FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

CONGRATS – Convolutional Networks in GPU-based Reliability Assessment of Transmission Systems

Rodrigo Gonçalves de Morais

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Professor Vladimiro Henrique Barrosa Pinto de Miranda, Ph.D.

Second Supervisor: Leonel de Magalhães Carvalho, Ph.D.

July 24, 2021

Resumo

A Simulação Monte Carlo é uma técnica bem conhecida e utilizada em estudos de fiabilidade de sistemas de energia porque pode incluir as particularidades do sistema de energia, principalmente o seu comportamento estocástico e a incerteza sobre a carga. No entanto, o tempo computacional requerido pela simulação é elevado, especialmente quando os sistemas são extremamente fiáveis e os estudos incluem o sistema de transmissão. Neste último caso, é necessário um procedimento complexo envolvendo vários algoritmos, que consomem muito tempo, para avaliar se os sistemas de transmissão operam dentro dos seus limites técnicos.

O principal objectivo desta dissertação é aplicar uma técnica de *Deep Learning*, *Convolutional Neural Network (CNN)*, treinada em GPU, à simulação de Monte Carlo para estudos de fiabilidade em sistemas de energia (incluindo a geração e transmissão) e verificar a relevância desta abordagem em termos de eficiência temporal.

Para atingir o objectivo proposto, a *Convolutional Neural Network* foi treinada para classificar instantaneamente os estados operacionais do sistema relativamente à perda de carga. A CNN é testada num pequeno conjunto de amostras com resultados encorajadores. Mais tarde, a CNN é aplicada à simulação de Monte Carlo. A estratégia desenvolvida, MCS-CNN, é utilizada para avaliar a fiabilidade do sistema IEEE RTS 79, considerando a simulação não sequencial e sequencial.

Abstract

The Monte Carlo Simulation is a well-known technique used in power system reliability studies because it can include the particularities of the power system, mainly his stochastic behaviour and uncertainty on the load. The flaw in this method is the computational time required by the simulation, especially when systems are highly reliable and studies include the transmission system. In the latter case, a complex procedure involving several time-consuming algorithms is needed to assess if transmission systems operate under its operational limits.

The main goal of this dissertation is to apply a deep learning technique, Convolutional Neural Network (CNN), trained in GPU, to the Monte Carlo Simulation for composite (generation and transmission) reliability assessment studies and verify the relevance of this approach in terms of time efficiency.

To accomplish the proposed goal, a Convolutional Neural Network was trained to instantly classify operational system states based on the loss of load. This network is tested on a small set of samples with encouraging results. Later, CNN is applied to Monte Carlo Simulation. The developed strategy, MCS-CNN, is used to assess the reliability of the IEEE RTS 79 system, considering non-sequential and sequential simulation.

Acknowledgments

I would like to begin this acknowledgement section by thanking Professor Doctor Vladimiro Miranda for his encouragement, key insights and the opportunity to work on such an exciting theme. To Doctor Leonel Carvalho and Engineer Inês Alves, I express my appreciation for all the valuable contributions to the work developed.

A special mention goes to my girlfriend Francisca for her presence, understanding, motivation and, most of all, her love.

Finally, I would like to express my extreme gratitude to my parents and my brothers. Their unconditional support and love was crucial during this five years journey.

Rodrigo Morais

"No muscles without strength,
friendship without trust,
opinion without consequence,
change without aesthetics,
age without values,
life without effort,
water without thirst,
food without nourishment,
love without sacrifice,
power without fairness,
facts without rigor,
statistics without logic,
mathematics without proof,
teaching without experience,
politeness without warmth,
values without embodiment,
degrees without erudition,
militarism without fortitude,
progress without civilization,
friendship without investment,
virtue without risk,
probability without ergodicity,
wealth without exposure,
complication without depth,
fluency without content,
decision without asymmetry,
science without skepticism,
religion without tolerance,
nothing without skin in the game."

Nassim Nicholas Taleb, in *Skin in the Game: Hidden Asymmetries in Daily Life*

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial Neural Network
API	Application Programming Interface
CNN	Convolutional Neural Network
CE	Cross Entropy
CPU	Central Process Unit
EENS	Expected Energy Not Served
EPNS	Expected Power Not Supplied
f.o.r	Forced Outage Rate
GPU	Graphics Processing Unit
HL0	Hierarchical Level 0
HLI	Hierarchical Level I
HLII	Hierarchical Level II
HLIII	Hierarchical Level III
IEEE RTS 79	IEEE Realibility Test System
LOLE	Loss of Load Expectation
LOLF	Loss of Load Duration
LOLP	Loss of Load Probability
MCS	Monte Carlo Simulation
MTTF	Mean Time to Failure
MTTR	Mean Time to Repair
NSMCS	Non-Sequential Monte Carlo Simulation
OPF	Optimal Power Flow
SMCS	Sequential Monte Carlo Simulation

Chapter 1

Introduction

1.1 Context and Motivation

From storing food, communicating with family, and performing financial transactions, nearly every facet of modern life depends on electricity. It is a central and fundamental pillar to society.

Electric Power systems are responsible for responding to load demand and thus keeping society moving. Reliability studies have a predominant role on assessing the power system's inherent risks. However, recent changes in electric grid structure have shifted the way reliability is seen.

The electric system is transitioning from a centralised and fossil-fuelled generation to a decentralised structure composed of small generators powered by renewable sources. Now, reliability studies have to consider the stochastic behaviour of these renewable energy sources in addition the *forced outage rate* of system components and load demand uncertainties. Also, as the power system grows in complexity, a more robust set of informative reliability indices needs to be calculated, and these studies are often repeated many times.

As a result of this changing paradigm, Monte Carlo Simulation arises as a valuable method in the field of power system reliability because of its capability to incorporate the behaviour of the modern power system. MCS can compute various reliability indices such as Loss of Load Probability and Loss of Load Expectation. When considering chronological modelling, frequency and duration indices can be calculated. Despite all this, the time required to obtain accurate estimations with this method remains his most significant drawback.

Techniques such as Importance Sampling [15] were developed and successful in reducing the overall process time by shortening the number of iterations needed to complete the simulation.

Deep learning methodologies have found their way in many areas, and power system reliability is no exception. Artificial Neural networks were applied to Monte Carlo simulation for reliability assessment [16] to decrease the computational effort required for the simulation on sample evaluation. Convolutional neural networks are typically used in image processing, and their ability on pattern recognition makes them attractive to be also applied on MCS.

This dissertation aims to develop a method to speed up Monte Carlo Simulation for Composite system reliability assessment based on Convolutional Neural Networks, trained on GPU.

1.2 Objectives

The main goal in this dissertation is to implement a Convolutional Neural Network in Monte Carlo process for reliability assessment studies.

To achieve this goal the following objectives will be set:

- Definition of a "Loss of Load Classifier" based on CNN capable of accurately determine, given the system operational information, if a system can supply the load demand. This classifier will be the building block of the thesis.
- Implement the trained classifier in Monte Carlo Simulation, MCS-CNN;
- Assessment of the composite reliability of electric systems using both models, the classic MCS and the developed MCS-CNN, and considering non-sequential and sequential simulation;

1.3 Dissertation Outline

The present thesis is composed by eight chapters (including this Introduction).

Chapter 2 provides an introduction to electric power systems reliability and concepts that outline the theorical methodology for Monte Carlo Simulation.

Chapter 3 addresses the state of the art techniques used to accelerate Monte Carlo simulation processes.

Chapter 4 provides information about deep learning and its place in the chain of Artificial Intelligence. The intuition behind Convolutional Networks is addressed, and the frameworks used to implement it in this thesis are presented.

In **Chapter** 5 details related to construction, training and testing of the developed classifier are documented. Information about the system data used to obtain samples for network training and data preprocessing are also mentioned.

Chapter 6 reports the methodology used on Monte Carlo composite system reliability assessment. The integration of a Convolutional Network in Monte Carlo simulation is explained.

In **Chapter** 7 results of the algorithms implementation, defined in the previous chapter, are presented and discussed. Two variations of the IEEE RTS 79 are used to assess the performance of the proposed method in comparison with crude versions of Monte Carlo Simulation.

Finally, on **Chapter** 8 a conclusion about the work carried out in this thesis and suggestions for future upgrades are presented.

Chapter 2

Power System Reliability

This chapter provides essential definitions of power system reliability and theoretical information about the concepts behind Monte Carlo simulation.

2.1 Adequacy and Security

Reliability studies on Power Systems evaluate if a system can perform its intended function adequately, supplying the load demand without interruptions. Given the complexity of the power system, reliability studies are divided into two fundamental fields: Adequacy and Security [1].

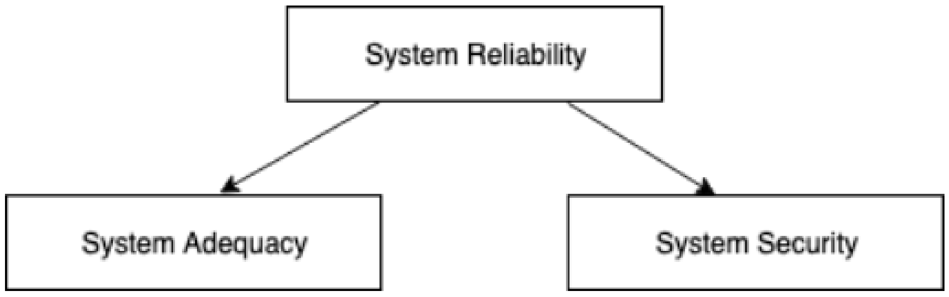


Figure 2.1: Division of Power System Reliability [1]

Adequacy is related to the existence of sufficient resources to meet the load demand and operational constrains [1]. These resources include the generation, transmission and distribution equipment needed to supply the electric energy to consumers. Shortly, adequacy refers to the static conditions of the system.

Security is linked with the ability of the system to respond to dynamic or transient disturbances that might arise within it [1]. The origin of these disturbances may be due to natural phenomena such as lightning. However, most of them are caused by internal events like load switching, breaker switching, fuse disconnection, short-circuit, and islanding [17].

Assessing security reliability of a system is an arduous task on account of the complexity to model dynamic behavior of the electric power systems [18].

2.2 Functional Zones and Hierarchical levels

Modern electric power systems are complex. Its extensive structures can be described with various levels of detail, allowing different system component representations and techniques for reliability adequacy assessment studies.

Typically, a power system is seen as a threefold system composed of Functional Zones: Generation, Transmission and Distribution. This division is relevant since most utilities are either divided into these zones for organization, planning, operation and analysis or are solely responsible for one of these functions [1].

The functional zones can be combined and organized into hierarchical levels, represented in Figure 2.2. Adequacy studies can be conducted in each functional zone or on hierarchical levels.

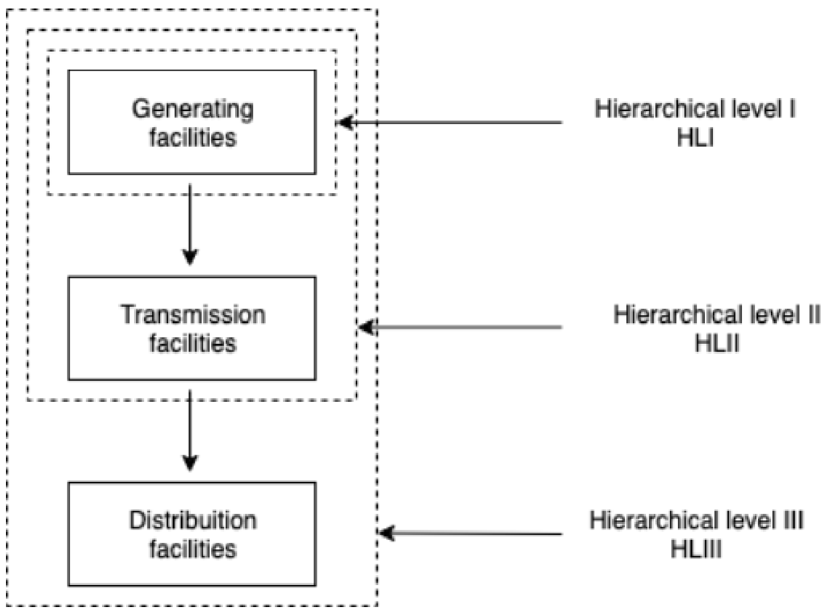


Figure 2.2: Hierarchical Levels [1]

Hierarchical Level I (HLI) is related to generation facilities, and the purpose of adequacy studies on this level is to determine the capability of the system generation to meet the total system demand [1].

Hierarchical Level II (HLII) includes generation and transmission components, and adequacy assessments aim to determine the ability to supply the bulk consumption points [1]. This dissertation is focused on HLII adequacy assessment.

Lastly, Hierarchical Level III (HLIII) comprises all functional zones. Adequacy studies on this level evaluate the overall system adequacy however due to the extensive dimension of the power system, this studies are not usually conducted since it would require a vast computation time and effort [1]. Alternately, adequacy assessment of distribution functional zone is done separately assuming approximate models from the rest of functional zones.

In the last decades, hierarchical level (HL) concept have been adjusted mainly due to important changes in the power industry [2]. A new functional zone, Energy (HL0 on figure 2.3), was added to take into account the renewable sources of energy [2] which are defined by their intermittence and variability on the contrary to classic non-renewable generation units, where primary resources are always available.

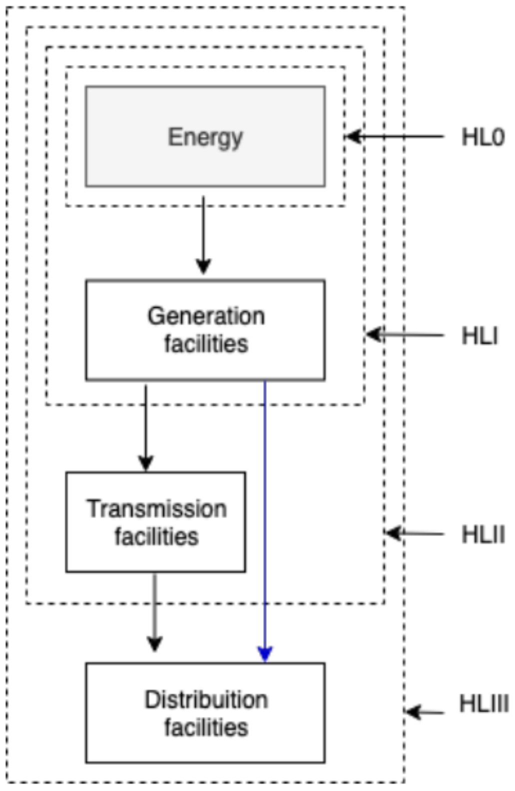


Figure 2.3: Evolution of the functional zones [2]

2.3 Reliability Assessment Methods

Nowadays, the use of probabilistic methods in power systems reliability assessment is consensual [19]. Data related to components behaviour such as failures and outage duration is collected and processed to create statistical measures and indices later used in reliability studies.

In this scope, two main categories define the techniques used: analytical and simulation.

Analytical techniques represent the system by mathematical models and can calculate the exact value of reliability indices. These techniques were highly developed, and many technical papers and books are dedicated to these methods [20] [21].

Simulation methods like the Monte Carlo Simulation, on the other hand, estimate the reliability indices and their confidence interval by simulating the random behaviour of the system [22]. The evolution of computation and the development of accessible and more powerful computers made these methods popular for power system reliability assessment.

Both methods can be used to execute reliability studies. However, with simulation methods, the system can be studied without "building it", which can be advantageous when power systems are vast and complex. On the downside, simulation time to retrieve accurate estimations can be vast and discouraging.

2.3.1 Analytical Methods

Analytical methods rely on an extensive analysis of all possible system states, therefore obtaining the probability density function of all system states. Then, reliability indices can be calculated according to equation 2.1 [23].

$$E[H(x)] = \sum_{x \in A} H(x)p(x) \tag{2.1}$$

On equation 2.1, x is a system state, A is the set of all system states, $p(x)$ is the probability of the system state x and $H(x)$ is the output of the test function H for the considered reliability index. $E[H(x)]$ is the exact value of the reliability index.

Since this method requires an analysis of all system states, the computational effort required for estimate an index is proportional to the complexity of the power system. As a means to reduce this effort is common to simplify the set of system states by ignoring the states which possess a probability inferior to *a priori* specified value. Truncation based on state probability is not the only criteria used for this simplification, for instance, first order contingencies or simply knowledge of the system by the operator can also be contemplated.

Although truncation of state space reduces computational effort this technique can lead to untrustworthy reliability indices, since there is always the chance that some ignored state, although unlikely to happen, might be important to the correct assessment of reliability.

2.3.2 Simulation Methods

Simulation methods, like MCS, provide estimates of the reliability indices with an interval of confidence. The advantage of the MCS is that the number of samples needed to ensure a certain level of confidence for the estimated indices is not related to the size of the power system but instead with the variance of the sampled states [22].

Simulation-based methods can be classified according to how system states are sampled. If a random state space representation is used, then the MCS method is called non-sequential. Conversely, when system states are sampled taking into account the events chronology, the method is called sequential. Nonetheless, methods like quasi-sequential [24] or pseudo-sequential [25] were developed and present a middle term between chronological and random sampling.

The MCS can be seen as recurrent application of a process called Sampling. It relies on a set of samples with N dimension inside the space of states and the posterior estimation of its expected value (equation 2.2) and variance (equation 2.3) [26].

$$E(X) = \sum_i p(X_i) X_i = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{2.2}$$

$$V(X) = E\left([X - E(X)]^2\right) = \frac{1}{N}\sum_{i=1}^{N} (X_i - E(X))^2 \tag{2.3}$$

The equations 2.4 and 2.5 are only valid when working with discrete and finite sets. For demonstration purposes consider state space as finite set although very large, where multiple N samples can be retrieved from.

$$E(\bar{X}) = E\left(\frac{1}{N}\sum_{i=1}^{N} X_i\right) = \frac{1}{N}\sum_{i=1}^{N} E(X_i) = \frac{1}{N} N\mu = \mu \tag{2.4}$$

$$V(\bar{X}) = V\left(\frac{1}{N}\sum_i = 1^N X_i\right) = \frac{1}{N^2}\sum_{i=1}^{N} V(X_i) = \frac{1}{N^2} N\sigma^2 = \frac{1}{N}\sigma^2 \tag{2.5}$$

Now, consider a random extraction process where a set of N samples is collected and $\bar{X}$ is the mean of the collected sample set.

By 2.4, $\bar{X}$ is an unbiased estimator of the state space mean μ [27].

