indicates that the model is learning specific features present solely in the training set and is not capturing the general trend, thereby signifying a high variance. Thus, a validation set is crucial to ensure that the model does not reach the overfitting stage.

To achieve this goal, the training set, which constitutes 80% of the complete dataset, is divided into an 80% training set and a 20% validation set. It is essential to highlight that the validation set differs from the test set. The validation set is leveraged during the training phase to update the neural network's parameters, whereas the test set is solely utilized to assess the model's performance after the training, at a point when updating the NN parameters is no longer feasible.

In order to find the optimal parameters W and b of the NN, a gradient descent method is adopted as described in a previous chapter. More precisely, an adapted stochastic gradient descent scheme, known as ADAM, is used. The key parameters for an iterative descent scheme, as outlined in Algorithm 1 and eq. (3.22) is the number of iterations or epochs, the batch size and the constant step-size, or learning rate α. The chosen values for this neural network are shown in table 4.1.

Each power network needs distinct neural network training that learns the weights and biases specific to the neural network. As a result, the training for each of the

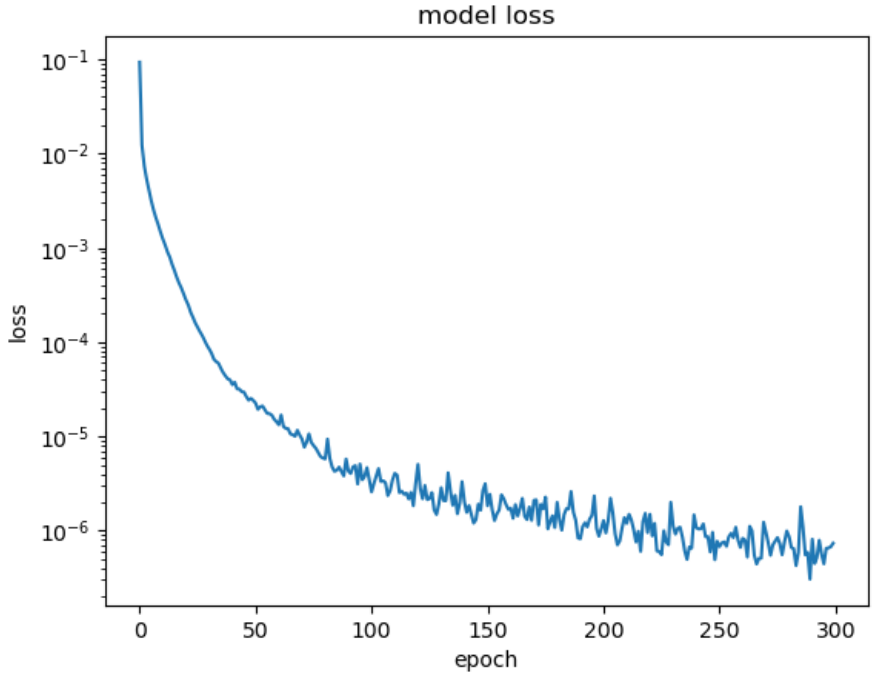


Figure 4.1: The NN training loss on a log-scale.

4 power networks under study has been done. Fig. 4.4 shows the loss function as a function of the epoch number on a log-scale plot for the case of the IEEE 57-bus system. The plot shows the successful learning of the neural network model and the convergence to a neighborhood of the order 10^{-6}. Although the results are very satisfactory, a diminishing step-size scheme may be explored to reach full convergence of the SGD.

4.2 Results

4.2.1 Neural Network Testing

After the training phase is done, the fully-trained neural network is tested with new instances present in the test set that it has not seen before. The testing set forms 30% of the original dataset. The model is only fed with the input vector containing the active power demand values at the three nodes. For each data point of the test set, the neural network will estimate the output vector. After looping over all the test points, certain evaluation criteria are computed for the NN performance:

1. **Mean-squared error (MSE)**: MSE is the metric that the neural network

Table 4.2: The performance of the neural network on various power networks.

Power Network	MSE	MAPE (%)
IEEE-30	2×10^{-8}	0.007
IEEE-57	5.2×10^{-6}	0.09
IEEE-118	4.7×10^{-8}	0.01
IEEE-300	1.5×10^{-7}	0.02

aims to minimize during training. As demonstrated in the previous section, the model achieved a satisfactory level of MSE, in the order of a millionth. This level of accuracy, if reflected in the test set results, underscores the good performance and generalizability of the model. Although there is no standard benchmark for MSE, a value significantly lower than that obtained by a dummy model (which is 0.5 in this case) is considered to be a positive result.

2. **Mean absolute percentage error (MAPE)**: MAPE computes the deviation of the results from the actual values in a more interpretable sense. A percentage error above 0.5% may not be tolerable.

Table 4.2 demonstrates the performance metrics for each power network. The comparable values of the mean-squared error (MSE) in both training and test sets indicate good accuracy of the neural network. Additionally, achieving a mean absolute percentage error (MAPE) of approximately one-hundredth of a percentage is highly satisfactory, especially considering the fact that neural networks are developed as approximators and can accomplish these outcomes in a shorter time than traditional solvers. It should be noted that the metrics vary from one network to the other due to the stochastic nature of SGD and the varying initialization of the weights and biases. Nevertheless, in all cases, a strong degree of convergence is achieved.

Table 4.3 verifies the neural network's speed-up over the traditional MATPOWER solver. This is due to the former requiring only one forward propagation iteration after being trained, while the latter needs to solve the optimal power flow problem every time. The speed-up is more evident for larger networks, as traditional solvers take longer to solve bigger networks, whereas neural networks have a nearly constant duration, regardless of network size. This is due to the unchanging neural network structure, resulting in a roughly constant number of computations. This speed-up is crucial, as it allows the energy control center to respond to fluctuations in renewable energy resources.