A variance estimation of state space distribution, according to 2.5 can be obtained by calculating the variance of $\bar{X}$. Keep in mind that the actual variance value of estimator $\bar{X}$ depends on the selected samples, however for a larger sample size, a lower variance for $\bar{X}$ will always be produced [27].

The convergence of the MCS methods is monitored by the coefficient of variation β of the reliability indices estimates, and it is calculated according to:

$$\beta^2 = \frac{V(X)}{\hat{E}(X)^2} \tag{2.6}$$

By rearranging equation 2.6, it is possible to obtain:

$$N = \frac{V(F)}{[\beta \hat{E}(F)]^2} \tag{2.7}$$

The equation 2.7 highlights an important aspect about the convergence characteristics of the Monte Carlo Simulation, later used on acceleration techniques described on chapter 3: the variation coefficient β is related to the number of iterations, concluding that, if one wants to reduce the number of simulation iterations, the variance of the samples must be reduced.

2.3.3 Confidence Intervals

As mentioned before, the estimation of reliability indices, done by the MCS, is followed by a confidence interval, meaning that there will be a given probability that this interval contains the exact value of the reliability indices. The knowledge of the variance of the samples allows the estimation of an interval of confidence for the estimated values.

Figure 2.4 represents the probability density function of the normal distribution N(0,1). With mean μ = 0 and variance σ^2 = 1, the symmetrical interval centred in the mean, and with a semi-width of two standard deviations, corresponds to a probability obtained by the integral of the distribution between the limits of the interval. Therefore, if an interval associated to 2σ is defined, one may say that the confidence level associated to the statement that a given result may be inside the interval is α = 95.45%. For σ and 3σ, the confidence level becomes α=68.27% and α=99.73%, respectively, as seen in figure 2.4.

It is common to take a confidence interval of 95% (1.96σ) rather than 95.45%, and a confidence interval of 99% (2.575σ) instead of 99.73%. The determination of a confidence interval (CI), for instance considering a confidence interval of 95% is calculated through equation 2.8.

$$CI(95\%) = \left[\hat{E} - 1.96\frac{\hat{\sigma}}{\sqrt{N}}, \hat{E} + 1.96\frac{\hat{\sigma}}{\sqrt{N}}\right] \tag{2.8}$$

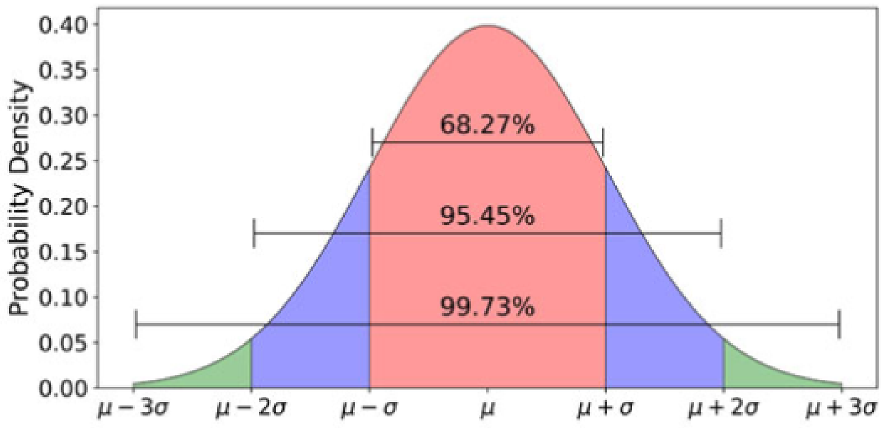


Figure 2.4: Normal Distribution N(0,1) with three standard deviations represented [3]

2.4 Reliability Indices for HLII studies

Reliability indices can be classified into two main categories: predictive or empirical [28]. Predictive indices are determined from information collected, for instance, through simulations where reliability data from system components is used to portrait the operational, physical and temporal characteristics of the system in study [28].

Empirical indices, contrarily, are determined from direct observation, such as by collecting data at the location of interest [28].

The focus of this thesis is on predictive indices. Within this category, there is also divisions such :

- **Probability and expectation indices**: These indices include loss of load probability (LOLP) and loss of load expectation, expressed in hours (LOLE);
- **Frequency and duration indices**: These indices indicate the frequency and duration of reliability events, and includes loss of load frequency (LOLF) and loss of load duration (LOLD);
- **Severity Indices**: These indices convey the severity of reliability indices [28], and includes, for instance, expected power not served (EPNS), expressed in MW per year and expected energy not served (EENS), expressed in MWh per year;

The above indices constitute probabilistic measures of the system reliability. Depending on the hierarchical level of study, is common to find different designations for this indices. The presented indices are used for composite reliability assessment (HLII studies).

2.5 Non-Sequential Monte Carlo Simulation for Reliability Assessment

In non-sequential MCS, system states are sampled by taking “snapshots” of its stochastic behavior, hence without considering any time-dependency between consecutive states. A system state is composed by the combination of all system components and the load state.

2.5.1 Component State

Each component state can be determined by sampling the probability that the component appears in that state [22]. A more detailed description of the algorithm behind the components state sampling is presented in Chapter 6.

The component model can be described by a two state Markov chain, as represented in Figure 2.5. The component's state is classified as a failure if it is out of service or a success if it is fully available. The transition between states is modelled by the expected failure rate (λ) and expected repair rate (μ). These variables are defined by statistical studies, which consider historical data about the previous failures of the component and the behaviour of similar components.

The probability of a generator to be out of service, also known as *forced outage rate* (*f.o.r*), is given by:

$$f.o.r = \frac{\lambda}{\lambda + \mu} \tag{2.9}$$

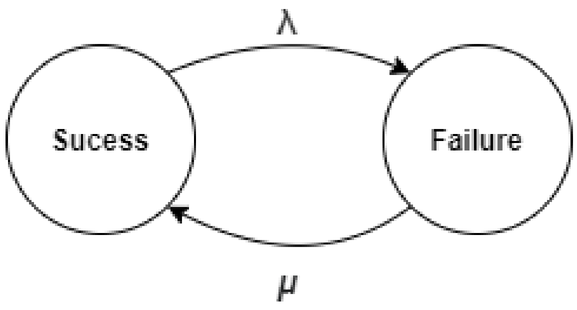


Figure 2.5: Two state Markov model for components

On components reliability data is usual to find the the parameters Mean Time to Failure (MTTF) and Mean Time to Repair (MTTF). These parameters are related to λ and μ by:

$$MTTF = \frac{1}{\lambda} \tag{2.10}$$

$$MTTR = \frac{1}{\mu} \tag{2.11}$$

2.5.2 Load Model

The load is modelled by a daily peak load variation curve which represents the cumulative probability of occurrence of certain load levels over a specified period, usually a year (Figure 2.6). These load values are obtained through forecasting studies and analyses of the previous series.

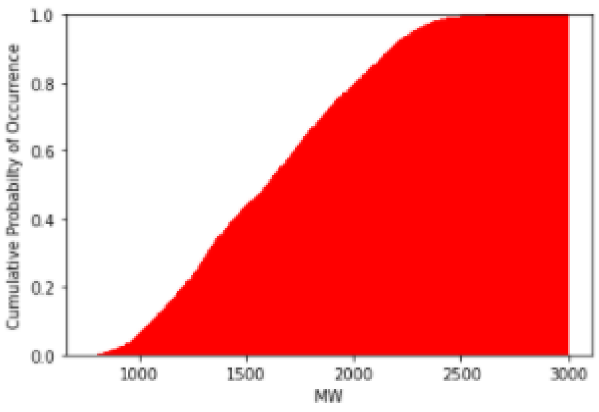


Figure 2.6: Example of a daily peak load variation curve obtained with load values from IEEE RTS case

2.5.3 Indices Estimation

The reliability indices on non-sequential MCS are estimated according to:

$$\hat{E}(H(X)) = \frac{1}{N}\sum_{i=1}^{N} H(x_i) \tag{2.12}$$

where x_i is a sampled system state, N is the number of states, $H(X_i)$ is the output of the test function H for each sampled system state and $\hat{E}(H(X))$ is the value estimated for a given reliability index that is defined by H.

An example of a test function used to calculate LOLP would be:

$$H_{LOLP}(x_i) = \begin{cases} 1, & \text{if failure state.} \\ 0, & \text{if sucess state.} \end{cases} \tag{2.13}$$

2.6 Sequential Monte Carlo Simulation for Reliability Assessment

In sequential MCS the state duration sampling approach is used. In this approach, chronological component state transition processes are sampled for all components. The chronological system state is created by combination of the chronological components state transitions [22].

2.6.1 Component State

Considering the two state component Markov model (2.5) and assuming that the operation and repair state duration distribution functions of the components are exponential [22], the residence time in the success and failure states ("Up" and "Down", respectively) is given by:

$$T_{Up} = -\frac{1}{\lambda}\ln U_1 \tag{2.14}$$

$$T_{Down} = -\frac{1}{\mu}\ln U_2 \tag{2.15}$$

where T_{Up} is the residence time where component is in service and T_{Down} the residence time where is out of service. U_1 and U_2 are uniformly distributed random numbers between [0,1].

When simulation time achieves the last hour of the yearly simulation, sampling process ends, and an "Up-Down" cycle of the component is obtained like the one presented in Figure 2.7. A more detailed view of the algorithm developed for this phase is described on chapter 6.

2.6.2 Load Model

The load is modelled using a chronological representation that contains a load level for each hour of the year. The sequential MCS method follows chronologically these loads steps as the simulation progresses.

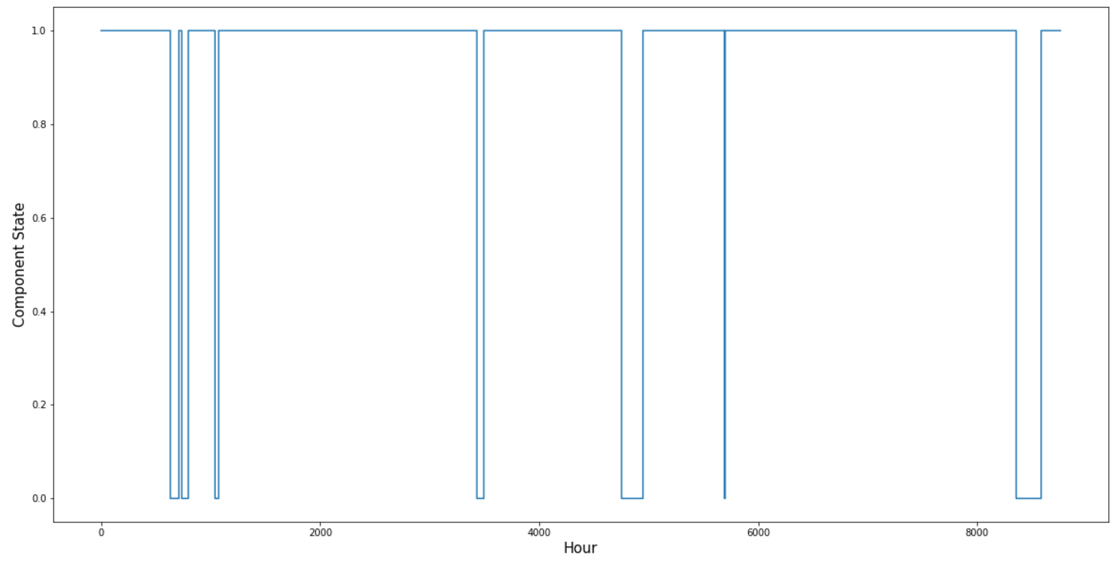


Figure 2.7: Example of component "Up-Down" cycle obtained at the end of sampling process

2.6.3 Indices estimation

The estimation of reliability indices in Sequential MCS is done by:

$$\hat{E}[H(X)] = \frac{1}{NY}\sum_{i=1}^{NY} H\left(x_n\right)_{n=1}^{S_i} \tag{2.16}$$

where x_n is the sequence of system states x over the period i and NY is the number of years simulated.

Similar to the non-sequential MCS, the test function H can take various forms depending on the intended reliability index. On sequential MCS it is possible to calculate duration and frequency indices (LOLD and LOLF).

LOLE and EENS will be the estimated indices in this thesis and their estimation follows equations 2.17 and 2.18, respectively.

$$LOLE = \frac{1}{NY}\sum_{n=1}^{NY}\sum_{i=1}^{NHY} H\left(X_{i_n}\right) \tag{2.17}$$

$$EENS = \frac{1}{NY}\sum_{n=1}^{NY}\sum_{i=1}^{NHY} H\left(X_{i_n}\right) \times \left(C_{i_n} - L_i\right) \tag{2.18}$$

Where $H(X_i)$ is equal to the one presented in 2.13 , C_i is the total capacity of state X_i and L_i is the total load.

2.7 Evaluation Stage

The evaluation stage is a crucial part of the MCS simulation. When composite reliability studies (HLII) are considered, it is the most complex part of the MCS process since it can include several algorithms.

On generation capacity assessment studies (HLI) this stage is simple because one just needs to verify if the generated capacity is sufficient to supply the load. If not, there is load curtailment, and its magnitude is calculated by the difference between the load demand and the available generation.

However, this thesis relates to the HLII reliability assessment. Therefore, the transmission system is also considered, meaning that the limits and reliability data of transmission components are held into account. In this case, the evaluation procedure can be divided into four steps. The first step is regarding the **detection of islands** in the system. Then the load demand of the system is assigned to the available generation units by a process called **generation dispatch**. Afterwards, a **Power Flow** is computed to assess if operational limits of the components are respected. If limits are violated, an **Optimal Power flow** is used to apply corrective measures and if it is the case, determine the minimum load curtailment.

Details regarding the algorithm used for this phase are presented in chapter 6.

Chapter 3

Acceleration of Monte Carlo Simulation for Reliability Assessment

Monte Carlo Simulation can be used to provide estimation of the reliability indices with a required level of precision and its simplicity makes it useful for extensive and complex power system reliability studies. However, when the system in question is very reliable these method requires a long time to converge, especially when higher levels of precision are required.

The MCS can be considered as threefold process as presented on Figure 3.1 where the vital stages are: State Sampling, State Evaluation and Index Calculation.

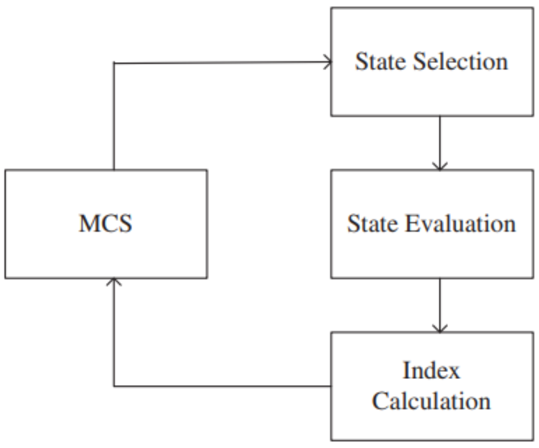


Figure 3.1: MCS stages [4]

Depending on the type of MCS process and the reliability studies category, these stages can take a considerable amount of time, affecting the overall time of the process. For instance, in sequential MCS, the state selection phase is time-consuming, and the evaluation phase becomes computationally intensive when reliability studies contemplate transmission systems.

Several techniques were developed with the purpose to reduce time burden of this stages, based on different concepts and generically can be be divided into two groups: Mathematical approaches and Pattern Recognition Techniques. The parallel implementation of MCS, constitutes also an approach to accelerate the MCS process. These techniques are briefly described in the following sections.

3.1 Mathematical approaches

Variance Reduction techniques are mathematical approaches that aim to reduce the variance between samples and consequently decrease the number of samples needed to complete the simulation while maintaining the same precision and without altering the expected value of the estimated index. Various techniques for variance reduction have been developed such as **Importance Sampling** [15], **control variates** [29] and **antithetic variates** [30]. In the following section, Importance Sampling (IS) and its application on **Cross Entropy Method** (CE) is described.

3.1.1 Importance Sampling

Importance sampling methodology is based on the distortion of the probability distribution of x, and consequently of $H(x)$ in order to raise the probability of relevant events (in this case events that lead to load curtailment).

As stated before, the probabilistic analysis of a power system can be seen as the expected value determination of a test function of the system H(x) such that:

$$\mathrm{E(H)} = \sum_{\mathrm{x \in X}} \mathrm{H(x) \cdot p(x)} \tag{3.1}$$

Where $\mathrm{p(x)}$ is the probabilistic distribution from where samples are obtained.

By multiplying and dividing the right member of the expected value in equation 3.1 by a new distribution function p^*:

$$E(H) = \sum_{x \in X} \frac{H(x)p(x)}{p^*(x)} p^*(x) = E\left(H^*\right) \tag{3.2}$$

It is possible to obtain a new function of test H^* given by:

$$H^*(x) = \frac{H(x)p(x)}{p^*(x)} \tag{3.3}$$

The expected value of both test functions remains the same but the variance of H^* is not necessarily the same. However, in reality, p^* is not known. Fortunately, if an approximate function $Z(x)$ is defined it is possible to derive an approximation p' to the optimal importance distribution p^*:

$$p'(x) = \frac{Z(x)p(x)}{E(Z)} \tag{3.4}$$

From which:

$$\mathrm{E(H)} = \sum_{\mathrm{x}\in\mathrm{X}} \frac{\mathrm{F(x)}}{\mathrm{Z(x)}}\mathrm{E(Z)p'(x)} \tag{3.5}$$

Then the expected value of H can be estimated by:

$$\hat{E}(H) = \frac{1}{N^*}\sum_{i=1}^{N^*} \frac{H\left(x^i\right)}{Z\left(x^i\right)}E(Z) \tag{3.6}$$

The vectors x^i are sampled according to $p'(x)$. The efficiency of this process depends on the relationship between $\frac{H(x)}{Z(x)}$, the knowledge kept on the function Z allows the reduction of variance of the Monte Carlo process.

3.1.2 Cross Entropy Method

The previous section showed the importance of the relationship between $\frac{H(x)}{Z(x)}$. Function $Z(x)$, which is an approximation of p' must be "close" to the optimal importance distribution p^* to achieve a substantial reduction in variance.

The cross-entropy (CE) method is an adaptive importance sampling procedure for the estimation of rare-event probabilities that uses the cross-entropy or Kullback–Leibler divergence as a measure of closeness between two sampling distributions and can be used for estimation and optimization [31] [32]. In this case, the parameters to be optimized are the system components *forced outage rates* and since the distribution of this variables belongs to the exponential family, the CE method can be applied [33]. The outcome of this method will be the distorted *for* of each component favouring the loss of load cases that can be used directly in Importance Sampling method [33] [34].