Table 4.3: The neural network speed-up over the traditional MATPOWER solver.

Power Network	NN (ms)	MATPOWER (ms)	Speed-up
IEEE-30	50	322	x 6
IEEE-57	50	522	x 11
IEEE-118	50	1131	x 23
IEEE-300	50	2166	x 43

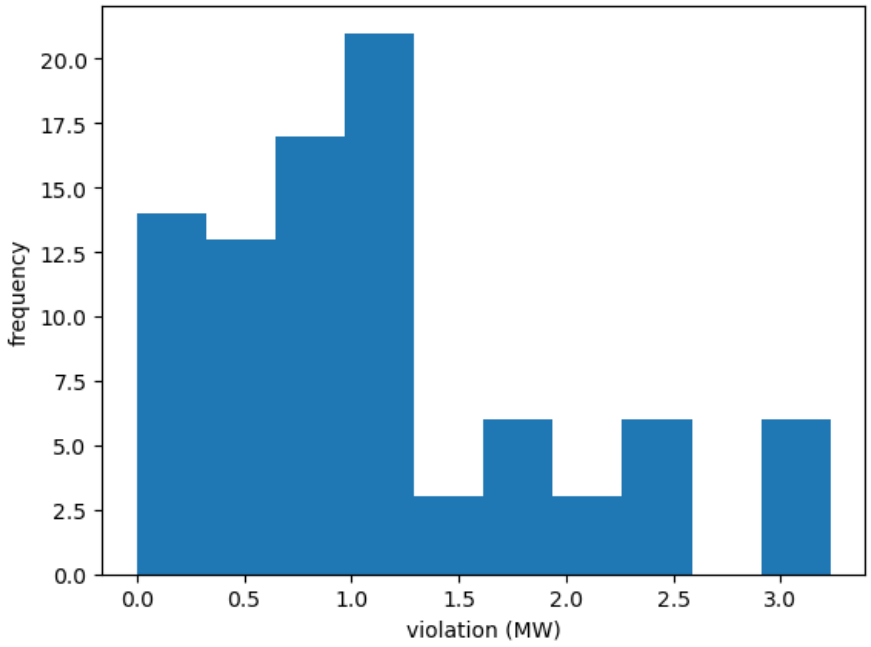


Figure 4.2: The distribution of generator limit violations for the IEEE-57.

4.2.2 Loss Function Modification

Despite the high accuracy achieved by the neural network, the lack of a theoretical guarantee for a feasible solution has resulted in occasional and minor violations of the constraints, specifically the upper and lower generation limits. Fig. 4.2 depicts the distribution of these violations for the IEEE-57 bus system, which account for 0.7% of the total cases and do not exceed 4 MW. Most of these violations are concentrated near the limit and occur when the actual solution approaches the generator limits, causing the estimated values by the neural network to slightly exceed the limits.

To overcome the issue of generator limit violations, there are two main adjustments that can be made to the neural network architecture:

- **Augmenting the neural network complexity**: One approach to mitigate the generator limit violations is to increase the neural network complexity, achieved by adding more layers or neurons per layer. However, this

Table 4.4: The performance of the neural network with an adjusted penalty term.

Metric	Score
MSE	4.7×10^{-6}
MAPE	0.1%
no. of generator limits violation	0

would result in a higher computational cost for each case, thus compromising the speed-up advantage of the neural network. Therefore, this approach is not employed.

- **Modifying the loss function:** A penalty term for violating the generator limits can be incorporated into the loss function, which was originally defined as a mean-squared error. The new loss function, as shown in equation (4.2), is a weighted sum of the mean squared error and penalty terms.

$$\mathcal{L}_{new} = w_{MSE}\mathcal{L}_{MSE} + w_{pen}\mathcal{L}_{pen} \tag{4.2}$$

where $\mathcal{L}_{MSE}$ and $\mathcal{L}_{pen}$ are the MSE and penalty loss functions, and w_{MSE} and w_{pen} are their respective weights.

Eq. (4.3) shows the penalty term in detail. As illustrated by the equation, the magnitude of the penalty term is proportional to the degree of violation.

$$\mathcal{L}_{pen} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|G|}\sum_{j=1}^{|G|} max(\hat{P}_{ij} - P_{ij}^{max}, 0) + max(P_{ij}^{min} - \hat{P}_{ij}, 0) \tag{4.3}$$

where N is the number of samples in the neural network, $|G|$ is the number of generators in the power network, P_{ij} is the estimated active power generation of the NN in sample i at node j, and P_{ij}^{min} and P_{ij}^{max} are the respective minimum and maximum active power generation bounds at node j.

The neural network is retrained using the modified loss function, where w_{MSE} and w_{pen} are optimized to the values of 1 and 5, respectively. Fig. 4.3 illustrates the loss function for the new training approach on the IEEE-57 bus, which exhibits a similar level of accuracy as the previous training methodology. The results demonstrate no violations and maintain a comparable level of precision in terms of the performance metrics, as illustrated in table 4.4.

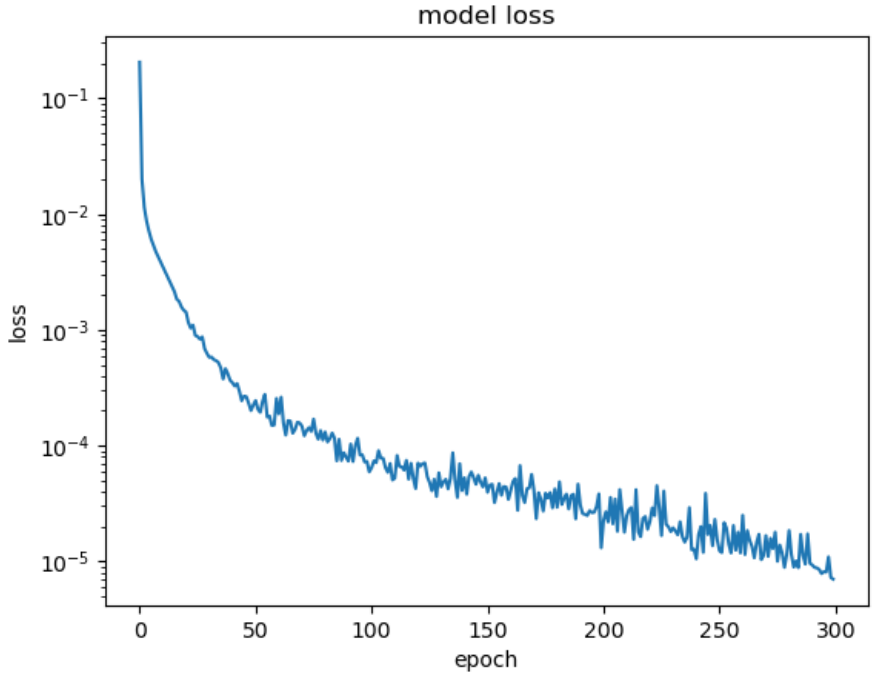


Figure 4.3: The training loss for the NN with adjusted penalty term.

4.2.3 Graphical User Interface

Figure **??** displays a snapshot of the graphical user interface (GUI). The interface enables users to enter the power network by entering the corresponding name. By default, the PyPower tool provides information on the network. However, users may also choose to provide a customized power network file in the same format as PyPower. For instance, the ieee9 power network serves as a case in point. Upon clicking "Submit," the training process is initiated, and the neural network is optimized.

After that, the user can solve the DC-OPF problem by entering a file representing the values of demand and renewable power generation, as shown in Fig. [**?**].

As depicted in Figure **??**, the neural network yields a prompt solution to the optimal power flow problem, along with corresponding performance metrics that attest to the solution's accuracy. In the event that users desire to solve a fresh instance of the same problem, they need not repeat the training process, as it is performed only when a new power network is introduced. Instead, they can conveniently reset the testing procedure and enter the new instance.

A demo explaining how to use the GUI can be found at this link: https://youtu.be/4xyF-D17Bnw. The code for the GUI can be found here: bit.ly/3pb3FAo

Figure 4.4: A snapshot of the GUI.

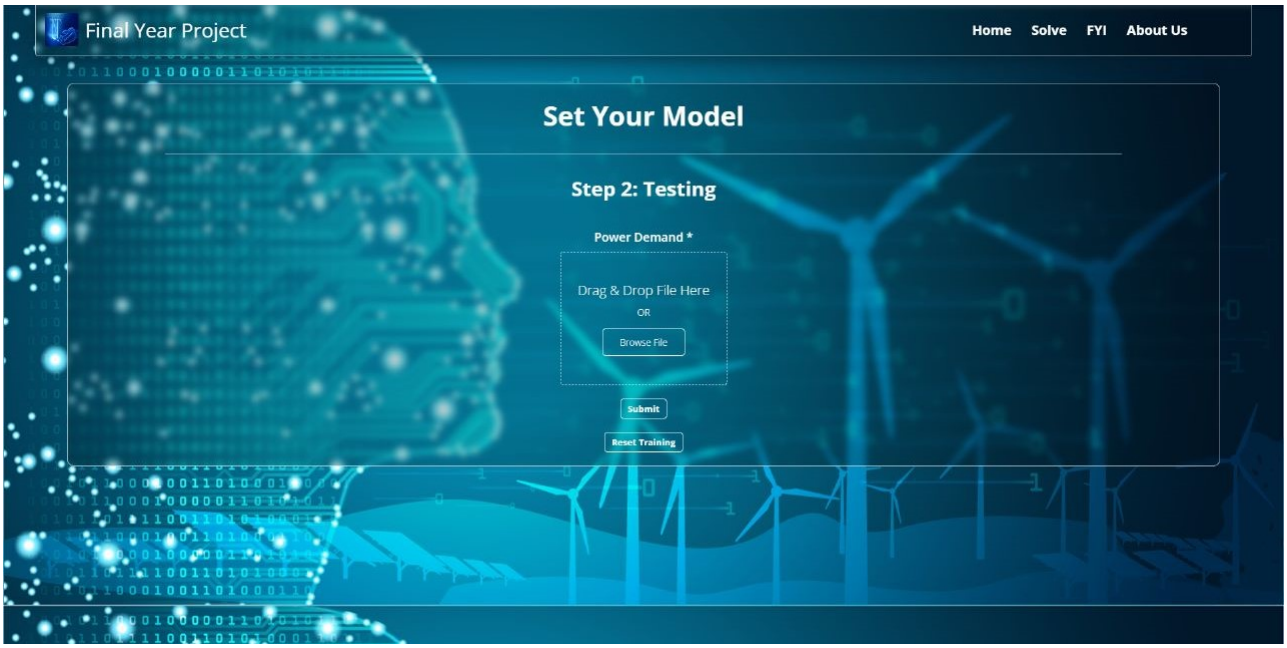


Figure 4.5: The user can now enter data to solve the DC-OPF.