3.2 Parallel Implementation of Monte Carlo Simulation

This technique comprises the division of MCS process into smaller tasks that can be executed simultaneously (parallel processing).

The work developed in [26] and [35] takes advantage of parallel computing using a GPU to speed up routines of the MCS process. The sampling stage is time-consuming in MCS process because state sampling occurs in serial mode (one after another). Applying parallel computing can reduce significantly the time spent on this stage, especially when sequential sampling is required.

It is also possible to apply this technique to the evaluation stage of MCS since it comprises continuously repeated matrix operations. However, a detailed study is necessary to analyse the routines suitable for parallelisation. In HLI studies, parallelisation in the evaluation stage is easily achieved since it only requires a comparison between the available generation and load demand. On the other hand, in HLII studies, a complex algorithm composed of optimisation tools is necessary and can be hard to implement in a parallel way.

3.3 Pattern Recognition Techniques

Deep learning algorithms have found application in many areas, and power system reliability is no exception.

On [16] a methodology for reliability evaluation of composite generation and transmission systems, based on non sequential Monte Carlo Simulation and Polynomial Neural Networks (GMDH), is described. GMDH is trained and used to identify system states with no load curtailments (success states). This way, only the states not identified as success states are evaluated by the MCS evaluation phase, thus reducing the overall simulation time.

Similar to the approach developed in this thesis, a Convolutional Neural Network (CNN) was applied on [36] to the non-sequential MCS. The application of this deep learning method aims, like the GMDH, to classify a system state instantly, thus reducing the number of samples analysed by the MCS evaluation phase. Since CNN is generally used in image processing, organising the input data in a graphical image is necessary. This way, images of the system are formed, and CNN can reveal essential patterns to distinguish them between cases of load loss and cases where the system can respond to the load demand.

Chapter 4

Convolutional Neural Networks

4.1 Artificial Intelligence, Machine Learning and Deep Learning

Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) are three terms often perceived and used to refer the same concept. The easiest way to think of their relationship is to visualize them as concentric circles with artificial intelligence, which is the idea that came first, then machine learning and finally deep learning, which is driving today's AI explosion [5]. A visual map illustrating this relationship is presented on Figure 4.1.

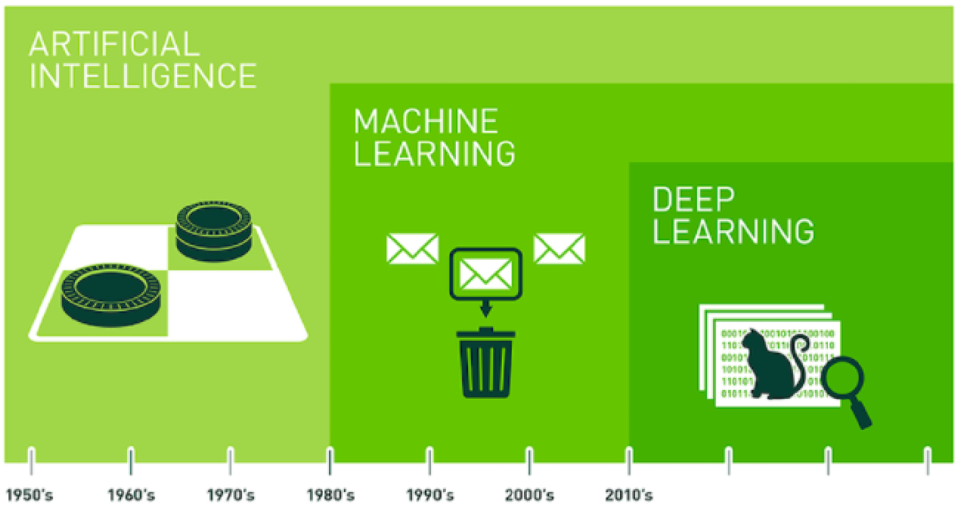


Figure 4.1: Relationship between AI, ML and DL and their evolution through years [5]

Artificial intelligence developments in these few decades are remarkable, and many applications are revolutionizing areas where solely humans could intervene like healthcare [37], finance [38] and engineering [39]. In conclusion, AI is a broad term reuniting all the techniques that aim to reproduce human intelligence.

Machine learning is a first attempt to achieve artificial intelligence by swapping the classical software routines, written for a specific purpose, with routines trained with many data giving the ability to the machine to learn and perform a task. Still, ML algorithms require a lot of hand-coding and often did not meet the expectations [5].

Deep learning came as a new way to implement machine learning, with algorithms and techniques less prone to error than his ancestor and automated learning. Artificial Neural Networks (ANNs) are the core of deep learning and are inspired by how the human brain processes information through neurons. Many deep learning models were developed on this basis, leading to the appearance of other methodologies like Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs), which are the core of this dissertation.

4.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a deep learning algorithm which can take an input image, assign importance (learn weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. While in machine learning methods, filters are hand-engineered, a CNN, with enough data and training, have the ability to learn these filters/characteristics.

The architecture of a CNN aims to replicate the connectivity pattern of neurons in human brain and is inspired by the organization of the visual cortex where individual neurons respond to stimulus only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

CNNs are characterized by two parts: Pattern Extraction and Classification. For Pattern Extraction, CNNs make use of two fundamental operations, Convolution and Pooling, and in Classification rely on a fully connected layer to make a correlation between the output neurons of the pattern extraction stage and the desired targets.

In Figure 4.2 a representation of an "artificial neuron" is presented, where x_i inputs are affected by weights, w_i, and a bias, b. The resulting product feeds a function called activation function to accentuate the neuron features. The most common activation functions are hyperbolic tangent, sigmoid, and ReLU (Rectified Linear Unit). The choice of a specific function depends on the nature of the problem and it is usually a process of "trial and error".

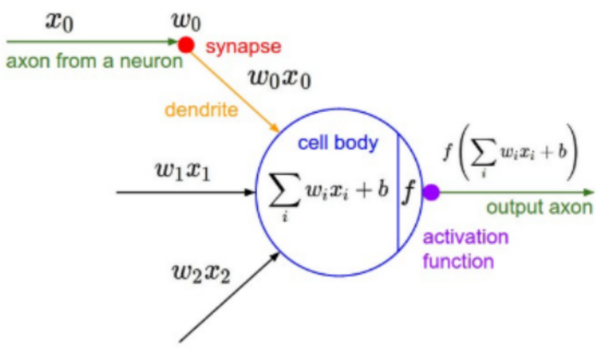


Figure 4.2: Visual representation of an artificial neuron used on neural networks[6]

The convolutional layer is the core building block of a CNN, based on convolution operation, it is where the majority of computation occurs. This layer processes input data through a filter, and produces features maps [7].

Consider the input data as gray scale image which can be interpreted as a matrix where each element of it represents a value of a pixel. The filter, also known as kernel, is a steady matrix of weights with a 2D shape and smaller than the input image. Filter is applied to an area of the input image, and convolution is calculated between the input pixels and the filter. Convolution operation is a dot product between the pixels values of the image and filter values.

Afterwards, the filter shifts its position, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products between input and the filter is known as a feature map [7].

The weights of the filter remain fixed as it moves across the input, this characteristic is also known as *parameter sharing*. As seen in Figure 4.3, feature map value (output array) does not connect to each pixel value in the input image, instead only connects where the kernel is being applied, this characteristic is known as *local connectivity* and it is why convolution layers are also called "partially connected” layers [7].

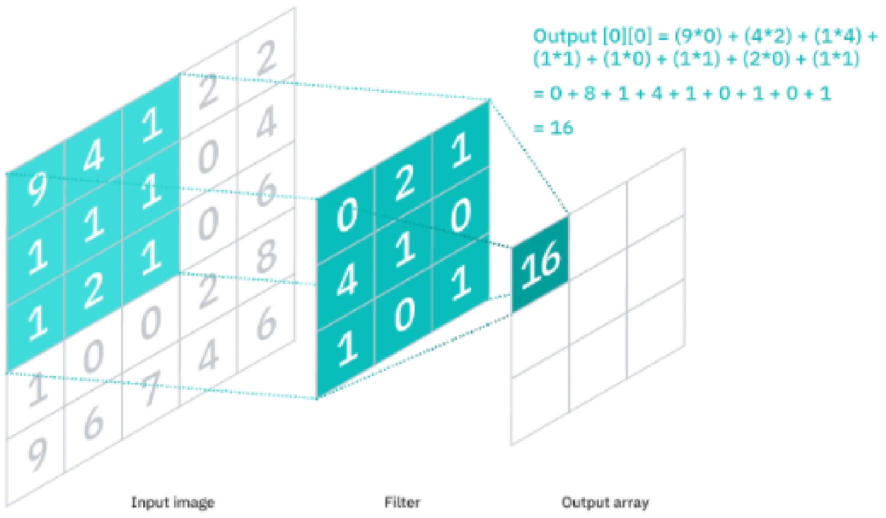


Figure 4.3: Representation of convolutional layer and a convolutional operation [7]

The pooling layer, also known as the downsampling layer, is responsible for dimension reduction of the convolutional layer’s previously obtained feature maps. A kernel swipes the feature map outputting the maximum value on the area selected by the kernel in pooling operation. This way, the most representative characteristics in feature maps are preserved while reducing the complexity of the input [7].

In a fully connected layer, in contrast with "partially connected" layers, each "neuron" in the output layer connects directly to a neuron in the previous layer. In convolution layers, it is common to use RelU as an activation function. In fully connected layers, softmax function is used to correctly classify the input, producing values between 0 and 1, so that they can be interpreted as probabilities [40]. It is also possible to use a sigmoid function as activation function in binary classification problems.

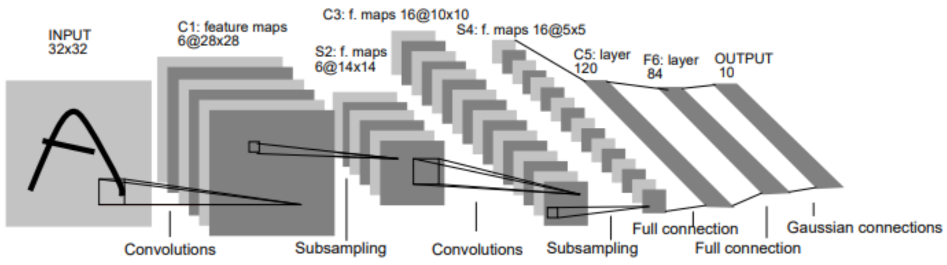


Figure 4.4: Example of an architectural Structure of a CNN (LeNet-5), where is possible to identify the principal layers: Convolutional, pooling and fully connected layer [8]

4.2.1 CNN applications on non image data and Power Systems

Convolutional Neural Networks are normally applied to image input, for instance the CNN represented on Figure 4.4, proposed by LeCun *et al.* [8], was used for handwritten and machine-printed character recognition. However there is no apparent limitation on the application of this technique on non image data, since this idea was already explored in distinct areas [41], inclusively on Power Systems.

In [42] an approach was developed to detect dynamic events in power systems by processing phasor measurement units (PMU) frequency data through a CNN. The work proved it is possible to represent a continuous variable, frequency, in an image to feed a CNN model and produce essential features used to characterize a specific event, like load shedding and generation tripping.

Another recent application of CNNs in power systems is the definition of a topology processor based on a CNN, capable of accurately determining the connectivity of a line in a certain point of the grid [43]. In this work, a method to organize power flow results in an image by applying information theory concepts was developed.

4.3 CNN's training on GPU

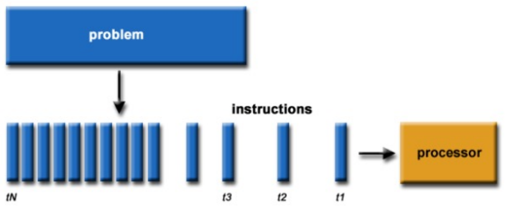
When working with deep learning models, the training phase is the most resource intensive task.

As explained in the previous section, CNN take as input an image, and this input is processed by the hidden layers through a operation called Convolution. This operation, at his core, is essentially matrix multiplication. CNNs, in general, require many filters because they aim to fully determine the patterns of an image, meaning that a significant amount of convolution processes are computed.

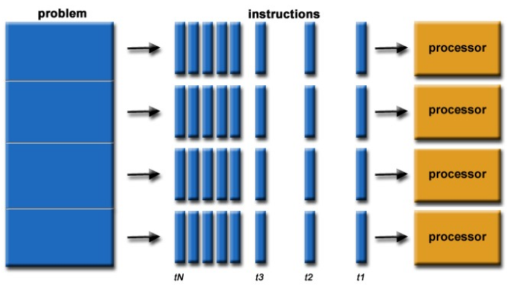
CPU (Central Processing Unit), is a core element of any computational device and focuses its smaller number of cores on individual tasks. By contrast, GPU (Graphic Processing Unit) breaks complex problems into thousands or millions of separate tasks and work them out at once [14], as illustrated on Figure 4.5 with parallel computing. A summary about both processors characteristics can be found in Table 4.1.

Table 4.1: CPU vs GPU [14]

CPU	GPU
Several cores	Many cores
Low latency	High throughput
Good for serial processing	Good for parallel processing
Can do a handful of operations at once	Can do thousands of operations at once



(a) Serial Programming (CPU)



(b) Parallel Programming (GPU)

Figure 4.5: Illustrative diagrams of serial programming and parallel programming [9]

The ability to run several processes simultaneously in all cores, such as mathematical operations like convolution makes GPU a well suited processor to speed up CNN training, especially when dealing with great amounts of data.

4.4 TensorFlow, Keras and Google Colab

For someone new to deep learning, developing CNN models from scratch in a short period may be challenging. Luckily there are frameworks like Tensorflow and Keras that simplify this process.

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive and flexible ecosystem of tools, libraries and other resources that provide workflows with high-level APIs [44].

Keras [45] is a high-level API, written in python, that runs on top of TensorFlow, specialized in deep learning models. Keras is intuitive to use, and because it offers "stock models", the user can focus on upgrading his "basic model" for the problem to solve and doing a more significant number of experiments. Also, models developed on this framework are "GPU-friendly", which means that, for a basic developed model, training can be executed on GPU without the need to enter in full detail about parallel computing. All this makes Keras the perfect platform choice to develop, in a fast and easy way, a CNN capable of solving the problem proposed in this thesis.

Google Colab [46] is a product from Google Research. Colab allows writing and executing arbitrary python code through the browser and is especially well suited to machine learning and data analysis. In this case, Google Colab was not used to execute python code but because it provides free resources like GPUs, although subject to time constraints. There are many types of GPUs available on Colab, and there is no way to choose what type of GPU one can connect to at any given time. However, most of the CNN's training was done in an Nvidia T4 GPU.

Chapter 5

Loss of Load Classifier based on CNN

This chapter addresses the methodology for a CNN classifier to identify load curtailment states.

The chapter starts with information about the test system, data set considerations, and data inputs to train the CNN. Secondly, a CNN's input structure study was conducted to decide which kind of "image" better organizes the data. Finally, CNN's training and architecture aspects are described, and the developed classifier evaluates a test set to collect insights about its performance.

5.1 Data Harvesting

5.1.1 Test System

In this dissertation, the IEEE Reliability Test System (IEEE RTS 79) [10] is considered for composite system reliability assessment and sampled data is collected regarding this system. The system has an annual peak load of 2850MW, the total installed generation is 3405 MW, and Figure 5.2 presents the system topology.

The generation system is composed of 14 plants (32 generating units) ranging from 12 to 400 MW. Complete information regarding unit's number, capacity and reliability data (MTTF, MTTR and f.o.r) is provided on Table 5.1.

Table 5.1: Generation System for IEEE RTS 79

Type	Capacity (MW)	N	MTTF	MTTR	f.o.r
Oil	12	5	2940	60	0.02
Oil	20	4	450	50	0.1
Hydro	50	6	1980	20	0.01
Coal	76	4	1860	40	0.02
Oil	100	3	1200	50	0.04
Coal	155	4	960	40	0.04
Oil	197	3	950	50	0.05
Coal	350	1	1150	100	0.08
Nuclear	400	2	1100	150	0.12

The transmission network consists of 24 bus locations connected by 38 lines and transformers, as shown in Figure 5.2. Table 5.2 presents information about the transmission network. Note that reliability data from the transmission system is not displayed, as, in this thesis, transmission components are considered in service all the time. Nonetheless, the operational limits of the network are taken into account (Rating).

Table 5.2: Transmission System for IEEE RTS 79

From bus	To Bus	X (p.u./ 100 MVA base)	Rating (MVA)	From bus	To Bus	X (p.u./ 100 MVA base)	Rating (MVA)
1	2	0.0139	175	12	13	0.0476	500
1	3	0.2112	175	12	23	0.0966	500
1	5	0.0845	175	13	23	0.0865	500
2	4	0.1267	175	14	26	0.0389	500
2	6	0.1920	175	15	16	0.0173	500
3	9	0.1190	175	15	21	0.0490	500
3	24	0.0839	400	15	21	0.0490	500
4	9	0.1037	175	15	24	0.0519	500
5	10	0.0883	175	16	17	0.0259	500
6	10	0.0605	175	16	19	0.0231	500
7	8	0.0614	175	17	18	0.0144	500
8	9	0.1651	175	17	22	0.1053	500
8	10	0.1651	175	18	21	0.0259	500
9	11	0.0839	400	18	21	0.0259	500
9	12	0.0839	400	19	20	0.0396	500
10	11	0.0839	400	19	20	0.0396	500
10	12	0.0839	400	20	23	0.0216	500
11	13	0.0476	500	20	23	0.0216	500
11	14	0.0418	500	21	22	0.0678	500

Figure 5.1 describes the annual load curve for 8736 hours of the year per MW basis. This model accounts for daily, weekly and seasonal patterns characterized in [10]. Training data considers this load curve, yet, the trained network will also be tested with an annual load curve constant at the peak of 2850 MW.

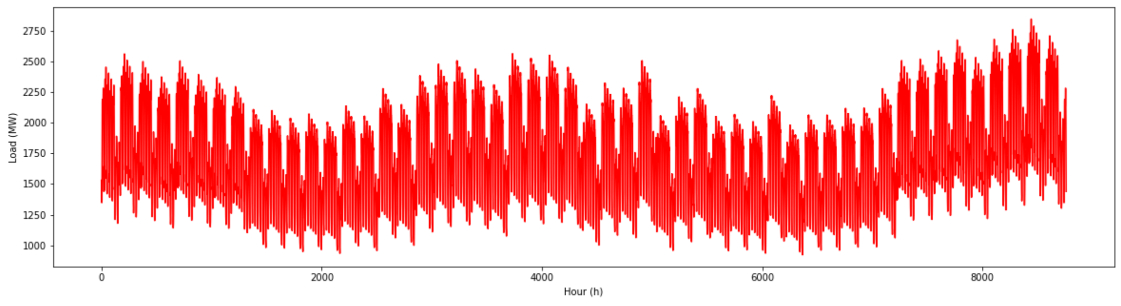


Figure 5.1: Annual Peak Load Variation Curve

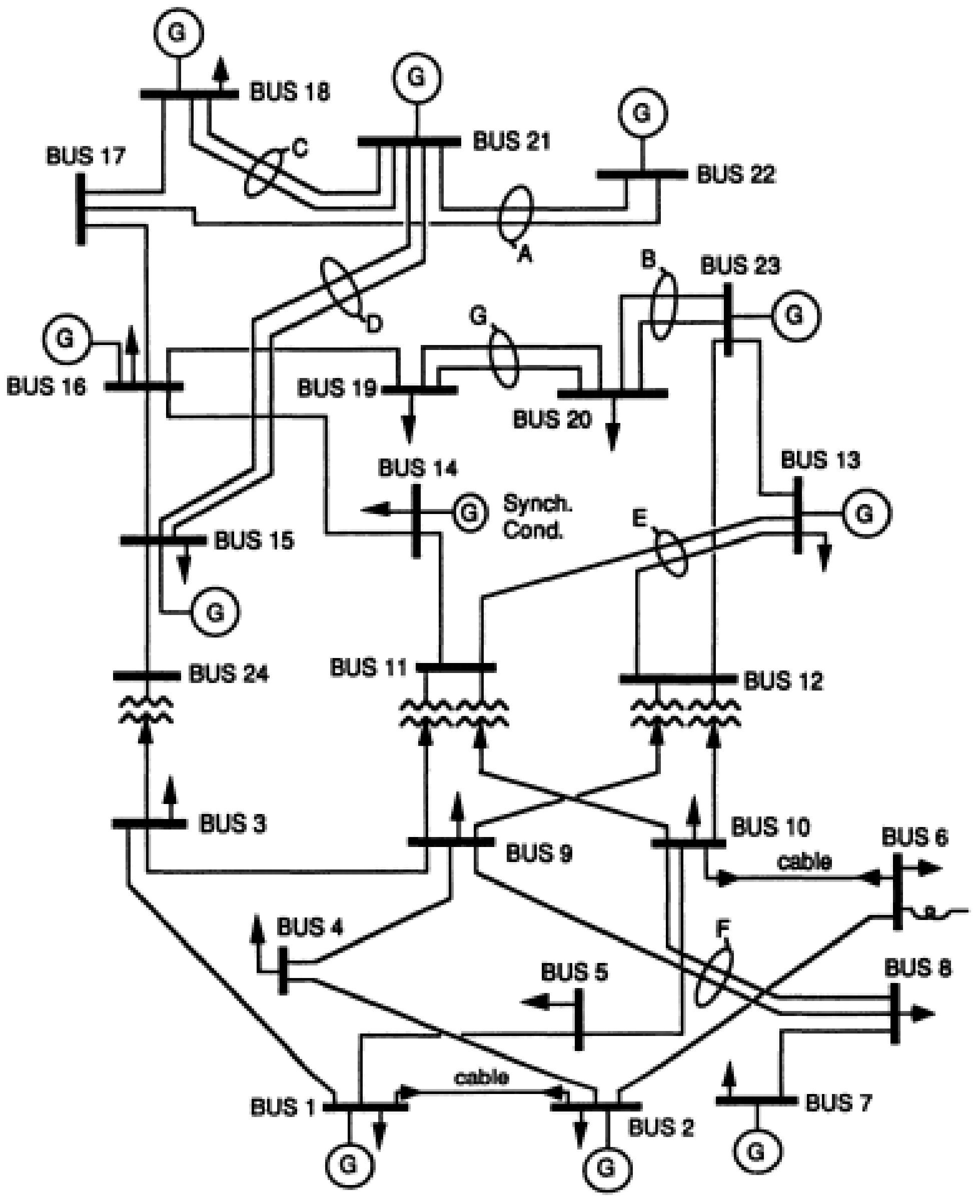


Figure 5.2: IEEE Reliability Test System [10]

5.1.2 Data Set Considerations

Deep learning methodologies commonly divide a data set into three groups: Train, Validation and Test [47]. Model trains on train set samples. The validation set evaluates the classifier performance and guides the choices and changes to various model hyperparameters like layers and filters. Finally, the test set samples evaluate the accuracy of the developed model, thus giving a glimpse of the model performance on its final use.

Typically, CNN's solutions are based on extensive image data sets, hence having plenty of information to train the model. However, when applying the loss of load identification task to CNN methodology, two problems are considered. One is related to the input structure. The data collected is purely numerical and does not take an image structure naturally. This issue is debated in the next section. The other is that data used to train and test this classifier does not simply exist on databases. Instead, it is generated for this sole purpose and given a specific test system.

A well-structured data set is usually needed to attain a good model performance, an equal amount of negative and positive samples, if the problem in hands is binary, like this case. Unfortunately, obtaining a data set for this specific problem requires computational effort since most power systems are highly reliable. Thus, it is harder to obtain samples representing a system operation that leads to loss of load.

The Cross-Entropy method tackles this barrier easing the sampling of loss of load cases. More information about the importance sampling algorithm and the method used to obtain the data set are presented in chapter 6.

The composition of each set was decided as:

- **Train set**: 7500 samples, where 2500 are failure cases (loss of load cases) and 5000 are success cases (no loss of load cases);
- **Validation set**: 1000 samples, equally divided into failure and success cases;
- **Test set**: 1000 samples, equally divided into failure and success cases;

Due to the high time consumed by the developed method to obtain loss of load samples, an unbalanced training set was settled for this task.

5.2 Data Preprocessing

Data is an essential aspect of a good CNN performance. However, in most cases, data does not come in an appropriate form to feed the model, and it becomes necessary to prepare it first. **Data Preprocessing** is a term used to refer to a combination of several steps that aim to treat data by making it more accessible for network interpretation.

These steps normally include:

- Data cleaning: to handle repetitions, missing values or noise;
- Data transformation: Normalization and Standardisation of data;
- Data reduction: deals with the dimensionality reduction of the data set, or attribute selection;

Data for training a deep learning model can be composed of many attributes, yet not all attributes are worthwhile to consider depending on the problem's nature. Given a problem, this raises the question: how could one find the relevant inputs that lead to a good model performance?

This dissertation focuses on the loss of load event identification on composite generation and transmission power systems. What variables can lead the model to identify a loss of load event? There are algorithms in the data preprocessing area, like Principal Component Analysis (PCA), [48], which aim to reduce the dimensionality of a data set, consequently identifying essential features for a data class. However, the so-called "engineering judgment" is a straightforward criterion for choosing the attributes to solve this problem.

The generated data contains many attributes that, when put together, can produce a proper pattern to distinguish between success and failure states, like system load, available generation reserve, circuit power flows and equipment outage rates. In this thesis, from the data available, the chosen attributes are the state of generation units (if they are in service or not), the total unavailable generation system capacity and the system load demand. Note that attributes related to the transmission system are not included because it is considered that transmission components are always available.

The inclusion of all system generation units states may be poorly informative for the present problem because the failure of higher capacity generators in the system will most likely lead to a loss of load event than the failure of the ones with lower capacity. Based on the *forced outage rate* vector obtained with the Cross-Entropy method, a heuristic was developed to check this reasoning and find which units are more important for this problem.

The absolute and relative variation between the original f.o.r's vector and the one obtained with the CE method are calculated for the proposed principle. Given a threshold, units with variations above that value are selected.

On Table 5.3, absolute and relative variation results are presented. Note that, the cross entropy method is a stochastic method for optimization hence the f.o.r obtained for units in the same group could not be exactly the same, however the results must be analysed relatively to a group of generators and not individually. To select the most important group of generators the threshold for absolute and relative variation is fixed at 0.05 and 100%, respectively. Consequently the state of 400, 350, 197 and 155 MW plants is chosen as attributes. By applying this technique, the attribute dimension space is reduced from 34 (32 generators state, load level and total unavailable system capacity) to 12 (10 generators state, load level and total unavailable system capacity).

Normalization is another crucial step in the data preprocessing stage. The data normalization is essential to achieve a good classification performance before the evaluation step since it aims to standardize the various types of variables to a common scale to make an equal contribution of each variable [49]. Two popular methods for data normalization are the Z-score and the Min-Max Normalization. The latter method is used in this dissertation to scale the values to the interval [0,1], and it is given by the expression:

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{5.1}$$

where x_{scaled} is the normalized variable, x the value of the variable, $\max(x)$ and $\min(x)$, are the maximum and minimum value that the variable can take, respectively.

Given the nature of the attributes, only the total unavailable capacity and load level are normalized since the attributes related to the state of the generators are categorical and represented as binary values, thus already inside the considered interval.

Table 5.3: Absolute and Relative Variation between the original f.o.r and the CE f.o.r (highlighted results represent the chosen plants)

Unit	Original f.o.r	CE f.o.r	Absolute Variation	Relative Variation (%)
12	0.02	0.02403	0.00403	20
12	0.02	0.02287	0.00287	14
12	0.02	0.02414	0.00414	21
12	0.02	0.02575	0.00575	29
12	0.02	0.02382	0.00382	19
20	0.1	0.11200	0.01200	20
20	0.1	0.11559	0.01559	16
20	0.1	0.12263	0.02263	23
20	0.1	0.12654	0.02654	27
50	0.01	0.01161	0.00161	16
50	0.01	0.01374	0.00374	37
50	0.01	0.01246	0.00246	25
50	0.01	0.01483	0.00483	48
50	0.01	0.01321	0.00321	32
50	0.01	0.01040	0.00040	4
76	0.02	0.02806	0.00806	40
76	0.02	0.03102	0.01102	55
76	0.02	0.03175	0.01175	59
76	0.02	0.02614	0.00614	31
100	0.04	0.05501	0.01501	38
100	0.04	0.06463	0.02463	62
100	0.04	0.05926	0.01926	48
155	0.04	0.08568	0.04568	114
155	0.04	0.09142	0.05142	129
155	0.04	0.09087	0.05087	127
155	0.04	0.09672	0.05672	142
197	0.05	0.16457	0.11457	229
197	0.05	0.16756	0.11756	235
197	0.05	0.15431	0.10431	209
350	0.08	0.28925	0.20925	262
400	0.12	0.52224	0.40224	335
400	0.12	0.52982	0.40982	342

5.3 CNN's Training Process

As mentioned in chapter 4, a CNN is composed of a feature extraction stage, comprised of convolutional layers, and a classification stage, where a fully connected layer gives meaning to the information retrieved from the earlier layers. The Loss of Load Classifier, as based on a CNN, follows the same structure.

On the training process of a deep learning model, hyperparameters like the batch size and the number of epochs are defined. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters, like node weights and filters are derived via training. The batch size, defines the number of samples to work through before updating the internal model parameters [50]. The number of epochs, defines the number times that the learning algorithm will work through the entire training dataset [50].

To understand the training process of this network is crucial to understand the concepts: Forward Pass, Back-propagation and Loss function.

The **Forward Pass** refers to the calculation of output values from the data inputs. When the forward pass process reaches the final layer of the CNN (the fully connected layer), the output, which is a result of the classification task, is compared to the true label of the input. A loss function calculates a prediction error.

The derivatives of the **loss function** are then used in the next step, **Back Propagation**, which is where all the learning of the network is done. The Back Propagation procedure uses a gradient descent algorithm to update the weights (in the fully connected layer) and the filters (in convolution layers). For each batch of data that feeds the network, a training step involving the forward and backward pass is completed, and the values of weights slowly converge towards a value that minimises the loss function, thereby producing the best quality predictions.

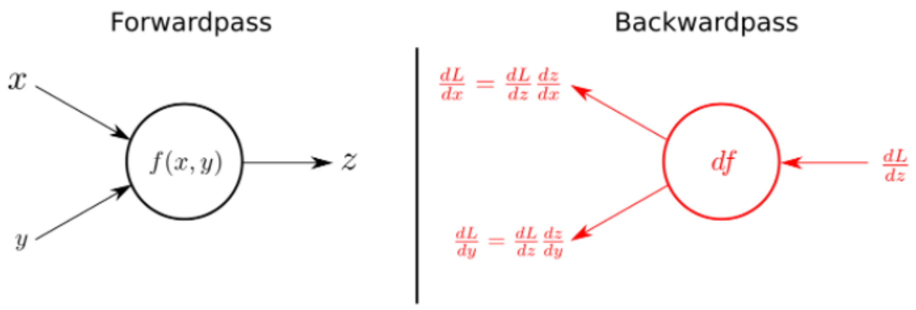

Forwardpass
x
y
$f(x,y)$
z
Backwardpass
$\frac{dL}{dx} = \frac{dL}{dz}\frac{dz}{dx}$
df
$\frac{dL}{dz}$
$\frac{dL}{dy} = \frac{dL}{dz}\frac{dz}{dy}$

Figure 5.3: Forward Pass and Back Propagation, essential processes on training deep learning models

It is also essential to refer to two learning strategies for deep learning algorithms: Supervised and Unsupervised Learning.

Supervised learning is defined by the use of labelled data pairs to train algorithms. It is supervised since the correct classification of data is known. The train approximates the mapping function, and when new inputs are fed to the network, an output value is formed accordingly. Supervised learning methods can be divided into two major groups: Classification, when the output variable is a category, such in this case, Loss of Load or not, or Regression when the output variable is a real value.

On the other hand, unsupervised learning uses unlabeled data. From that data, it discovers patterns that help solve clustering or association problems.

Although CNNs are usually seen as a supervised learning procedure, since the filters and weights are updated concerning labelled data, in this dissertation, the network's training will involve both methods mentioned above. The following sections will make more explicit how this train is done.

5.4 Classification Stage

As stated before, CNNs retrieve patterns from an image by a series of convolution and pooling operations. The patterns collected by convolutional layers does not offer a response to the classification problem by itself. It is necessary to give interpretation to the acquired patterns. Hence, adding a final hidden layer after the last convolutional layer enables the classification of the features retrieved into the desires categories of the problem.

This layer is called a Multilayer Perceptron - MLP, a feedforward neural network responsible for providing a nonlinear mapping between an input vector and a corresponding output vector [51]. In terms of architecture, it is composed of three types of layers: the input layer, hidden layer and output layer, as shown in Figure 5.4. In several MLP architectures, it is common to see more than one hidden layer when the number of inputs is extensive, enabling progressive downsampling of the values and better correlation between these values and the output, thus providing better results for complex problems. On the Loss of load Classifier, one hidden layer is understood as sufficient for the task since more does not improve the classification task and makes the model heavier on parameters to train.

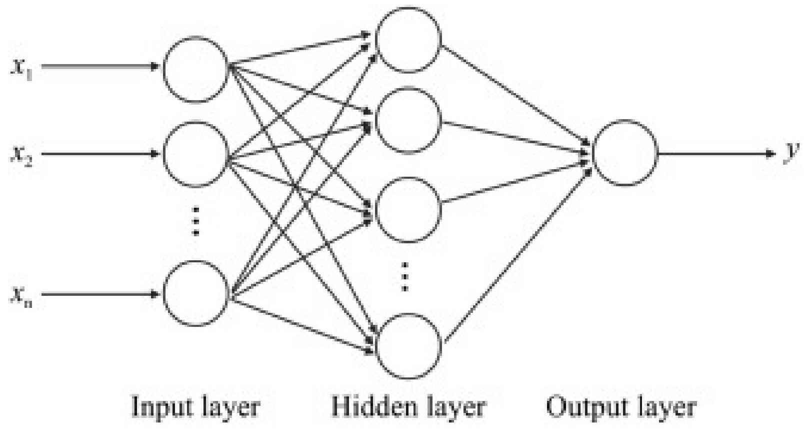


Figure 5.4: An example of a MLP with one hidden layer, similar to the one used on the Loss of Load Classifier [11]

Connections between the hidden layer and the output are essential to classify the reunited information. In this connections, Logistic Regression is applied.

Logistic regression uses the logistic sigmoid function (Figure 5.5) to return a probability value which, by defining a threshold, can then be mapped to two or more discrete classes. In this case, the problem is binary classification, so there are only two classes, failure (loss of load) or success(no loss of load). If $p(x) \geq 0.5$, where x are the values of the hidden layer and $p(x)$ the probability calculated by the Logistic Regression, then the value at the output node will be 1, corresponding to a case of failure. If $p(x) < 0.5$, the value at the output node will be 0, corresponding to a success case.

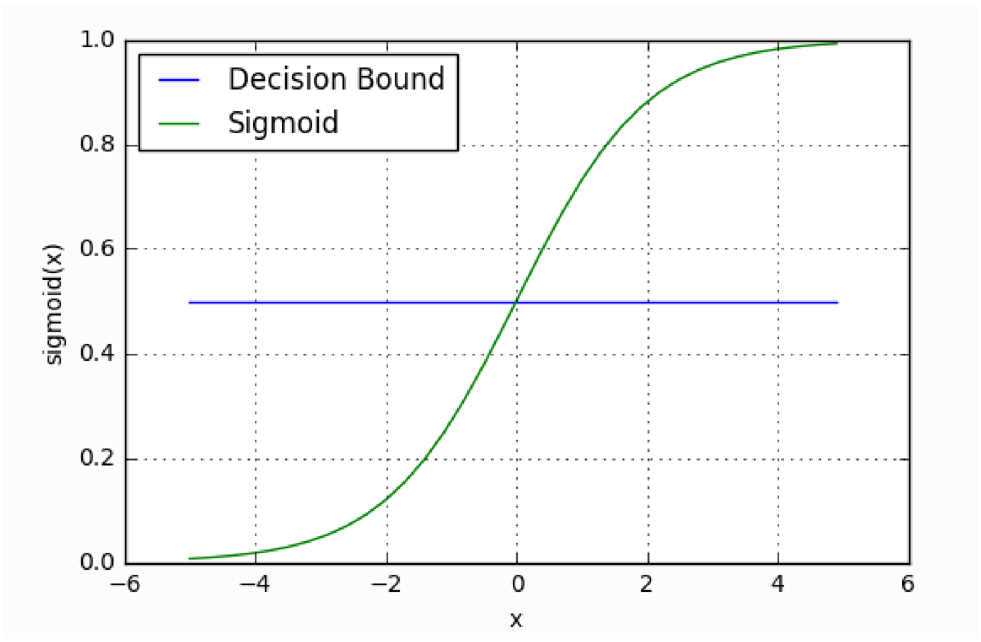


Figure 5.5: Logistic sigmoid function, p(x) is represented on vertical axis and x is the input values [12]

A loss function, also called cost function, is necessary to calculate the classification error. For binary classifiers, the typical function is the binary cross-entropy function [52]. Equation 5.2 represents the binary cross entropy loss function used in this problem.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - p\left(y_i\right)\right) \tag{5.2}$$

where y_i stands for the label assigned to the sample (1, for loss of load case or 0, for not loss of load case), $p(y_i)$ is the predicted probability of the sample being a case of load loss, N is is number of samples evaluated and $H_p(q)$ is the binary cross-entropy loss for the given set distribution.