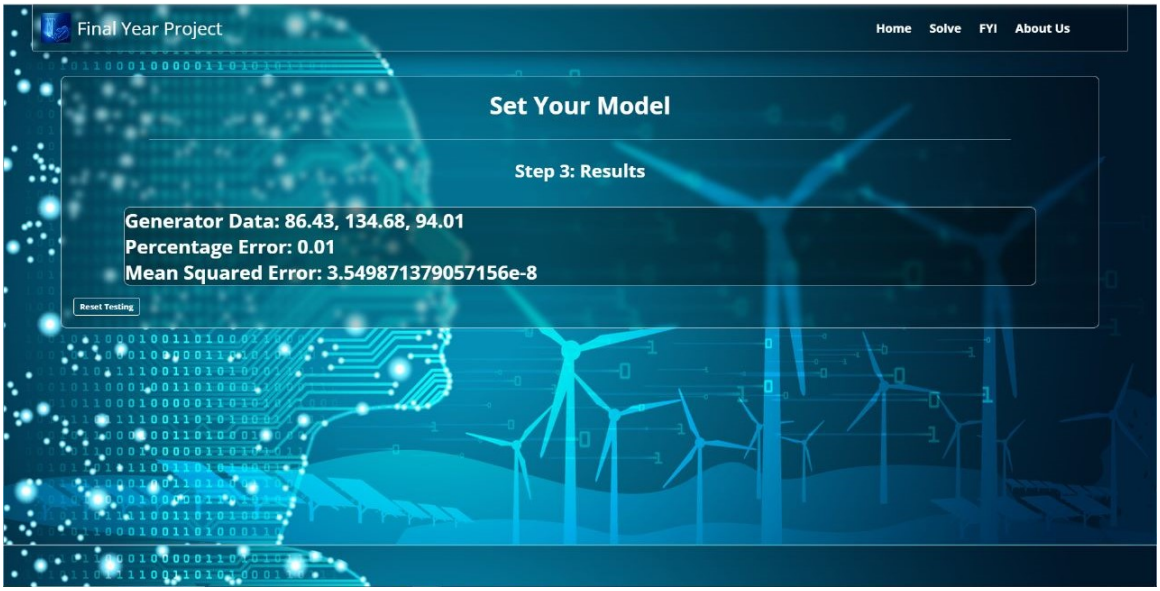


Figure 4.6: The results of the DC-OPF.

4.3 Discussions and Future Work

4.3.1 Projection Scheme

Although the NN model yielded satisfactory results even in the case of more strict constraints, theoretically, neural networks do not have convergence guarantees. The neural network solution may not lie within the feasible region. For that purpose, a projection scheme can be constructed to project back the solution obtained from the NN into the feasible region. By considering the L-1 projection (absolute value), the projection can be done by solving the following linear program:

$$min_U \|\hat{P_G} - U\|_1 \tag{4.4}$$

s.t. U satisfies constraints (3.9) - (3.11)

It should be noted that the absolute value objective function can be linearized by adding the constraint (4.5) and transforming the objective function into (4.6).

$$-K \leq \hat{P_G} - U \leq K \tag{4.5}$$

$$min_U \, K \tag{4.6}$$

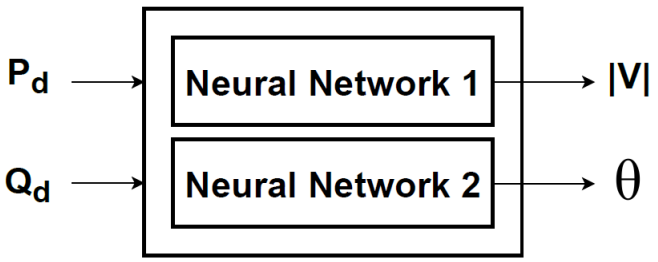


Figure 4.7: The AC-OPF neural network solver.

The projection scheme is able to retrieve a feasible solution from the original neural network solution. It can be used as a final step after obtaining the neural network output. However, considering, on one hand, the accuracy of the neural network and the role that the penalty term plays in eliminating infeasible cases, and on another hand the added time needed to solve the projection problem that could limit the speed-up, the projection scheme has been decided to be left out.

4.3.2 Generalization to AC-OPF

The project represents an initial endeavor in the pursuit of a comprehensive solution to the complete AC-OPF problem. This involves developing a neural network solver specifically designed for the DC-OPF case. By employing a comparable methodology, it is feasible to construct a neural network model capable of resolving the entire AC-OPF problem. Two neural networks are used, where the input of each is the active and reactive power demand, while the output of one is the voltage magnitude and that of the other is the voltage phase angle. The scheme is displayed in Fig. 4.7.

4.3.3 Deployment of the Platform on the Cloud

The platform provides an easy-to-use interface for addressing the optimal power flow problem. As previously mentioned, the platform's main advantage is its user-friendliness, requiring no prior expertise in machine learning or programming. The platform has been deployed locally on a computer, with plans to make it accessible to a wider audience by deploying it online. However, due to budgetary constraints and project timeline limitations, this step has been deferred. Further modifications can be implemented to the platform to enable its integration into supervisory control and data acquisition (SCADA) systems, particularly in terms of input and output data handling.

Chapter 5

Broad Impact: United Nations SDGs

The present project is in line with the fundamental principles of sustainable development, with a particular emphasis on the environmental and economic dimensions. It is pertinent to note that the project aligns with various Sustainable Development Goals (SDGs):

- **SDG 7 - Affordable Clean Energy**: The presented neural network solver exhibits remarkable computational efficiency by providing solutions within a significantly reduced time interval. This characteristic enables the solver to effectively tackle the intermittency challenge associated with renewable energy integration into the power grid. Moreover, the optimal power flow (OPF) problem formulation incorporates renewable energy resources as negative load nodes, while their associated generating units are assigned negligible costs due to the objective function's emphasis on minimizing system operating expenses.
- **SDG 12 - Responsible Production and Consumption**: Solving the OPF problem gives the most economic scheme for electric power production.
- **SDG 13 - Climate Action**: Besides enabling the incorporation of renewable energy sources, the OPF problem can also be leveraged to guarantee minimal emissions.

Chapter 6

List of Resources and Engineering Tools Needed

1. **Visual Studio**:the main software, we use it to write the codes and run our simulations.
2. **React**:The library for web and native user interfaces.Where we design and built our GUI. We had to learn how to use it.
3. **Cascading Style Sheets (CSS)**:is a stylesheet language used to describe the presentation of a document written in HTML or XML. We used to shape the screen of our website.
4. **MATPOWER**: we use it to build our power network and as an input data to our code. We had to learn how to use it, thus we watch videos and read papers about it.
5. **PyPower**: we use it to build our power network and as an input data to our code. We had to learn how to use it, thus we watch videos and read papers about it..
6. **IEEE Xplore**: a library dedicated to the world's highest quality technical literature in Optimal Power Flow and machine learning.
7. **Flask**: Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. We use it in our Backend code and we had to learn how to use it .
8. **Python**: the main programming language used for neural network training, testing and evaluation.
9. **Google Colab**: an integrated development environment (IDE) that runs the python code on Google servers, which has much more computational power.

10. **Pandas**: A Python library used for manipulating large datasets.
11. **NumPy**: A Python library for numerical computations for different data structures.
12. **Keras**: A Python library for implementing neural networks.
13. **Sci-kit Learn**: A Python library for evaluating the performance criteria used in neural network testing.

Appendix A

Weekly Meeting Minutes

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
2	3/2/20	12 pm	45 min	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehade

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- discussed the projection scheme
- discussed the GUI specifications
- discussed the penalty term in the NN loss function

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- the progress is good
- the division of work is acceptable

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- integrate the projection scheme - add penalty term - train NN for large power networks	Mohammad Fares El Hajj Chehade	10/2/2022
GUI Back-end	Mohamad Al Tawil	17/2/2022
GUI Front-end		10/2/2022

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
3	10/2/20	12 pm	30 mins	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehade

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- GUI demo
- NN progress
- projection scheme
- penalty for loss function

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- progress is good
- use for GUI a spreadsheet import
- try contingency analysis later

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- fix the NN projection and loss - some stats on the NN performance	Mohammad Fares El Hajj Chehade	17/2/2022
- fix the backend	Mohamad Al Tawil	17/2/2022
- fix the frontend	Karim Khalife	17/2/2022

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
3	2/17/23	12 pm	30 mins	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehade

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- discussed enhancements to the NN model
- discussed the GUI front-end improvements

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- progress is good
- the NN should be tried for different power systems
- GUI improvements are seen, back-end integration should be workd on

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- modify the loss function to account for branch flow limits - try NN for different power networks	Mohammad Fares El Hajj Chehade	24/2/2023
- write a program for the back-end - test on a simple NN model	Mohamad Al Tawil	24/2/2023
- improve the front-end - test on a simple NN model	Karim Khalife	24/2/2023

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
4	2/24/23	12 pm	30 mins	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehade

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- discussed the gui and needed improvements
- discussed the NN loss function
- discussed the possible use of PyPower

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- progress in NN is good
- certain improvements must be done regarding the gui

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- check the need for line flow violation update in the loss function	Mohammad Fares El Hajj Chehade	3/3/2023
- improve the gui by generalizing it to all power networks	Mohamad Al Tawil	3/3/2023
- check the working principle of PyPower	Karim Khalife	3/3/2023

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
5	3/3/20	12 pm	40 mins	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehade

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- we discussed the phase angle calculation for the code
- the possible shift to PyPower
- the possible shift to PyTorch

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- progress is good
- we do not need to add a penalty term
- we can shift to AC in a smooth way

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- shift to AC - find some statistics on the performance of the DC	Mohammad Fares El Hajj Chehade	10/03/2023
- try the GUI for the generalized power network	Mohamad Al Tawil	10/03/2023
- examine PyPower - create the dataset using PyPower	Karim Khalife	10/03/2023

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
6	10/3/20	12 pm	10 mins	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehade

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- discussed the MATPOWER branch flow limits
- discussed the transition to AC-OPF

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- AC-OPF shall be examined for the upcoming meetings
- line flow limits do not need to be necessarily examined

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- develop the NN for the case of AC-OPF	Mohammad Fares El Hajj Chehade	18/03/2023
- run the GUI for the generalized NN	Mohamad Al Tawil	18/03/2023
- examine PyPower as a possible replacement for MATPOWER	Karim Khalife	18/03/2023

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
8	3/17/23	12 pm	30 mins	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehade

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- the AC-OPF model working principle
- GUI for the DC model
- PyPower performance

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- the progress is good, we need to try to increase the range of variability
- two separate neural networks can be tried as a replacement to 1 larger network

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- check possibility of replacing 1 large NN with 2 smaller NNs for P and Q	Mohammad Fares El Hajj Chehade	24/03/2023
- design a GUI that stores the different datasets in its database	Mohamad Al Tawil	24/03/2023
- create the dataset using PyPower	Karim Khalife	24/03/2023

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
8	3/24/23	12 pm	30 mins	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehade

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- we discussed the performance of the decoupled AC OPF using NN
- we discussed the conversion of the code from MATPOWER to PyPower

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- we need to accelerate the testing
- we need to have meetings between us, team members, alone

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- convert the dataset generation code to Python	Karim Khalife	31/03/2023
- create a results section in the GUI	Mohamad Al Tawil	31/03/2023
- test for higher power networks and wider variability - check for generation limits violation	Mohammad Fares El Hajj Chehade	31/03/2023

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
9	3/31/23	12 pm	20 mins	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehade

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- we discussed the performance of the AC-OPF NN model
- we discussed the generation of the dataset using PyPower
- we discussed the wanted features in the GUI

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- we should focus for now on DC-OPF
- we should construct the GUI now for DC-OPF

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- test generation limits for DC-OPF - test a more complicated model for AC-OPF	Mohammad Fares El Hajj Chehade	06/04/2023
- construct the GUI for DC-OPF	Mohamad Al Tawil	06/04/2023
- construct the dataset using PyPower for larger networks	Karim Khalife	06/04/2023

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
10	6/4/20	6 pm	15 mins	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehade

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- we discussed the results of DC OPF for large power networks
- we discussed the schedule for the next meetings
- we discussed updates on the GUI

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- the progress is good
- we need to check the degree of generator limit violation in DC-OPF

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- connect the dataset generation code to the NN training code - generate datasets for DC-OPF for larger networks	Karim Khalife	12/04/2023
- fix the GUI to allow for power network uploads by the user in the form of a Python file	Mohamad Al Tawil	12/04/2023
- check the degree of generator limits violation in DC-OPF - test the AC-OPF NN with a more complex NN architecture	Mohammad Fares El Hajj Chehade	12/04/2023

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
11	4/13/23	6 pm	30 mins	(•) Advisor ( ) Students	Mohammad Fares El Hajj Chehae

Attendees:

Mohammad Fares El Hajj Chehade - Mohamad Al Tawil - Karim Khalife - Rabih Jabr

Briefly summarize the main discussions during the meeting:

- we discussed the performance of the AC-OPF solver
- we discussed the performance of the DC-OPF solver
- we discussed the performance of the GUI

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- results are excellent with regard to DC-OPF
- results are excellent with regard to GUI

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- wrap up DC-OPF NN and clean up the code	Mohammad Fares El Hajj Chehade	20/04/2023
- work on generating dataset with new version of pandas	Karim Khalife	20/04/2023
- add the testing of data on new instance to GUI	Mohamad Al Tawil	20/04/2023

American University of Beirut
EECE 502 – Final Year Project
Course Coordinator – Dr. Youssef Tawk
Minutes for the Weekly Group Meeting – Submitted per Group

Meeting #	Date:	Time:	Duration:	Meeting called by:	Minutes Taker:
12	4/20/23	11 am	5 mins	○ Advisor ◉ Students	Mohamad Fares El Hajj Chehade

Attendees:

Mohamad Fares El Hajj Chehade - Karim Khalife - Mohamad Al Tawil

Briefly summarize the main discussions during the meeting:

- discussed the GUI progress

Briefly summarize the conclusions drawn regarding the above-mentioned discussions:

- week progress is slow
- we need some time to meet with the advisor again

Enumerate the assigned tasks by the FYP advisor for each student + the deadline for delivery:

Assigned Task / Per Student	Name of the Student	Deadline
- clean the DC-OPF NN code - work on poster for conference	Mohamad Fares El Hajj Chehade	25/04/2023
- work on the poster for the conference - GUI	Mohamad Al Tawil	25/04/2023
- work on the poster for the conference	Karim Khalife	25/04/2023

Appendix B

FYP Poster

Optimal Power Flow via Machine Learning

EECE 502 Final Year Project

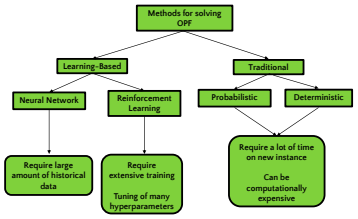
Mohammad F. El Hajj Chehade, Mohamad Al Tawil, Karim Khalife
Department of Electrical and Computer Engineering

Abstract

The optimal power flow (OPF) problem is fundamental to ensuring the efficient and reliable operation of a power system. The problem's objective is to minimize the operating costs of thermal resources within a power system. The recent integration of renewable energy sources into the power grid has led to rapid fluctuations in power generation, necessitating the presence of a faster OPF solver capable of balancing accuracy and speed. This paper proposes Artificial Neural Networks (ANNs) as a viable solution to the OPF problem. An algorithm is introduced to generate the necessary dataset, while a custom loss function is implemented to ensure that constraints are not violated. The results demonstrate a high level of accuracy for the solver, as well as a significant improvement in speed when compared to traditional solvers. Finally, a user-friendly platform is developed for solving the OPF problem using ANNs, which eliminates the need for knowledge of programming or machine learning.

Key words: optimal power flow, Artificial Neural Networks (ANNs), custom loss function, user-friendly platform

Background



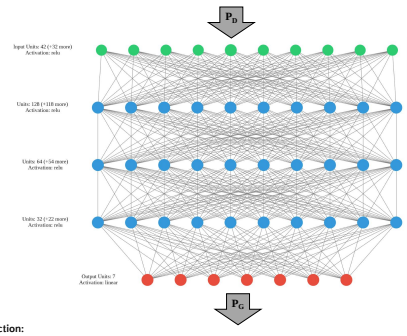
Objectives

Our project aims to develop an efficient neural network solver that meets the following requirements:

- Accelerate the solution process compared to traditional solvers
- Deliver highly accurate results
- Ensure that the solver does not violate any problem constraints
- Integrate the solver into a user-friendly graphical user interface (GUI) for ease of use.

Method

Neural Network Architecture:



Loss Function:

$$\mathcal{L}_{new} = w_{MSE}\mathcal{L}_{MSE} + w_{pen}\mathcal{L}_{pen}$$

$$\mathcal{L}_{MSE} = \sum_{i=1}^{N}\frac{1}{|G|}\sum_{j=1}^{|G|}(\widehat{P_{Gij}} - P_{G_{ij}})^2$$

$$\mathcal{L}_{pen} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|G|}\sum_{j=1}^{|G|} max(\hat{P}_{ij} - P_{ij}^{max}, 0) + max(P_{ij}^{min} - \hat{P}_{ij}, 0)$$

Algorithm Data Generation Scheme

Specify case file, N and δ
$P_D \leftarrow$ default demand values
for N **do**
  for $i \in$ demand nodes **do**
    $\tilde{P}_{D_i} \sim U(P_{D_i} - \delta P_{D_i}, P_D + \delta P_{D_i})$
  end for
  $\tilde{P}_G \leftarrow$ run (DC-OPF | $\tilde{P}_D$)
  Save $\tilde{P}_D, \tilde{P}_G$
end for