After the error calculation by the loss/cost function, a gradient descent method updates the weights and biases of the layers. The gradient descent procedure applied to the weights (W) and biases (b) can be described by the following expressions:

$$\begin{cases} W_i = W_{i-1} - \alpha \cdot \nabla J_i(W,b) \\ b_i = b_{i-1} - \alpha \cdot \nabla J_i(W,b) \end{cases} \tag{5.3}$$

New values of the parameters are obtained concerning the gradient of the loss function. The convergence to minimum local error is done, iteratively, in the direction of the negative gradient $(-\nabla J_i(W,b))$. Figure 5.6 presents a visual representation of the gradient descent method. Although represented as single variable optimization, note that the optimization considers both variables, weights and biases.

In deep learning methodology is common to find various designations for the gradient descent

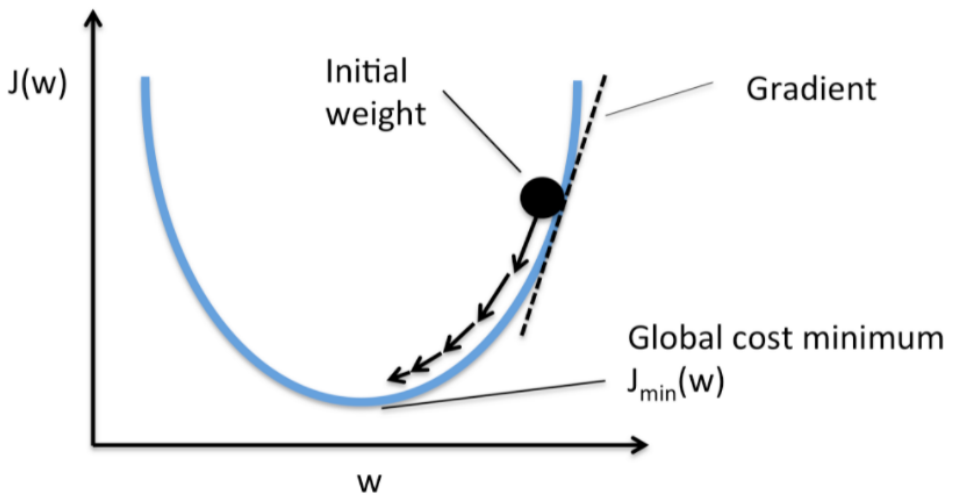


Figure 5.6: Gradient descent procedure concept applied to a cost function [13]

method depending on the timing of the weight update. For example, in "Gradient Descent (GD) Optimization", the weights are updated incrementally after each epoch (pass over the training dataset) [13]. Another method is the "Stochastic Gradient Descent (SGD)", where weights are updated after each training sample [13]. Finally, the "Mini-Batch Gradient Descent" is a middle term between GD and SGD, where weights are updated based on smaller groups of training samples called batch [13]. This method is used in the classifier training, and the batch dimension is settled to the value of 200 samples.

A crucial parameter that affects the gradient descent method is the learning rate α, as seen in equation 5.3. With a high learning rate value, it is possible to cover more ground with each step, but it may overshoot the lowest point of the loss function. Inversely, convergence is guaranteed with a low learning rate value, but it takes a long time to find the optimal point. Thus choosing a fixed learning rate can be difficult and requires a few tries to find the optimal value.

In response to this issue, adaptive learning gradient descent methods like ADAM were developed. ADAM [53] is a well-known algorithm in the area of deep learning, and it is based on the computation of individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients [53]. The training of the developed classifier uses ADAM, and the implementation present on Tensorflow/Keras framework eases his use. [54].

Figure 5.7 presents a summary of the training procedure of this stage. Training occurs in data batches composed of 200 input samples. First, the weights and biases of the layers are randomly initialised. Then, the process is done iteratively until the loss function minimum error is achieved. Note that this process only refers to the classification layer (fully connected layer) of the network. The training of convolutional layers is mentioned in the following section.

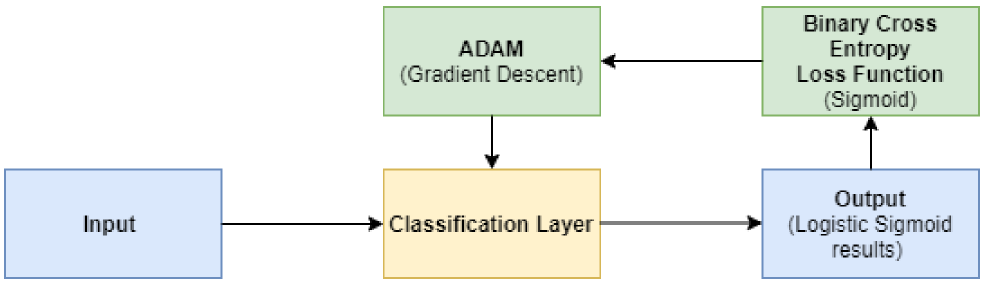


Figure 5.7: Diagram representing the iterative training process of the classification phase

5.5 Pattern Extraction Stage

This stage takes place before the events in the training of the classification phase. The convolutional layers are trained separately from the MLP to guarantee that they can retain important sectors of the image without the influence of the labelled data. The training process of this phase is similar to the previously described since a loss function and a gradient descent method is needed to update the filters. The difference is the absence of labelled data. Hence the objective of this training is to reconstruct the input image as presented in Figure 5.8.

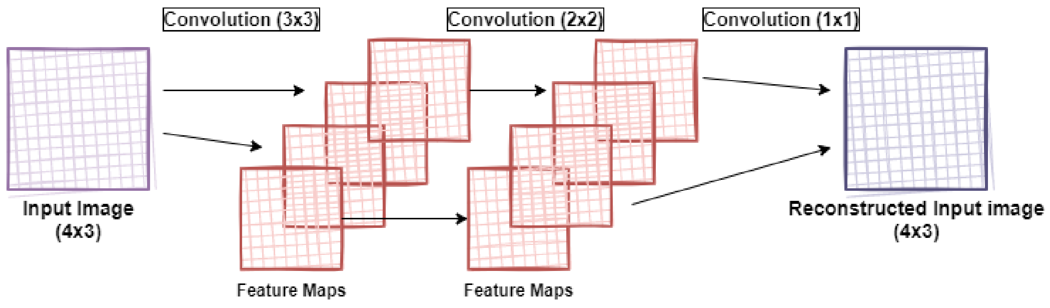


Figure 5.8: Convolutional layers training approach

As described in chapter 4, convolution operations are executed on the input image by convolution layers to gather feature maps. Several convolutional layers can be added on top of one another to retain more intricate patterns in the image. In this approach, feature maps are reunited to compose one image. Finally, this image is compared to the original image.

To compare the two images the Mean Square Error (5.4) function is used as a loss function:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - O(i,j)]^2 \tag{5.4}$$

where I is the input image and O is the output image. The difference between the images is computed by subtracting the pixel values of both images, squaring that value and finally sum them. To obtain the MSE, the sum of squares is divided by the total number of pixels in the image.

Like classification phase training, ADAM is used as the gradient descent method to update the convolutional layers filters. The training process 5.9 runs for 200 epochs, and in the end, filters values are kept to be used in the classification phase.

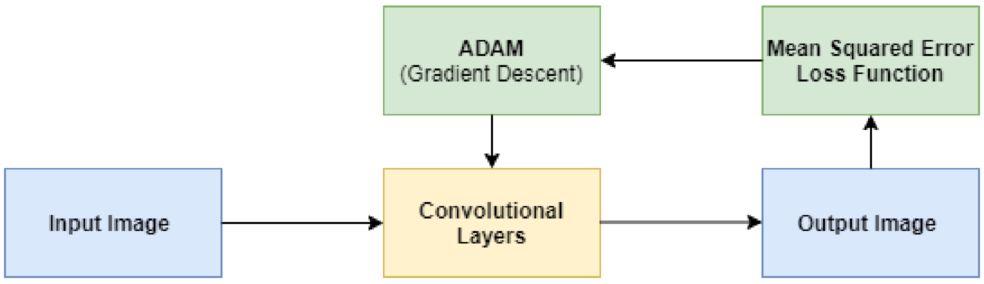


Figure 5.9: Diagram representing the iterative training process of the pattern extraction phase

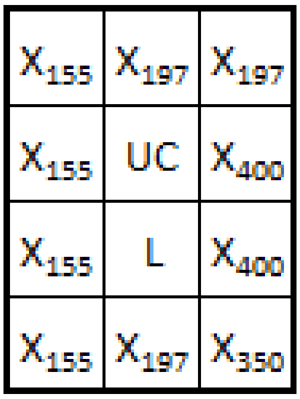
5.6 Input Structure and CNN Architecture

CNN accepts an image, which can be viewed as a matrix of size m x n, and through the convolution and pooling operations, it can extract patterns and do classification tasks. An image is a collection of pixels arranged in a specific way to represent, for instance, an object, where each pixel has a spatial relation to the neighbouring pixel. Hence, the utility of CNNs to interpret images since one of the advantages of this type of networks is detecting high order relationships and non-linear correlations between pixels.

In this case, the features selected are not naturally present in an image. Therefore one needs to find a way to organise the features in an image to feed the CNN. This task may be hard since there is no information *a priori* about the relationship between features and the loss of load events, only assumptions that can be made. Hence the only option to find out the "perfect image structure" to organise the features while taking advantage of the power of CNN to find relations between the variables is by constructing several structures and analyse how the network performs for every image.

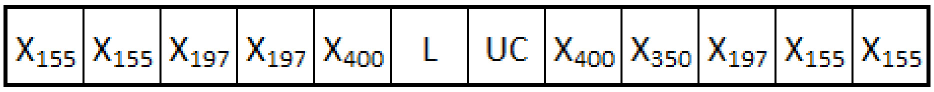
For the proposed study, two input structures were developed to place the 12 features. Figure 5.10 represents input structure A, a 4x3 matrix, where features related to load and unavailable capacity of the system are placed in the centre of the image and the features related to the state of the generation units are located in the surroundings of the image. The concept behind this structure is the belief that load and unavailable capacity are necessary inputs. Therefore, by putting them in the centre, the CNN can easily capture relations between the other variables.

Figure 5.11 represents input structure B, a 1x12 matrix, and contrary to matrix A, it is not possible to assign a robust spatial relation between the variables given the dimension nature, but the same principle is applied, load and unavailable generation feature are placed in the centre of the vector and the other features are arranged in the following "pixels".



X_{155}	X_{197}	X_{197}
X_{155}	UC	X_{400}
X_{155}	L	X_{400}
X_{155}	X_{197}	X_{350}

Figure 5.10: Representation of matrix A



X_{155}	X_{155}	X_{197}	X_{197}	X_{400}	L	UC	X_{400}	X_{350}	X_{197}	X_{155}	X_{155}

Figure 5.11: Representation of Matrix B

In both matrix representations, L stands for the load feature, UC for unavailable system capacity and X_{MW} is the state of the generation unit with the respective capacity.

With these two different structures, the idea is to study how feature positioning and the dimension of the matrix affects the pattern extraction and classification performance of CNN.

5.6.1 Models Performance Evaluation

For classification tasks, the terms true positive (tp), true negative (tn), false positive (fp), and false negative (fn) compare the results of the classifier under test with data labels. The terms positive and negative refer to the classifier's prediction (in this case, positive means a sample classified as a loss of load case and negative as not), and the terms true or false refer to whether that prediction corresponds to the data label or not. For the evaluation of developed models, the following metrics were considered: Accuracy, Recall and Specificity.

Accuracy (5.5) is a statistical metric used to measure the proportion of correct predictions (true positives and true negatives) among the total number of examined cases. This measure provides general insight into the classifier performance and takes values in the interval [0,1], 0 when the classifier provides only false predictions and 1 when the classifier correctly predicts all the test data labels.

$$\text{Accuracy} = \frac{tp+tn}{tp+tn+fp+fn} \tag{5.5}$$

Specificity and *Recall* are metrics needed to evaluate the performance of the classifier in the application context. In this dissertation, CNN is used in the Monte Carlo process to replace the necessity of examining all the states by a power flow analysis since only the cases classified as positive are analysed. Thus it is crucial to guarantee that CNN provides a good classification of positive cases, or in other words, minimises the number of false positives. Otherwise, instead of reducing the computational effort, pointless analyses will be computed.

It is also essential to assure a good classification of negative cases. False negatives affect how well reliability indices are estimated because the loss of load cases can be neglected.

Specificity (5.6) provides insight about the false positive rate. Ideally, a value of 1 is the desired result in the specificity metric since the negatives cases are all correctly identified and, consequently, there are no false positive cases.

$$\text{Specificity } = \frac{tn}{tn + fp} \tag{5.6}$$

On the other hand, Recall (5.7) provides awareness about the false negative rate. The maximum value is 1 when all the evaluated cases are correctly identified as positives. Therefore no false negatives exist.

$$\text{Recall } = \frac{tp}{tp + fn} \tag{5.7}$$

The architecture of a CNN is centred in the input image. To construct comparable models between both input structures, some aspects were considered:

- Models have two convolutional layers in the pattern extraction phase;
- Convolutional filters, although not equal in shape, have the same image area covered;
- Classification phase is composed by one hidden layer;

To study the filter influence, two groups of models (1 and 2) were created regarding the number of filters in convolutional layers: 64 and 8. Table 5.4 presents the architecture for models with 64 filters in each convolutional layer. Table 5.5 presents the architecture for models with 8 filters in each convolutional layer. In both tables, Model A receives, as input, the matrix A (4x3) and Model B receives the matrix B (1x12).

To evaluate the performance of the proposed CNN architectures there is one aspect to take into account: CNN models are stochastic. These models use randomness while being fit on a dataset, such as random initial weights and filters, random shuffling of data in each training epoch, and while applying the gradient descent method. This means that each time the same model fits the same training data, it may give different predictions on the test data and thus return different performance levels.

To overcome this barrier, a strategy was developed: for each model, convolutional layers are trained once, as described in the training section, and then with those filters the classifier phase of the network is trained on the same data 30 times. When a training round is finished, CNN makes

Table 5.4: Developed models with 64 filters in each convolutional layer

Layers	**Properties**	**Model A_1**	**Model B_1**
Input	Size	4x3	1x12
Convolutional	No. of Filters	64	64
	Filter Size	3x3	1x9
Convolutional	No. of Filters	64	64
	Filter Size	2x2	1x4
Pooling	Pool Size	2x2	1x2
Fully Connected	Input units	128	192
	Hidden units	64	64
	Activation function	ReLU	ReLU
Logistic Regression	Input Units	64	64
	Output Units	1	1
	Activation function	Sigmoid	Sigmoid

Table 5.5: Developed models with 8 filters in each convolutional layer

Layers	**Properties**	**Model A_2**	**Model B_2**
Input	Size	4x3	1x12
Convolutional	No. of Filters	8	8
	Filter Size	3x3	1x9
Convolutional	No. of Filters	8	8
	Filter Size	2x2	1x4
Pooling	Pool Size	2x2	1x2
Fully Connected	Input units	16	24
	Hidden units	8	8
	Activation function	ReLU	ReLU
Logistic Regression	Input Units	8	8
	Output Units	1	1
	Activation function	Sigmoid	Sigmoid

predictions on the test data set. Finally, a mean of the metrics values obtained for the test set is computed. The results are presented in Table 5.6 for each model.

Table 5.6: Classification performance for each model in the test set

Metric	**Model A_1**	**Model B_1**	**Model A_2**	**Model B_2**
Accuracy	0.992	0.988	0.987	0.958
Recall	0.9835	0.976	0.975	0.918
Specificity	0.999	0.999	0.999	0.999

Regarding the results in Table 5.6, Models A, with input matrix 4x3, have a better accuracy performance than the Models B, with input matrix 1x12, even with less filters, concluding that an input where features are arranged in a 2D (two dimensions) shape its more suitable for this problem than arranging the features in a 1D (one dimension) shape. Organizing features in a 2D shape makes it easier for the CNN to discover patterns that lead to a better classification since the "pixels" in the centre of the image share their border with more "pixels" than when arranged in a 1D shape, where at the most, just share their border with the immediate neighbour "pixel".

Every filter collected describes a pattern discovered in the image, so it is expected that when the number of filters increases, the accuracy also increases because more patterns are available for the network to consider in the classification phase. However, in studies developed with a number of filters higher than 64, no improvement in the classifier's performance was found for the two input structures, stating that a more complex model with more parameters does not provide a better classification.

One point to be noticed is the high value on specificity test achieved by all models developed, proving that variables chosen to construct both input structures provide sufficient information to identify loss of load cases correctly. On the other hand, recall results seem to vary with the number of filters and input arrangement, demonstrating that learning a pattern for cases with no loss of load is more challenging with the available variables.

Results of Table 5.6 were computed on a test set with 500 failure samples and 500 success samples. In Monte Carlo Simulation, where the CNN will be applied, the number of samples analysed is much higher and less balanced than this test data set. However, based on these results is possible to choose the architecture that will likely have a better performance when applied to the MCS. In light of the results presented, the chosen CNN architecture to apply in the Monte Carlo Simulation is the model A_1 with input structure 4x3.

5.7 Final Remarks

This chapter presented a deep learning methodology to evaluate operational system states, the Loss of Load Classifier based on CNN. In the performed tests, the developed classifier proved to be an efficient tool to distinguish between failure (loss of load) and success cases. Besides the accuracy of this classifier, the training time also affects the relevance of this solution. Fortunately, using a GPU for network training reduces the time spent on this task, making the classifier quickly available for its final use.

Although the purpose of this classifier is to be integrated into a Monte Carlo Process, the CNN should be seen as a by-product of the developed work since there is a range of applications for this model besides the MCS, for instance, as an auxiliary tool for engineers on dispatch centres.

Chapter 6

Methodology for Monte Carlo Simulation Composite System Reliability Assessment based on CNN

In this chapter, the methodology used in this dissertation is presented.

First, the classic versions of the MCS are reviewed: the Non-Sequential and Sequential Simulation. The algorithm for the evaluation stage of MCS is described. Finally, the MCS-CNN proposed method is addressed, and comparisons are made with the classic version. It should be mentioned that classic versions of MCS were guided by the work developed in [26] [35] [23].

6.1 Non-Sequential Monte Carlo Simulation

Algorithm 1: Non-Sequential Monte Carlo for Composite Reliability Assessment

```
Input :
$P_{Gen}$ as the capacity of each system generator;
$for$ of each generator;
$\beta$ as the confidence interval coefficient;
Peak load variation curve;
$max_{it}$ as as the maximum number of iterations;
while $\beta^2 < \frac{V(X)}{N \times LOLP^2}$ $or$ $k < \max_{it}$ do
    Sample vector of states regarding generation units;
    Sample vector of states regarding transmission system components;
    Sample load value from the yearly peak load variation curve;
    Evaluate if it is a case of loss of load or not ;
    Estimate $L\hat{O}LP = avg(X)$;
    Compute the variance V(X) between the sampled states ;
Output: $L\hat{O}LP$
```

The algorithm of the NSMCS process for composite system adequacy is described on Algorithm 1. As explained in Chapter 2, the main objective is to estimate a reliability index, in this case LOLP, within a confidence interval bounded by β, selected *a priori*.

Algorithm 2: Sampling the state of generation units

```
Input  : $for$ of each generator
while The state of every generator i have not been defined do
    Sort a number (u), according to the uniform distribution, between 0 and 1;
    if $for_i > u$ then
        State of generator i = 0;
    else
        State of generator i = 1;
Output: Vector of states regarding generation system units
```

The first task in the simulation is the components state sampling, and it is done according to Algorithm 2. When the sorted number u is smaller than the generator's for, the generator is considered out of service or in his downstate (0). Otherwise, the generator is in service or in his upstate (1). In this case, only the generation units states are sampled because it is considered that components from the transmission system are always available.

The sampling of the load value from the annual peak load variation curve is done by sorting an integer number from a "discrete uniform" distribution in the interval 0 to 8736. Then the load value stored in the index corresponding to the sorted number is retrieved and used in the next task.

The next step is the loss of load evaluation using the state of system components and the load value. The methodology used for this evaluation is explained later in this chapter.

6.2 Sequential Monte Carlo Simulation

The algorithm of SMCS process for composite system adequacy is described in Algorithm 3.

The crucial difference between NSMCS and SMCS is that, in the latter, samples are generated across a year, thus having a chronological relation among them. This particularity allows the calculation of reliability frequency and duration indexes. In this case, LOLE and EENS are estimated, within a confidence interval bounded by β, selected *a priori*.

The state of system components is sampled year by year, making possible to construct the component "Up-Down" cycle which is a record of the component state in every hour of the year.

Algorithm 4 shows the steps to obtain this record. In the beginning of the simulation, the initial state of each generator is drawn based on a comparison between a random sorted number u and generator's f.o.r. Then the rest of the simulation is done sequentially, by modelling service and failure periods according to expressions presented in Chapter 2.

Similar to the NSMC process, the "Up-Down" cycle is generated only for the generation units, transmission components are considered available all hours of the year.

On SMCS is not necessary to sample the load level since it takes as input the chronological annual load curve.

The evaluation stage occurs in the same manner as in the NSMCS process. The difference is that evaluation is done hour by hour until the whole year is evaluated.

Algorithm 3: Sequential Monte Carlo for Composite Reliability Assessment

```
Input :
$P_{Gen}$ as the capacity of each system generator;
$MTTF$ of each system generator;
$MTTR$ of each system generator ;
$\beta$ as the confidence interval coefficient;
Hourly Peak load variation curve;
$max_{it}$ as as the maximum number of iterations;
while $\beta^2 < \frac{V(X)}{N \times LOLP^2}$ $or$ $k < \max_{it}$ do
    Sample yearly "life line" of every system generator;
    Sample yearly "life line" of every transmission system component;
    i=0;
    $LOLE_k$=0;
    $EENS_k$=0:
    while $i$< $8760$ do
        Evaluate if there is loss of load; if $LossofLoad = True$ then
            Calculate magnitude of the load loss;
            $EENS_k$= $EENS_k$ + Magnitude of the load loss (in MW);
            $LOLE_k$= $LOLE_k$ + 1;
        Estimate $L\hat{O}LE = avg(LOLE_k)$
        Estimate $EE\hat{N}S = avg(EENS_k)$
        Compute the variance V(X) between the sampled years;
Output: $L\hat{O}LE$, $EE\hat{N}S$
```

6.3 Evaluation Stage

This section presents a description of the algorithms used for the evaluation stage of composite system reliability assessment.

6.3.1 Transmission Grid Configuration

The objective of the transmission grid configuration is to identify the electric islands and the nodes/branches they enclose [23]. The grid is assumed to be constituted by electric nodes, buses, and branches that are transmission circuits. Each branch connects to two different nodes. This method is described in Algorithm 5.

Initially, every bus in the list of system buses is associated with an indicator, that can be the ordinal number which they appear in the list. This indicators are saved to a vector F and replicated to an auxiliary vector F'.

Algorithm 4: Sampling the yearly state of generation units ("Up-Down" cycle)

```
Input  :
$\lambda$ of each system generator, obtained with $MTTF$;
$\mu$ of each system generator, obtained with $MTTR$;
$f.o.r$ of each system generator;
k=0;
while The yearly state of every generator have not been defined do
    k=k+1;
    Initialize a vector of zeros, size 8736, to allocate the generator state for every hour ($Lifeline_k$);
    Sort a number, $u$, between 0 and 1, according to the uniform distribution;
    if $u>f.o.r$ then
        while $i<$ 8736 do
            Sort a number, $u_1$, between 0 and 1, according to the uniform distribution;
            Calculate the time in service by : $T_{Up} = -\frac{1}{\lambda} \ln u_1$;
            $Lifeline_k$ [i, i+ $T_{Up}$] =1;
            i= i+ $T_{Up}$
            Sort a number, $u_2$, between 0 and 1, according to the uniform distribution;
            Calculate the time out of service by : $T_{Down} = -\frac{1}{\mu} \ln u_2$;
            i= i+ $T_{Down}$
    else
        while $i<$ 8736 do
            Sort a number, $u_1$, between 0 and 1, according to the uniform distribution;
            Calculate the time out of service by : $T_{Down} = -\frac{1}{\mu} \ln u_1$;
            i= i+ $T_{Down}$
            Sort a number, $u_2$, between 0 and 1, according to the uniform distribution;
            Calculate the time in service by : $T_{Up} = -\frac{1}{\lambda} \ln u_2$;
            $Lifeline_k$ [i, i+ $T_{Up}$] =1;
            i= i+ $T_{Up}$
Output: Yearly state of every system generator
```

While searching the list of branches in system, the origin node (i) and the end node (j) are checked and if that branch is in service, the respective bus indicators in vector F are updated to the same value (min(F_i, F_j)).

When all branches are covered, Vector F and F' are compared and if vectors values are different, meaning that configuration evaluation is not concluded, vector F values are copied to F' and the search restarts.

When vector values are equal, configuration evaluation is finished and the number of isles in the system is obtained by counting the number of different indicators in F, since the nodes belonging to the same electric isle have the same indicator [23] [55].

Algorithm 5: Transmission Network Configuration

```
Input :
Initialize Vector F, size number of buses in the system, by assigning each entry i an
  indicator, according to $F_i = i$ ;
Initialize Vector F', equal to vector F;
while True do
    while All the branches are not checked do
        if $Branch_{ij}$ is in service then
            $F_i = F_j = minF_i, F_j$
    if F equal to F' then
        break;
    else
        F'= F
Number of isles in the system = number of different indicators in F;
Output: Number of isles in the system
```

6.3.2 Generation Dispatch

Generation Dispatch process aims to distribute load demand through the available generation units considering restrictions like the minimum and maximum power output of each unit [23]. There are two ways to do this distribution, a merit order or a proportional strategy.

Firstly, load is assigned to every generator with respect to the minimum power output of each unit. The remaining load is then allocated using a proportional strategy based on expression 6.1:

$$P_i = \frac{\bar{P}_i - \underline{P}_i}{\sum_{j=1}^{NG} (\bar{P}_j - \underline{P}_j)} \times \left(L - \sum_{j=1}^{NG} \underline{P}_j \right) + \underline{P}_i \tag{6.1}$$

Where P_i is the production of the generation unit i, $\bar{P}_i$ is the maximum capacity of the unit, $\underline{P}_i$ is the minimum capacity of the unit, NG is the number of generating units available and L is the load demand.

6.3.3 DC Power Flow

A Power Flow analysis is run to check if the components from the transmission grid are operating outside their operational limits. However, the complete form (AC power flow) problem is complex to solve given the non linearity of expressions and the restrictions to be taken into account. Luckily a linearised method (DC power flow) can be adopted, reducing the computational effort but only analysing the impact of active power on the adequacy of the system. Considering these method some assumptions are made [56]:

- No transmission losses;
- Resistance and admittance from the line are negligible ($R \cong 0$ and $Y \cong 0$);
- $\sin\theta_{ik} \approx \theta_{ik}$ supposing small $\theta_{ik} = \theta_i - \theta_k$, and i and k are contiguous nodes;
- Voltage Nodes is considered close to the nominal value ($V_i \approx 1p.u$)

The DC Power flow is based on the solution of the following system of linear equations:

$$\mathbf{P}_{inj} = \mathbf{P}_G - \mathbf{P}_L = \mathbf{B}'\theta \tag{6.2}$$

where $\mathbf{P}_{inj}$ is a vector of the active power injected at each bus, $\mathbf{P}_G$ and $\mathbf{P}_L$, respectively are the vectors of active power produced and consumed at each bus , $\mathbf{B}'$ is the susceptance matrix and θ is the vector of bus voltage angles.

The elements of the susceptance matrix are calculated by:

$$B'_{ik} = -\frac{1}{x_{ik}}, i \neq k \tag{6.3}$$

$$B'_{ii} = -\sum_{i\neq k}^{NB} \frac{1}{x_{ik}}, i = k \tag{6.4}$$

where NB is the number of buses and x_{ik} the reactance of the branch connecting bus i to k.

The equation system provided by 6.2 is undetermined, but by fixing a system bus as reference and zeroing that bus voltage angle, one equation can be removed resulting in

$$\hat{\mathbf{P}}_G - \hat{\mathbf{P}}_L = \hat{\mathbf{B}}'\hat{\theta} \tag{6.5}$$

And the active power flow through the branch connecting bus i to k is obtained by:

$$P_{ik} = \frac{\theta_i - \theta_k}{x_{ik}}. \tag{6.6}$$

In this dissertation, instead of programming this process from scratch, the framework PYPOWER, already developed for this purpose, is used. PYPOWER [57] is a Power Flow and Optimal Power Flow (OPF) solver to be applied with python programing language.

6.3.4 Optimal DC Power Flow

If transmission components are operating outside their operational limits, a DC OPF is computed. This process aims to determine the best operating strategy by adjusting the active power injected at the buses while keeping into account components functional limits. This optimization is done with the objective to minimize the total load curtailment, if is not possible to respond to the total load demand [23].

The DC optimal power flow problem can be formulated by:

$$z = min\sum_i M_i P_{FGi} \tag{6.7}$$

Where M_i is a constant that reflects the load curtailment priority at bus i and P_{FGi} is the active power produced by the fictitious generation unit in bus i that count for the load curtailment.

Subject to:

$$\mathbf{0} \le \mathbf{P}_{GF} \le \mathbf{P}_L \tag{6.8}$$

$$\underline{\mathbf{P}_G} \le \mathbf{P}_G \le \overline{\mathbf{P}_G} \tag{6.9}$$

$$\underline{\mathbf{P}_{ik}} \le \Gamma\mathbf{Z}\theta \le \overline{\mathbf{P}_{ik}} \tag{6.10}$$

$$\mathbf{P}_G + \mathbf{P}_{GF} - \mathbf{P}_L + \mathbf{B}'\theta = \mathbf{0} \tag{6.11}$$

Where P_G the active power produced by generation units, P_L is the system load demand, Γ is a diagonal matrix whose values are the branches admittance and $\mathbf{Z}$ is the system incidence matrix.

The method selected to solve this optimization problem was the Interior Point method [58], although other approaches could also be considered like the Simplex Method.

After defining the objective function, the decision variables and constrains, the interior point method was applied with the SciPY [59] optimization framework.

6.4 The Monte Carlo-CNN Simulation for Composite System Reliability Assessment

This section describes the methodology for including the CNN in the MCS process for composite system reliability assessment. The stages that make part of this method are presented in Figure 6.1.

The training phase of CNN has already been explained in chapter 5, the sampling stage and the application of CNN in MCS will be the focus in this section.

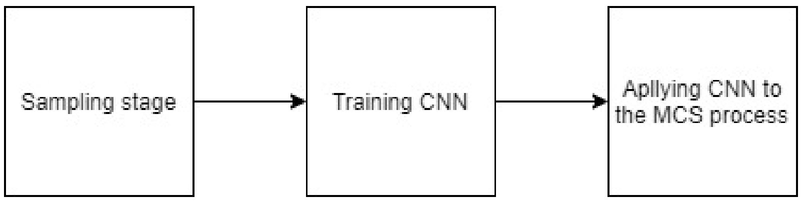


Figure 6.1: Stages of the proposed method

6.4.1 Sampling stage

The sampling stage is an important component in this application since the quality of CNN's classification depends on the quality of the samples retrieved and used for training. The routine of sampling stage is presented on Figure 6.2 and makes use of the functions already defined for the evaluation stage of MCS. Instead depending on a variance coefficient to end the sampling, the number of cases with and without loss of load to be obtained are fixed *a priori*.

Through the Cross Entropy method, Importance Sampling is applied, thus reducing the computational effort needed to sample cases with loss of load.

A test was conducted to evaluate the impact of this method on collecting the loss of load samples. Considering the original f.o.r and the f.o.r obtained with CE, the sampling stage routine was computed to obtain 1000 operational samples. With the original f.o.r, it was not possible to obtain loss of load samples, and with the CE f.o.r, 47 loss of load samples were collected. The CE method has fulfilled its purpose. However, the 47/1000 ratio demonstrates that the computational effort is still a drawback to obtain a more extensive set of loss of load samples, like the one used to train the CNN classifier (2500 samples).

The Cross Entropy algorithm used for this method is described in Algorithm 6 [26].

To obtain a new set of forced outage rates for each generator, the original for , the units maximum capacity and the system load are needed. Also the definition of some algorithm variables are necessary as the sample size N for the optimisation process (in this case settled as 10000 samples), the multilevel parameter ρ (typically between 0.01 and 0.1), the smoothing parameter, α used to avoid the occurrence of 1's and 0's in the new vector of f.o.r's, and the maximum number of iterations.

The algorithm starts by assigning the original f.o.r vector to a vector created with the purpose to keep the new f.o.r values. The original f.o.r vector is used to sample a vector of states with size N, defined *a priori*. Then the available generation is computed by equation 6.12.

$$S(X_i) = \sum_{j=1}^{N} X_{ij} \times C_j \tag{6.12}$$

Where X_{ij} is the state of the generator j in the hour i and C_j is the maximum capacity of the generator. The evaluated states, $S_{[i]}$, are ordered in descending order. The value of $\hat{L}_k$ is actualized as $S[(1-\rho)N]$, if it is greater than L, if not, $\hat{L}_k = L$.