Results

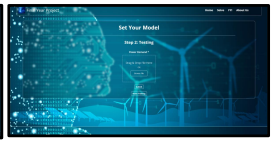
Testing Results:

Power Network	MSE	MAPE(%)	NN(M)	MATPO WER(ms)	Speed-up
IEEE-30	2x10	0.007	50	322	x6
IEEE-57	5.2x10	0.09	50	522	x11
IEEE-118	4.7x10	0.01	50	1131	x23
IEEE-300	1.5x10	0.02	50	2166	x43

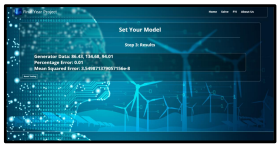
Graphical User Interface (GUI):

Step 1: Enter the power network to train:

Step 2: Enter the demand data to solve for new examples:



Step 3: The solution of the OPF with the performance metrics of the neural network:



Conclusions

- This study presents a neural network approach to solve the optimal power flow problem.
- The neural network is evaluated using the DC-OPF case, and it provides accurate solutions and faster speed compared to traditional solvers.
- A custom loss function is proposed to ensure that the constraints are not violated.
- The developed solver is presented and tested.
- A user-friendly platform is created for the solver, which does not require expertise in programming or machine learning.

Bibliography

[1] M. B. Cain, R. P. O'neill, A. Castillo, *et al.*, "History of optimal power flow and formulations," *Federal Energy Regulatory Commission*, vol. 1, pp. 1–36, 2012.

[2] M. Huneault and F. D. Galiana, "A survey of the optimal power flow literature," *IEEE transactions on Power Systems*, vol. 6, no. 2, pp. 762–770, 1991.

[3] M. A. Abido, "Optimal power flow using particle swarm optimization," *International Journal of Electrical Power & Energy Systems*, vol. 24, no. 7, pp. 563–571, 2002.

[4] J. A. Momoh, R. Adapa, and M. El-Hawary, "A review of selected optimal power flow literature to 1993. i. nonlinear and quadratic programming approaches," *IEEE transactions on power systems*, vol. 14, no. 1, pp. 96–104, 1999.

[5] R. D. Christie, B. F. Wollenberg, and I. Wangensteen, "Transmission management in the deregulated environment," *Proceedings of the IEEE*, vol. 88, no. 2, pp. 170–195, 2000.

[6] J. Momoh, R. Koessler, M. Bond, B. Stott, D. Sun, A. Papalexopoulos, and P. Ristanovic, "Challenges to optimal power flow," *IEEE Transactions on Power systems*, vol. 12, no. 1, pp. 444–455, 1997.

[7] H. W. Dommel and W. F. Tinney, "Optimal power flow solutions," *IEEE Transactions on power apparatus and systems*, no. 10, pp. 1866–1876, 1968.

[8] R. P. O'Neill, T. Dautel, and E. Krall, "Recent iso software enhancements and future software and modeling plans," *Federal Energy Regulatory Commission, Tech. Rep*, 2011.

[9] Y. Tang, K. Dvijotham, and S. Low, "Real-time optimal power flow," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2963–2973, 2017.

[10] M. Aien, M. Rashidinejad, and M. F. Firuz-Abad, "Probabilistic optimal power flow in correlated hybrid wind-pv power systems: A review and a new approach," *Renewable and Sustainable Energy Reviews*, vol. 41, pp. 1437–1446, 2015.

[11] T. Niknam, H. Z. Meymand, and H. D. Mojarrad, "An efficient algorithm for multi-objective optimal operation management of distribution network considering fuel cell power plants," *Energy*, vol. 36, no. 1, pp. 119–132, 2011.

[12] E. Naderi, H. Narimani, M. Fathi, and M. R. Narimani, "A novel fuzzy adaptive configuration of particle swarm optimization to solve large-scale optimal reactive power dispatch," *Applied Soft Computing*, vol. 53, pp. 441–456, 2017.

[13] R. A. Jabr, "A primal-dual interior-point method to solve the optimal power flow dispatching problem," *Optimization and Engineering*, vol. 4, no. 4, pp. 309–336, 2003.

[14] R. A. Jabr, "Radial distribution load flow using conic programming," *IEEE transactions on power systems*, vol. 21, no. 3, pp. 1458–1459, 2006.

[15] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power systems*, vol. 27, no. 1, pp. 92–107, 2011.

[16] J. A. Momoh and J. Zhu, "Improved interior point method for opf problems," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 1114–1120, 1999.

[17] R. Burchett, H. Happ, and D. Vierath, "Quadratically convergent optimal power flow," *IEEE Transactions on Power Apparatus and Systems*, no. 11, pp. 3267–3275, 1984.

[18] W. Stadlin and D. Fletcher, "Voltage versus reactive current model for dispatch and control," *IEEE Transactions on Power Apparatus and Systems*, no. 10, pp. 3751–3760, 1982.

[19] D. I. Sun, B. Ashley, B. Brewer, A. Hughes, and W. F. Tinney, "Optimal power flow by newton approach," *IEEE Transactions on Power Apparatus and systems*, no. 10, pp. 2864–2880, 1984.

[20] M. Rahli and P. Pirotte, "Optimal load flow using sequential unconstrained minimization technique (sumt) method under power transmission losses minimization," *Electric Power Systems Research*, vol. 52, no. 1, pp. 61–64, 1999.

[21] D. Cao, W. Hu, X. Xu, Q. Wu, Q. Huang, Z. Chen, and F. Blaabjerg, "Deep reinforcement learning based approach for optimal power flow of distribution networks embedded with renewable energy and storage devices," *Journal of Modern Power Systems and Clean Energy*, vol. 9, no. 5, pp. 1101–1110, 2021.

[22] X. Yan and V. H. Quintana, "Improving an interior-point-based opf by dynamic adjustments of step sizes and tolerances," *IEEE Transactions on Power Systems*, vol. 14, no. 2, pp. 709–717, 1999.