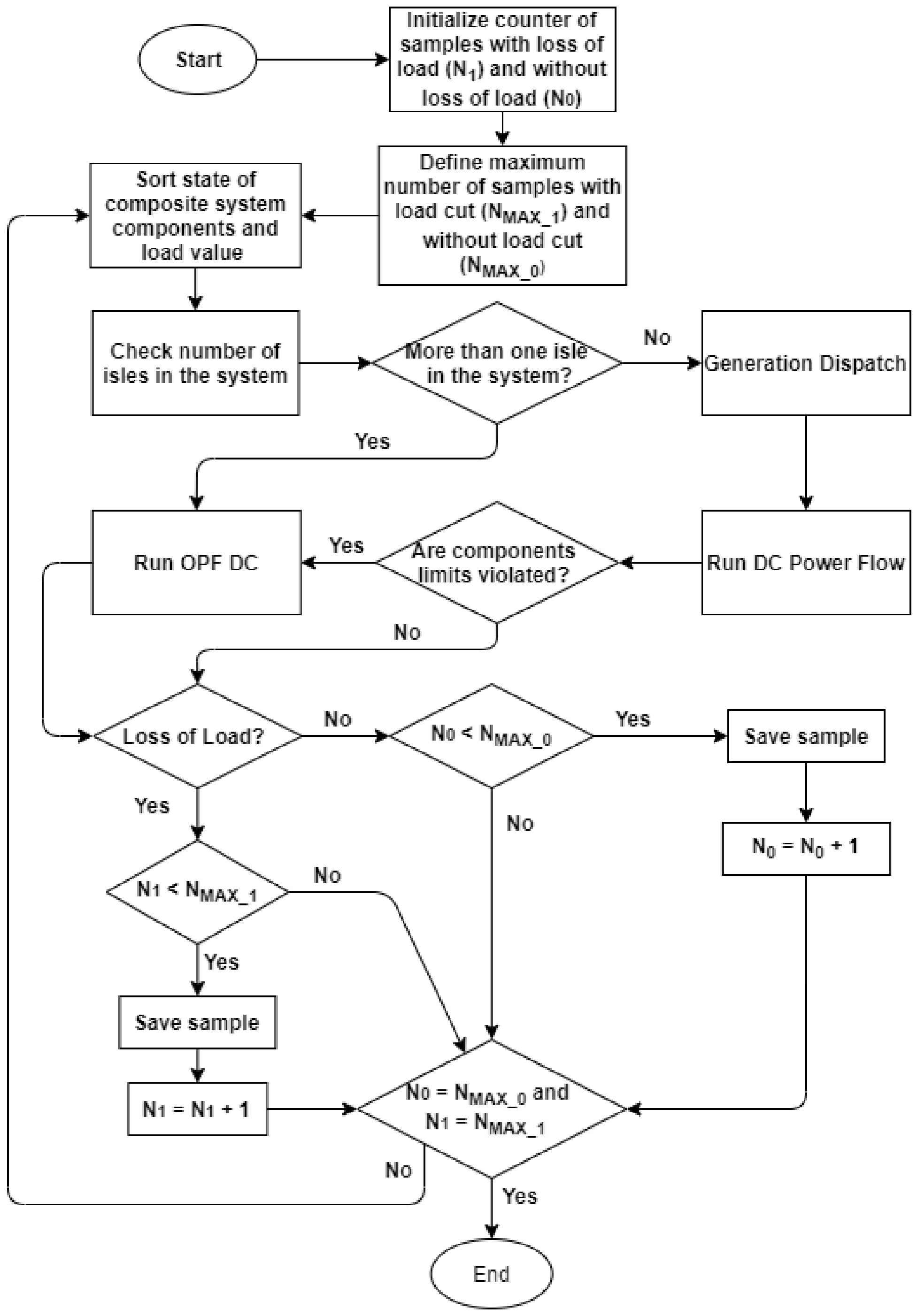

Start
Initialize counter of samples with loss of load (N1) and without loss of load (N0)
Define maximum number of samples with load cut (NMAX_1) and without load cut (NMAX_0)
Sort state of composite system components and load value
Check number of isles in the system
More than one isle in the system?
No
Generation Dispatch
Yes
Run DC Power Flow
Are components limits violated?
Yes
Run OPF DC
No
Loss of Load?
No
N0 < NMAX_0
Yes
Save sample
N0 = N0 + 1
No
Yes
N1 < NMAX_1
No
Yes
Save sample
N1 = N1 + 1
N0 = NMAX_0 and N1 = NMAX_1
No
Yes
End

Figure 6.2: Sampling State routine

The next step is evaluating the test function for LOLE index according to equation 6.13:

$$H(X_i) = \begin{cases} 1 \text{ if } S(X_i) < \hat{L}_k \\ 0 \text{ if } S(X_i) \geq \hat{L}_k \end{cases} \tag{6.13}$$

Algorithm 6: Cross Entropy Method

```
Input  :
$P_{Gen}$ as the max capacity of each system generator;
$f.o.r$ of each generator;
Set the maximum number of iterations $it_{max}$;
Set the sample size N
Set the multilevel parameter $\rho$;
Set the distortion target, $\hat{L}_k$ as the peak load value;
while $\hat{L}_k < L_{\max}$ or $k < k_{\max}$ do
    for old = for new
    Sample the generators using $for_{old}$ and Algorithm 2;
    Computes the available generation in system state and sort them in descending order;
    $\hat{L}_k = S[(1-\rho)N]$;
    if $\hat{L}_k > L$ then
        $\hat{L}_k = S[(1-\rho)N]$;
    else
        $\hat{L}_k = L$
    while All $S_i$ are not checked do
        if $S_i < \hat{L}_k$ then
            H[$X_i$]=1
    Compute W according equation 6.14 ;
    Update $for_{new}$ for each unit according equation 6.15;
Output: $for_{old}$
```

The likelihood ratio (W), that represents the correlation between the $f.o.r_{\text{old}}$ and $f.o.r_{\text{new}}$, is calculated by equation 6.14.

$$W_i\left(X_i,\ \text{f.o.r}_{\text{ new}},\ \text{f.o.r}_{\text{ old}}\right) = \frac{\prod_{j=1}^{N}\left(1-\ \text{f.o.r}_{\text{ new }j}\right)^{X_{ij}} \times \text{f.o.r}_{\text{ new }j}^{\ (1-X_{ij})}}{\prod_{j=1}^{N}\left(1-\ \text{f.o.r}_{\text{ old }j}\right)^{X_{ij}} \times \text{f.o.r}_{\text{ old }j}^{\ (1-X_{ij})}} \tag{6.14}$$

Finally a new forced outage rate for each generator in the system is obtained by 6.15:

$$\text{f.o.r}_{\text{ new },j} = 1 - \frac{1}{NG} \times \frac{\sum_{j=1}^{N} H\left(X_i\right) W_i\left(X_i,\ \text{f.o.r}_{\text{ new}},\ \text{f.o.r}_{\text{ old}}\right) X_{ij}}{\sum_{j=1}^{N} H\left(X_i\right) W_i\left(X_i,\ \text{f.o.r}_{\text{ new}},\ \text{f.o.r}_{\text{ old}}\right)} \tag{6.15}$$

6.4.2 CNN application to the MCS process

The CNN application in MCS is straightforward: after the sampling stage and training, CNN incorporates the MCS process by replacing all the essential algorithms for a case evaluation. The premise of using the proposed algorithm (MCS-CNN) is to save time by only running the complex evaluation process when CNN classifies a loss of load case instead of running it for all samples, as demonstrated by Figure 6.3.

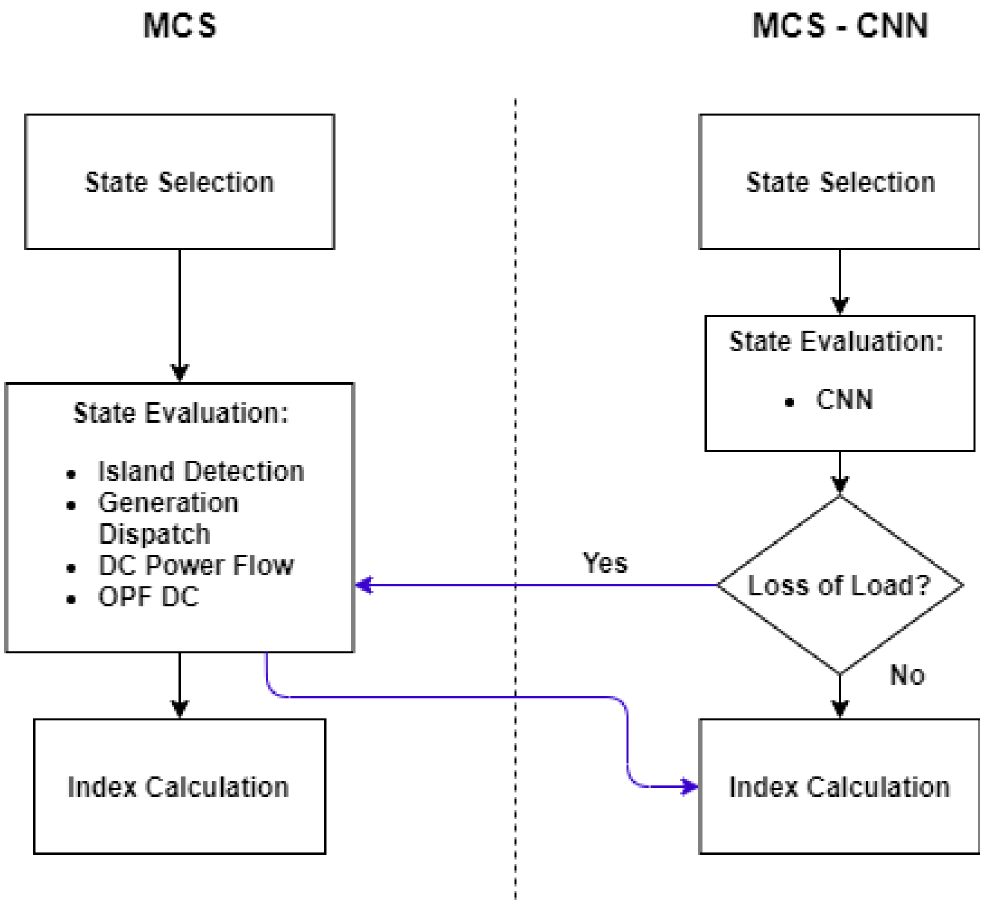


Figure 6.3: Diagram representing the two methods: the classic MCS (left) and the method proposed, MCS-CNN (right)

Chapter 7

Results

This chapter presents the implementation results of the proposed method MCS-CNN. The classic version of Non-sequential Monte Carlo (NSMCS) and Sequential Monte Carlo (SMCS) will be compared with the proposed MCS-CNN versions (NSMCS-CNN and SMCS-CNN) in terms of the estimation accuracy of the reliability indices and time necessary to complete the simulation. Both classic and proposed versions will be tested on 2 cases based on IEEE RTS 79. The first case is the IEEE RTS 79, with an annual load curve constant at its peak value of 2850 MW. The second is the IEEE RTS 79 with the original annual load curve. In both cases, generation system reliability data is taken into account, but transmission system components are considered available.

All simulations were computed in Python and executed in a PC with Intel Core i7-3630QM CPU (2.40GHz). The CNN training was done in a Nvidia T4 GPU provided by the Google Colab services.

To make a comparison between the two methods the "seed" responsible for the randomness in sample generator was fixed, this way both methods are evaluating the same samples.

7.1 IEEE RTS 79 with an annual load curve constant at 2850 MW (Modified IEEE RTS 79)

7.1.1 Non-Sequential Simulation

In this section results for the NSMCS and the NSMCS-CNN are presented with respect to the IEEE RTS 79 case with an annual load curve constant at its peak value of 2850 MW. Two values for variation coefficient, 5% and 1% were considered.

The value for LOLP obtained for $\beta \leq 1$, published on [36] is used to gauge the accuracy of the proposed method, NSMCS-CNN, for both values of β. LOLE value, although not present in [36], is deducted from LOLP. A crude NSMCS is computed as means to compare the process time and index values obtained.

Table 7.1 and 7.2 present the computed results for both methods (NSMCS and NSMCS-CNN) considering a 5% and a 1% coefficient of variation, respectively. Table 7.3 presents the process time analysis, including the metric that measures how many samples both methods evaluate per time interval and the process time relation between methods considering NSMCS process time as a baseline.

The **NSMCS-CNN** has proved to be **1.8 times faster** (7.3) than the **NSMCS** estimating LOLP (and inherently LOLE) for both values of variation coefficient. These results support that even with the increase in required precision (and consequently more samples to evaluate) NSMCS-CNN remains as the fastest method.

Also for both values of variation coefficient, NSMCS-CNN evaluates more samples than NSMCS , meaning that the trained CNN provides some false classifications that extend the convergence of the method. However since NSMCS-CNN evaluates more samples per second (in average - 7.3) than NSMCS, this does not seem a disadvantage for the proposed method.

Independently from CNN accuracy, NSMCS-CNN was able to estimate LOLP with small deviation when compared to the LOLP value reported in [36], in fact when $\beta \leq 1\%$, values are almost identical.

On table 7.2 and 7.1, LOLP and LOLE confidence intervals obtained with NSMCS-CNN include the index values reported in [36]. This is evidence of the excellent performance of NSMCS-CNN, which is capable of estimating reliability indices equally accurately as the NSMCS but quicker.

Table 7.1: Non-Sequential results for Modified IEEE RTS 79 with $\beta \leq 5\%$

$\beta \leq 5\%$	**LOLP**	**LOLE(h)**	**Iterations (Samples)**	**Process Time (s)**
NSMCS [36]	0.084	735.84	-	-
NSMCS	0.0919	805.21	3960	81
β (%)	4.99	4.99		
CI (95%)	[0.0829, 0.1009]	[726.38, 884.04]		
NSMCS-CNN	0.0902	790.05	4036	45
β (%)	4.99	4.99		
CI (95%)	[0.0814, 0.0990]	[712.63, 867.47]		

Table 7.2: Non-Sequential results for Modified IEEE RTS 79 with $\beta \leq 1\%$

$\beta \leq 1\%$	**LOLP**	**LOLE(h)**	**Iterations (Samples)**	**Process Time (min)**
NSMCS [36]	0.084	735.84	-	-
NSMCS	0.0851	745.5	107517	34
β (%)	0.99	0.99		
CI (95%)	[0.0834, 0.0868]	[730.89, 760.11]		
NSMC-CNN	0.0834	730.6	109900	19
β (%)	0.99	0.99		
CI (95%)	[0.0818, 0.0850]	[716.29, 744.93]		

Table 7.3: Process Time Analysis for Non-Sequential Modified IEEE RTS 79

	$\beta \leq 5\%$ **(Samples/s)**	$\beta \leq 1\%$ **(Samples/s)**
NSMCS	49	53
NSMCS-CNN	90	96
Process Time Acceleration	1.8	1.8

7.1.2 Sequential Simulation

In this section results for the SMCS and the SMCS-CNN are presented with respect to the IEEE RTS 79 case with an annual load curve constant at its peak value of 2850 MW. Two values for variation coefficient, 5% and 1% were considered.

Although results on [36] rely on non-sequential MCS, the LOLE value deducted from LOLP in the previous section is used to assess the accuracy of SMCS-CNN. Unfortunately, EENS values obtained for this specific case were not found in literature.

Table 7.4 and 7.5 present the computed results for both methods (SMCS and SMCS-CNN) considering a 5% and a 1% coefficient of variation, respectively. Table 7.6 presents the process time analysis, including the metric that measures how long, in average, each method takes to analyse a whole year and the process time relation between methods considering SMCS process time as a baseline.

Despite results on 7.6 display slightly different accelerations for the two values of β, its secure to say that **SMCS-CNN** can estimate EENS and LOLE, at least **2 times faster** than **SMCS** for both variance coefficient values. This also corroborated by the time SMCS-CNN takes to examine a full year that is half of what SMCS needs. Note that this examination contains the process of sampling the system state for all hours of the year and then evaluation. Since the only equal process between SMCS-CNN and SMCS is the one concerning sampling, its possible to see the CNN role on making SMCS-CNN quicker.

Looking to the results expressed on tables 7.4 and 7.5, SMCS-CNN established LOLE confidence intervals that include the LOLE estimate obtained in [36]. When comparing with confidence intervals obtained by SMCS for both reliability indices, there is a superposition, meaning that both methods expect similar index values. All this observations certify the good performance of SMCS-CNN, as an alternative method to compute reliability indices via sequential simulation.

Before collecting this results, it was expected that the number of sampled years by SMCS-CNN would always be greater than the number of years sampled by SMCS, following the trend observed when non-sequential simulation was computed. However this was not true when higher precision was required ($\beta \leq 1$).

Table 7.4: Sequential results for Modified IEEE RTS 79 with $\beta \leq 5\%$

$\beta \leq 5\%$	**LOLE (h/yr)**	**EENS (GWh/yr)**	**Sampled Years (No. samples)**	**Process Time (min)**
NSMCS [36]	735.8	-	-	-
SMCS	774.65	128.97	46 (401856)	125
β (%)	3.82	4.94		
CI (95%)	[716.70, 832.60]	[116.48, 141.45]		
SMCS-CNN	741.41	129.12	56 (489216)	69
β (%)	3.58	4.99		
CI (95%)	[689.43, 793.39]	[116.48, 141.76]		

Table 7.5: Sequential results for Modified IEEE RTS 79 with $\beta \leq 1\%$

$\beta \leq 1\%$	**LOLE (h/yr)**	**EENS (GWh/yr)**	**Sampled Years (No. samples)**	**Process Time (min)**
NSMCS [36]	735.8	-	-	-
SMCS	746.69	130.39	1474 (12876864)	3196
β (%)	0.69	0.99		
CI (95%)	[736.58, 756.81]	[127.84, 132.95]		
SMCS-CNN	731.10	127.92	1400 (12230400)	1493
β (%)	0.70	0.99		
CI (95%)	[721.01, 741.21]	[125.42, 130.43]		

Table 7.6: Average Time (in minutes) spent to analyse a whole year for $\beta \leq 5\%$ and $\beta \leq 1\%$ (Modified IEEE RTS 79) and Process time acceleration provided by SMCS-CNN in relation to SMCS

	$\beta \leq 5\%$ **(min/sampled yr)**	$\beta \leq 1\%$ **(min/sampled yr)**
SMCS	2.7	2.2
SMCS-CNN	1.2	1.1
Process Time Acceleration	1.81	2.14

7.2 IEEE RTS 79 with original load curve (IEEE RTS 79)

7.2.1 Non-Sequential Simulation

In this section, results for the NSMCS and the NSMCS-CNN are presented with respect to the IEEE RTS 79 case with the original load curve. Two values for variation coefficient, 5% and 1% were considered.

The benchmark used to evaluate the performance of NSMCS-CNN on reliability indices estimation is provided by the NSMCS since it was not possible to find specific results for this case in literature.

Table 7.7 and 7.8 present the computed results for both methods (NSMCS and NSMCS-CNN) considering a 5% and a 1% coefficient of variation, respectively. Table 7.9 presents the process time analysis, including the metric that measures how many samples both methods evaluate per time interval and the process time relation between methods considering NSMCS process time as a baseline.

In this case, according to table 7.9, **NSMCS-CNN** was **3 times faster** than **NSMCS** to estimate LOLP, and consequently, LOLE. There is no noticeable difference in speed when the required precision increases and NSMCS-CNN maintains its position as the fastest.

As observed in the previous case, NSMCS-CNN evaluates more samples than NSMCS.

In both precision levels, NSMCS-CNN provided estimates for LOLP that are essentially equal to the ones supplied by NSMCS. The same can be concluded for the estimated LOLE.

Confidence intervals obtained for LOLP and LOLE by NSMCS-CNN are fully included in the intervals determined by NSMCS, which proves, even more, the agreement between both methods.

Table 7.7: Non-Sequential results for IEEE RTS 79 with $\beta \leq 5\%$

$\beta \leq 5\%$	**LOLE**	**LOLE(h)**	**Iterations (Samples)**	**Process Time (min)**
NSMCS	0.00127	11.12	315025	74
β (%)	4.99	4.99		
CI (95%)	[0.00115, 0.00139]	[10.03, 12.21]		
NSMC-CNN	0.00126	11.05	317036	25
β (%)	4.99	4.99		
CI (95%)	[0.00114, 0.00138]	[9.97, 12.14]		

Table 7.8: Non-Sequential results for IEEE RTS 79 with $\beta \leq 1\%$

$\beta \leq 1\%$	**LOLE**	**LOLE(h)**	**Iterations (Samples)**	**Process Time (min)**
NSMCS	0.00121	10.63	8227033	1849
beta (%)	0.99	0.99		
CI (95%)	[0.00119, 0.00124]	[10.43, 10.84]		
NSMC-CNN	0.00121	10.57	8280955	614
beta (%)	0.99	0.99		
CI (95%)	[0.00118, 0.00123]	[10.36, 10.77]		

Table 7.9: Process Time Analysis for Non-Sequential IEEE RTS 79

	$\beta \leq 5\%$ **(Samples/min)**	$\beta \leq 1\%$ **(Samples/min)**
NSMCS	4257	4449
NSMCS-CNN	12681	13487
Process Time Acceleration	2.96	3.01

7.2.2 Sequential Simulation

In this section, results for the SMCS and the SMCS-CNN are presented with respect to the IEEE RTS 79 case with the original load variation curve. Just one value for variation coefficient, 5% was considered, however it is sufficient to evaluate the proposed method potential in this study case.

Results on 7.11 present **SMCS-CNN** has a method **3.38 times faster** for reliability index calculation than **SMCS**. In average, SMCS-CNN can reduce the time needed to sample a whole a year by three times when comparing to the time consumed by SMCS.

In precision matter, LOLE and EENS estimated values by SMCS-CNN are a little far from the ones obtained with SMCS however the confidence intervals interception confirm that both expect the same values for the studied reliability indices.

As in sequential simulation of the previous case, the number of sampled years by SMCS-CNN is inferior to the number of sampled years in SMCS, which was not expected. However, SMCS-CNN was capable of estimating LOLE and EENS with the same precision as SMCS but quicker.

Table 7.10: Sequential results for IEEE RTS 79 with $\beta \leq 5\%$

$\beta \leq 5\%$	**LOLE (h/yr)**	**EENS (MWh/yr)**	**Sampled Years (No. samples)**	**Process Time (min)**
SMCS	10.17	1284.85	1452 (12684672)	1924
β (%)	3.39	4.99		
CI (95%)	[9.49, 10.84]	[1159.09, 1410.60]		
SMCS-CNN	10.49	1331.74	1237 (10806432)	569
β (%)	3.37	4.99		
CI (95%)	[9.79, 11.18]	[1201.33, 1462.16]		

Table 7.11: Average Time (in minutes) spent to process a whole year for $\beta \leq 5\%$ (IEEE RTS 79) and Process time acceleration provided by SMCS-CNN in relation to SMCS

	$\beta \leq 5\%$ **(min/sampled yr)**
SMCS	1.3 min
SMCS-CNN	0.5 min
Process Time Acceleration	3.38

7.3 Comments

For the cases considered (original load curve and constant load curve), MCS-CNN was accurate and faster than MCS.

Additionally is possible to form a more detailed insight about the performance of MCS-CNN in relation to the reliability of a system.

Case system with original load curve is more reliable than the one with a constant load curve at its peak.

Considering the results obtained in non-sequential version for both case studies, MCS-CNN was capable to estimate reliability indices 3 times faster for case with the original curve while for case with the constant curve was capable to estimate indices 2 times quicker. The same can be observed for sequential simulation.

This concludes that MCS-CNN, although performing well in both cases, is suited for more reliable systems. This is expected since, CNN evaluates the high number of negative samples (not loss of load) without the need to resort to the complex evaluation algorithm used by MCS, thus saving a lot of time.

It was proven the ability of MCS-CNN in reducing simulation time. Nevertheless, process times are still extensive and unacceptable for a solution implemented in an actual power system study. If simulations were computed using more powerful hardware, processes time might be shorter and enable more simulations to be carried out.

In sequential simulations, the number of sampled years by MCS-CNN was inferior to the number of sampled years by MCS, which was not expected. This may be related to the way sample variance is updated in each simulation type. In sequential simulation, samples are collected in year batches and variance is updated at the end of every year. False classifications provided by the CNN on yearly batch do not seem to affect so much the variance (thus the convergence of the method) as the individual analysis of the samples, that occurs in non-sequential simulation. Although MCS-CNN was able to estimate indices with equal precision as MCS with less samples, the objective of the proposed method was not to reduce the number of samples evaluated, but speed up evaluation process of this samples.

Chapter 8

Conclusion and Future Work

This dissertation aimed to apply a Convolutional Neural Network, a deep learning methodology, to Monte Carlo Simulation for composite generation and transmission system reliability assessment to improve the computational effort required by the simulation.

Many challenges were faced during this elaboration of this work.

The chosen programming language to implement the developed algorithms was Python. However, despite being at ease with programming, the writer of this thesis did not have any prior knowledge of it. Therefore the initial steps were focused on gaining awareness of this new language.

As mentioned in the dedicated chapter, this network receives an image as input, and the data available for this work does not take that form. So much effort was put into merging power systems data with this powerful deep learning technique.

Fortunately, the implementation of this network was eased with already existing frameworks, leaving more time for experimentation and "trial and error".

Monte Carlo simulation on reliability assessment was not a new concept for the writer of this thesis. He had the chance to explore it in other courses along with the masters he is enrolled in. Nonetheless, the composite system reliability assessment was approached in the context of this work. Thus a complete study concerning the algorithms involved in this kind of reliability studies was performed.

A study was carried out to explore the optimal input image structure to feed the CNN. The study concluded that a 2D input structure is ideal for organising the non-image data inputs, thus providing an optimal way to extract essential patterns to the classification problem.

Although used on MCS-CNN, the developed convolutional network should be considered a solo product of this work, with applications in power systems besides Monte Carlo simulation and reliability, for instance, as a tool for engineers on dispatch centres.

To study the proposed method, MCS-CNN, two types of simulation, non-sequential and sequential, were computed considering the IEEE RTS 79 case with the original load curve and a load curve constant at his peak of 2850 MW. A crude MCS was also tested in the same conditions for comparison.

On IEEE RTS 79 case with the original load curve, MCS-CNN has proved to be two times faster than crude MCS estimating reliability indices with the highest required precision ($\beta \leq 1$) for both simulations. Values estimated by MCS-CNN are accurate since confidence intervals obtained are coincident with the ones obtained with MCS.

On IEEE RTS 79 case with the constant load curve, MCS-CNN has also proved to be faster than MCS estimating reliability indexes with the highest required precision by three times, in both simulations. Index accuracy is also confirmed by the superposition of confidence intervals obtained.

Overall, MCS-CNN demonstrated that when applied to Monte Carlo simulation for composite system reliability assessment, deep learning techniques, like Convolutional Neural Networks, can reduce the time and necessary computational effort, especially when power systems are very reliable.

The main objective of this dissertation was achieved yet there is space for improvement, namely:

- Include the methodology to obtain the "Loss of Load Classifier" as a preamble in the MCS-CNN script.
- Parallelise the MCS-CNN with the methodology described in [35].
- Include transmission system reliability data on the developed CNN's input structure.

References

[1] R. Billinton and R.N. Allan. Power-system reliability in perspective. *Electronics and Power*, 30(3):231–236, 1984. doi:10.1049/ep.1984.0118.

[2] Mauro Augusto da Rosa. *Agent-based technology applied to power systems reliability*. PhD thesis, Universidade do Porto, 2009.

[3] S. A. McLeod. Introduction to the normal distribution (bell curve). URL: https://www.simplypsychology.org/normal-distribution.html.

[4] Mauro A. da Rosa, Armando M. Leite da Silva, and Vladimiro Miranda. Multi-agent systems applied to reliability assessment of power systems. *International Journal of Electrical Power & Energy Systems*, 42(1):367–374, November 2012. URL: https://doi.org/10.1016/j.ijepes.2012.03.048, doi:10.1016/j.ijepes.2012.03.048.

[5] Michael Copeland. What's the difference between artificial intelligence, machine learning and deep learning? URL: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/.

[6] Karpathy Andrej. Cs231n convolutional neural networks for visual recognition. URL: https://cs231n.github.io/convolutional-networks/#fc.

[7] IBM Cloud Education. Convolutional neural networks. URL: https://www.ibm.com/cloud/learn/convolutional-neural-networks#toc-how-do-con--z4UwR2M.

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, December 1989. URL: https://doi.org/10.1162/neco.1989.1.4.541, doi:10.1162/neco.1989.1.4.541.

[9] Introduction to parallel computing tutorial. URL: https://hpc.llnl.gov/training/tutorials/introduction-parallel-computing-tutorial.

[10] Probability Subcommittee. IEEE reliability test system. *IEEE Transactions on Power Apparatus and Systems*, PAS-98(6):2047–2054, November 1979. URL: https://doi.org/10.1109/tpas.1979.319398, doi:10.1109/tpas.1979.319398.

[11] Aref Hashemi Fath, Farshid Madanifar, and Masood Abbasi. Implementation of multilayer perceptron (MLP) and radial basis function (RBF) neural networks to predict solution gas-oil ratio of crude oil systems. *Petroleum*, 6(1):80–91, March 2020. URL: https://doi.org/10.1016/j.petlm.2018.12.002, doi:10.1016/j.petlm.2018.12.002.

[12] Logistic regression. URL: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html.

[13] Gradient descent and stochastic gradient descent. URL: http://rasbt.github.io/mlxtend/user_guide/general_concepts/gradient-optimization/.

[14] What's the difference between a cpu and a gpu? URL: https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/.

[15] X.Yu and C.Singh. Application of importance sampling in power system reliability studies. *National Power Systems Conference*, 2004. URL: https://www.iitk.ac.in/npsc/Papers/NPSC2004/305-Art-305.pdf.

[16] Armando M. Leite da Silva, Leonidas Chaves de Resende, Luiz AntÔnio da Fonseca Manso, and Vladimiro Miranda. Composite reliability assessment based on monte carlo simulation and artificial neural networks. *IEEE Transactions on Power Systems*, 22(3):1202–1209, August 2007. URL: https://doi.org/10.1109/tpwrs.2007.901302, doi:10.1109/tpwrs.2007.901302.

[17] Lou Sluis. *Transients in power systems*. Wiley, Chichester New York, 2001.

[18] R. Billinton and E. Khan. A security based approach to composite power system reliability evaluation. *IEEE Transactions on Power Systems*, 7(1):65–72, 1992. doi:10.1109/59.141688.

[19] Pei Zhang, Ke Meng, and Zhaoyang Dong. *Probabilistic vs Deterministic Power System Stability and Reliability Assessment*, pages 117–145. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. URL: https://doi.org/10.1007/978-3-642-04282-9_5, doi:10.1007/978-3-642-04282-9_5.

[20] Roy Billinton and Ronald N. Allan. *Reliability Evaluation of Power Systems*. Springer US, 1996. URL: https://doi.org/10.1007/978-1-4899-1860-4, doi:10.1007/978-1-4899-1860-4.

[21] Roy Billinton and Ronald N. Allan. *Reliability Assessment of Large Electric Power Systems*. Springer US, 1988. URL: https://doi.org/10.1007/978-1-4613-1689-3, doi:10.1007/978-1-4613-1689-3.

[22] Roy Billinton and Wenyuan Li. *Reliability Assessment of Electric Power Systems Using Monte Carlo Methods*. Springer US, 1994. URL: https://doi.org/10.1007/978-1-4899-1346-3, doi:10.1007/978-1-4899-1346-3.

[23] Leonel de Magalhães Carvalho. *Advances on the sequencial Monte Carlo reliability assessment of generation-transmission system using cross-entropy and population-based Methods*. 2013.

[24] Armando M. Leite da Silva, Reinaldo A. Gonzalez-Fernandez, Warlley S. Sales, and Luiz A.F. Manso. Reliability assessment of time-dependent systems via quasi-sequential monte carlo simulation. In *2010 IEEE 11th International Conference on Probabilistic Methods Applied to Power Systems*. IEEE, June 2010. URL: https://doi.org/10.1109/pmaps.2010.5528326, doi:10.1109/pmaps.2010.5528326.

[25] J.C.O. Mello, M.V.F. Pereira, and A.M. Leite da Silva. Evaluation of reliability worth in composite systems based on pseudo-sequential monte carlo simulation. *IEEE Transactions on Power Systems*, 9(3):1318–1326, 1994. URL: https://doi.org/10.1109/59.336134, doi:10.1109/59.336134.

[26] Inês Maria Afonso Trigo de Freitas Alves. *Monte Carlo Parallel Implementation for Reliability Assessent*. 2019.

[27] Steven Thompson. *Sampling*. John Wiley & Sons, Hoboken, N.J, 2012.

[28] Introduction to power system reliability. In *Electric Power Grid Reliability Evaluation*, pages 185–191. John Wiley & Sons, Inc., December 2018. URL: https://doi.org/10.1002/9781119536772.ch7, doi:10.1002/9781119536772.ch7.

[29] G.C. Oliveira, M.V.F. Pereira, and S.H.F. Cunha. A technique for reducing computational effort in monte-carlo based composite reliability evaluation. *IEEE Transactions on Power Systems*, 4(4):1309–1315, 1989. URL: https://doi.org/10.1109/59.41680, doi:10.1109/59.41680.

[30] C. Marnay and T. Strauss. Effectiveness of antithetic sampling and stratified sampling in monte carlo chronological production cost modeling (power systems). *IEEE Transactions on Power Systems*, 6(2):669–675, May 1991. URL: https://doi.org/10.1109/59.76711, doi:10.1109/59.76711.

[31] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method*. Springer New York, 2004. URL: https://doi.org/10.1007/978-1-4757-4321-0, doi:10.1007/978-1-4757-4321-0.

[32] Zdravko I. Botev, Dirk P. Kroese, Reuven Y. Rubinstein, and Pierre L'Ecuyer. The cross-entropy method for optimization. In *Handbook of Statistics - Machine Learning: Theory and Applications*, pages 35–59. Elsevier, 2013. URL: https://doi.org/10.1016/b978-0-444-53859-8.00003-5, doi:10.1016/b978-0-444-53859-8.00003-5.

[33] Leonel de Magalhaes Carvalho, Reinaldo Andres Gonzalez-Fernandez, Armando Martins Leite da Silva, Mauro Augusto da Rosa, and Vladimiro Miranda. Simplified cross-entropy based approach for generating capacity reliability assessment. *IEEE Transactions on Power Systems*, 28(2):1609–1616, May 2013. URL: https://doi.org/10.1109/tpwrs.2012.2213618, doi:10.1109/tpwrs.2012.2213618.

[34] Armando M. Leite da Silva, Reinaldo A. G. Fernandez, and Chanan Singh. Generating capacity reliability evaluation based on monte carlo simulation and cross-entropy methods. *IEEE Transactions on Power Systems*, 25(1):129–137, February 2010. URL: https://doi.org/10.1109/tpwrs.2009.2036710, doi:10.1109/tpwrs.2009.2036710.

[35] Ines M. Alves, Vladimiro Miranda, and Leonel M. Carvalho. Parallel GPU implementation for fast generating system adequacy assessment via sequential monte carlo simulation. In *2020 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*. IEEE, August 2020. URL: https://doi.org/10.1109/pmaps47429.2020.9183554, doi:10.1109/pmaps47429.2020.9183554.

[36] Dogan Urgun and Chanan Singh. Composite system reliability analysis using deep learning enhanced by transfer learning. In *2020 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*. IEEE, August 2020. URL: https://doi.org/10.1109/pmaps47429.2020.9183474, doi:10.1109/pmaps47429.2020.9183474.

[37] Artificial intelligence in healthcare. URL: https://www.siemens-healthineers.com/pt/digital-health-solutions/artificial-intelligence-in-healthcare.

[38] Bonnie G Buchanan. Artificial intelligence in finance. 2019. URL: https://zenodo.org/record/2612537, doi:10.5281/ZENODO.2612537.

[39] Bernard Marr. How is artificial intelligence and machine learning used in engineering? URL: https://www.forbes.com/sites/bernardmarr/2020/02/07/how-is-artificial-intelligence-and-machine-learning-used-in-engineering/?sh=4a2531344a85.

[40] Thomas Wood. Softmax function definition | deepai. URL: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer.

[41] Alok Sharma, Edwin Vans, Daichi Shigemizu, Keith A. Boroevich, and Tatsuhiko Tsunoda. DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific Reports*, 9(1), August 2019. URL: https://doi.org/10.1038/s41598-019-47765-6, doi:10.1038/s41598-019-47765-6.

[42] Vladimiro Miranda, Pedro A. Cardoso, Ricardo J. Bessa, and Ildemar Decker. Through the looking glass: Seeing events in power systems dynamics. *International Journal of Electrical Power & Energy Systems*, 106:411–419, March 2019. URL: https://doi.org/10.1016/j.ijepes.2018.10.024, doi:10.1016/j.ijepes.2018.10.024.

[43] Luís Miguel Brito Teixeira. *Miss SAIGON - Missing Signal Appraising in Globally Optimized Networks*. 2019.

[44] An end-to-end open source machine learning platform. URL: https://www.tensorflow.org/.

[45] Introduction to keras for engineers. URL: https://keras.io/getting_started/intro_to_keras_for_engineers/.

[46] Google colab. URL: https://research.google.com/colaboratory/faq.html.

[47] Jason Brownlee. What is the difference between test and validation datasets? URL: https://machinelearningmastery.com/difference-test-validation-datasets/.

[48] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, April 2016. URL: https://doi.org/10.1098/rsta.2015.0202, doi:10.1098/rsta.2015.0202.

[49] Dalwinder Singh and Birmohan Singh. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97:105524, December 2020. URL: https://doi.org/10.1016/j.asoc.2019.105524, doi:10.1016/j.asoc.2019.105524.

[50] Jason Brownlee. Difference between a batch and an epoch in a neural network. URL: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/.

[51] PRAMOD GUPTA and NARESH K. SINHA. Neural networks for identification of non-linear systems: An overview. In *Soft Computing and Intelligent Systems*, pages 337–356. Elsevier, 2000. URL: https://doi.org/10.1016/b978-012646490-0/50017-2, doi:10.1016/b978-012646490-0/50017-2.

[52] Understanding binary cross-entropy / log loss: a visual explanation. URL: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac602518

[53] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. arXiv:arXiv:1412.6980.

[54] Adam. URL: https://keras.io/api/optimizers/adam/.

[55] Antonio Simões Costa. Configurador de redes elétricas (in portuguese). https://simoes.sites.ufsc.br/assp/configurador_de_redes.pdf.

[56] Manuel António Matos. Introdução ao trânsito de potências (in portuguese). https://paginas.fe.up.pt/~mam/tp.pdf.

[57] Pypower. URL: https://pypi.org/project/PYPOWER/.

[58] M.Glavic F.Capitanescu and L.Wehenkel. An interior-point method based optimal power flow. *ACOMEN conference*, 2005. URL: https://orbi.uliege.be/bitstream/2268/28224/1/paper.pdf.

[59] Optimization and root finding (scipy.optimize). URL: https://docs.scipy.org/doc/scipy/reference/optimize.html.