[23] T. Niknam, M. Zare, and J. Aghaei, "Scenario-based multiobjective volt/var control in distribution networks including renewable energy sources," *IEEE Transactions on Power Delivery*, vol. 27, no. 4, pp. 2004–2019, 2012.

[24] Y. Xu, Z. Y. Dong, R. Zhang, and D. J. Hill, "Multi-timescale coordinated voltage/var control of high renewable-penetrated distribution systems," *IEEE Transactions on Power Systems*, vol. 32, no. 6, pp. 4398–4408, 2017.

[25] M. R. Adaryani and A. Karami, "Artificial bee colony algorithm for solving multi-objective optimal power flow problem," *International Journal of Electrical Power & Energy Systems*, vol. 53, pp. 219–230, 2013.

[26] M. A. Shaheen, H. M. Hasanien, S. Mekhamer, and H. E. Talaat, "Optimal power flow of power networks with penetration of renewable energy sources by harris hawks optimization method," in *2020 2nd International Conference on Smart Power & Internet Energy Systems (SPIES)*, pp. 537–542, IEEE, 2020.

[27] J. Li, R. Zhang, H. Wang, Z. Liu, H. Lai, and Y. Zhang, "Deep reinforcement learning for optimal power flow with renewables using spatial-temporal graph information," *arXiv preprint arXiv:2112.11461*, 2021.

[28] T. Soares, R. J. Bessa, P. Pinson, and H. Morais, "Active distribution grid management based on robust ac optimal power flow," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 6229–6241, 2017.

[29] J. F. Franco, L. F. Ochoa, and R. Romero, "Ac opf for smart distribution networks: An efficient and robust quadratic approach," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4613–4623, 2017.

[30] E. Dall'Anese, K. Baker, and T. Summers, "Chance-constrained ac optimal power flow for distribution systems with renewables," *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3427–3438, 2017.

[31] P. Fortenbacher, A. Ulbig, S. Koch, and G. Andersson, "Grid-constrained optimal predictive power dispatch in large multi-level power systems with renewable energy sources, and storage devices," in *IEEE PES Innovative Smart Grid Technologies, Europe*, pp. 1–6, IEEE, 2014.

[32] W. Stadlin and D. Fletcher, "Voltage versus reactive current model for dispatch and control," *IEEE Transactions on Power Apparatus and Systems*, no. 10, pp. 3751–3760, 1982.

[33] V.-H. Bui, A. Hussain, and H.-M. Kim, "Double deep q-learning-based distributed operation of battery energy storage system considering uncertainties," *IEEE Transactions on Smart Grid*, vol. 11, no. 1, pp. 457–469, 2019.

[34] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE access*, vol. 7, pp. 133653–133667, 2019.

[35] J. Li, R. Zhang, H. Wang, Z. Liu, H. Lai, and Y. Zhang, "Deep reinforcement learning for optimal power flow with renewables using spatial-temporal graph information," *arXiv preprint arXiv:2112.11461*, 2021.

[36] H. Zhen, H. Zhai, W. Ma, L. Zhao, Y. Weng, Y. Xu, J. Shi, and X. He, "Design and tests of reinforcement-learning-based optimal power flow solution generator," *Energy Reports*, vol. 8, pp. 43–50, 2022.

[37] Z. Yan and Y. Xu, "Real-time optimal power flow: A lagrangian based deep reinforcement learning approach," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 3270–3273, 2020.

[38] E. R. Sanseverino, M. Di Silvestre, L. Mineo, S. Favuzza, N. Nguyen, and Q. Tran, "A multi-agent system reinforcement learning based optimal power flow for islanded microgrids," in *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, pp. 1–6, IEEE, 2016.

[39] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia*, pp. 21–49, Springer, 2008.

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[41] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, "Short-term residential load forecasting based on resident behaviour learning," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1087–1088, 2017.

[42] X. Pan, T. Zhao, and M. Chen, "Deepopf: Deep neural network for dc optimal power flow," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–6, 2019.

[43] K. Baker, "Emulating ac opf solvers with neural networks," *IEEE Transactions on Power Systems*, vol. 37, no. 6, pp. 4950–4953, 2022.

[44] X. Pan, T. Zhao, M. Chen, and S. Zhang, "Deepopf: A deep neural network approach for security-constrained dc optimal power flow," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 1725–1735, 2020.

[45] Y. Zhou, B. Zhang, C. Xu, T. Lan, R. Diao, D. Shi, Z. Wang, and W.-J. Lee, "A data-driven method for fast ac optimal power flow solutions via deep reinforcement learning," *Journal of Modern Power Systems and Clean Energy*, vol. 8, no. 6, pp. 1128–1139, 2020.

[46] X. Qi, G. Wu, K. Boriboonsomsin, M. J. Barth, and J. Gonder, "Data-driven reinforcement learning–based real-time energy management system for plug-in hybrid electric vehicles," *Transportation Research Record*, vol. 2572, no. 1, pp. 1–8, 2016.

[47] R. D. Zimmerman and C. E. Murillo-Sánchez, "Matpower 6.0 user's manual," *Power Systems Engineering Research Center*, vol. 9, 2016.

[48] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, 2011.

[49] "Ieee draft standard for electric power systems communications -distributed network protocol (dnp3)," *IEEE P1815/D08, January 2012*, pp. 1–858, 2012.

[50] "Ieee standard for exchanging information between networks implementing iec 61850 and ieee std 1815(tm) [distributed network protocol (dnp3)]," *IEEE Std 1815.1-2015 (Incorporates IEEE Std 1815.1-2015/Cor 1-2016)*, pp. 1–358, 2016.

[51] "Ieee grid vision 2050 roadmap," *IEEE Grid Vision 2050 Roadmap*, pp. 1–15, 2013.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015.