Hochschule Zittau/Görlitz
UNIVERSITY OF APPLIED SCIENCES

Master Thesis

Title of the master thesis

AI-Based Static Voltage Stability Analysis of Power Grids

submitted by
Timon Conrad

born on
March 9, 1999

born in
Görlitz, Germany

in partial fulfilment of the requirements for the degree of

Master of Engineering
(M.Eng.)

Faculty of Mechanical Engineering
at Hochschule Zittau/Görlitz – University of Applied Sciences

---

Study programme:	Energie- und Umwelttechnik
Supervisor:	Prof. Dr. rer. nat. Stefan Bischoff
Assessor:	Prof. Dr.-Ing. Cezary Dzienis
Second assessor:	Prof. Dr.-Ing. Matthias Kunick
Date of submission:	September 22, 2023

Aufgabenstellung für die Masterarbeit

Name: **Timon Conrad, MEZm22S**

Studiengang: Energie- und Umwelttechnik

Studienrichtung: Erneuerbare Energien und Energieeffizienz

Thema: **KI-gestützte Stabilitätsbewertung elektrischer Netze**

Zielstellung:

In dieser Masterarbeit soll die Anwendung künstlicher Intelligenz (KI) zur Bewertung der statischen Stabilität elektrischer Netze unter Berücksichtigung der bisherigen Entwicklungen in diesem Forschungsgebiet untersucht werden. Es wird ein mehrschichtiges neuronales Netz implementiert und evaluiert. Hierbei werden zahlreiche simulierte Netzwerkzustände als Datengrundlage verwendet. Die Anwendung und Bewertung der Implementierung erfolgen am Beispiel eines IEEE 5-Bus-Systems. Zudem wird die Skalierbarkeit der Methode auf größere Netze untersucht, um die Anwendbarkeit der Methode für unterschiedliche Netzwerkgrößen beurteilen zu können.

Folgende Teilaufgaben sind zu lösen:

- Literaturrecherche zu bestehenden Methoden zur statischen Stabilitätsanalyse in elektrischen Netzen;
- Erstellung und Analyse von Trainingsdaten mittels Lastflussberechnungen für ein IEEE 5-Bus-System und mit mindestens einem alternativen Ansatz;
- Konzeption und Implementierung eines mehrschichtigen neuronalen Netzes zur Stabilitätsanalyse;
- Training und Bewertung der Ergebnisse des KI-Modells anhand der erzeugten Trainings- und Testdaten;
- Analyse der Skalierbarkeit des Ansatzes für Netze verschiedener Größen;
- Diskussion der Ergebnisse und Ausblick bezüglich der Anwendbarkeit in der Praxis.

Betreuer: Herr Prof. Dr. rer. nat. S. Bischoff, Hochschule Zittau/Görlitz

Gutachter: Herr Prof. Dr.-Ing. C. Dzienis, Hochschule Zittau/Görlitz
Herr Prof. Dr.-Ing. M. Kunick, Hochschule Zittau/Görlitz

Ausgehändigt am: 31.03.2023

Einzureichen bis: 29.09.2023

Registrier-Nr.: MMA23004

B. Bellair

Prof. Dr.-Ing. Bernd Bellair
Dekan

Statutory Declaration

I hereby declare that I have prepared this thesis without the unauthorised assistance of third parties and and without the use of any other means than those indicated. All direct or indirect sources used are acknowledged as references. In the selection and evaluation of the material as well as in the production of the manuscript, I received support services from the following persons:

Prof. Dr.-Ing. Cezary Dzienis
Prof. Dr. rer. nat. Stefan Bischoff
Prof. Dr.-Ing. Wolfgang Kästner

No other persons were involved in the intellectual production of the present work.

For translations into English, the tool "DeepL" in version 4.7.1.9722 was used. The translated text was written by myself. Furthermore, the function "DeepL Write" in beta version was used to correct grammatical and spelling mistakes and to adjust the style of writing. Furthermore, the tool "Chat GPT" from OpenAI in version 4.0 was used as a help for programming, in particular for the support during the implementation and bug fixing. These programs are not part of the following scientific work.

I am aware that failure to comply with this declaration may result in the subsequent withdrawal of the master degree.

This thesis was not previously presented to another examination board and has not been published.

Zittau, September 22, 2023

Signature .....
Timon Conrad

Acknowledgements

I take this opportunity to thank all those who have supported me directly or indirectly in making this thesis possible.

I would like to start by thanking Prof. Dzienis for giving me the opportunity to work on this innovative topic. I would also like to thank him for his valuable literature recommendations on load flow calculation and stability assessment, have repeatedly led me out of dead ends. The regular exchanges and discussions on various topics have been very enriching.

I also want to mention Prof. Bischoff, who inspired me with the MLP approach to AI. When I reached a point where successful training with Keras seemed impossible, he referred me to Prof. Kästner.

I have to thank Prof Kästner for introducing me to DataEngine and giving me the opportunity to use the software. It helped me a lot with the pruning procedure and also with the Kohonen network.

I'm grateful that I was allowed to work at the HSZG while writing my master thesis. This allowed me to work regularly and with focus. In addition, I was able to use the computing infrastructure and did not have to do the computationally intensive training data generation at home.

Special thanks to Nicola Bell, Lukas Müller and Jonas Conrad for proofreading the thesis.

Finally, I would like to thank my family and friends for their support and motivation during this time of uncertainty.

Abstract

Titel: AI-based static voltage stability analysis of power grids

Key words: AI (artificial intelligence), ML (machine learning), ANN (artificial neural networks), Power Grids, Smart Grid, Voltage Stability Analysis, dV/dQ Sensitivity Analysis, PowerFactory, Keras

The objective of this thesis was to implement an AI system to analyse the static voltage stability of power grids and assess its efficacy in achieving faster computation times compared to existing approaches.

To carry out this implementation, a training dataset of randomly generated cases for a 5 bus grid was created using the load flow calculation and dV/dQ sensitivity analysis in the PowerFactory program. Various artificial neural networks were optimised using the pruning function in the DataEngine programme and implemented using the Python library Keras. The evaluation focused on applicability, the impact of reducing features and cleaning the dataset, as well as examining variations in power grid size.

The thesis indicated that, despite obtaining high-accuracy results during training, the models lacked sufficient generalisation ability. Expanding the dataset did not enhance this ability and the model necessitated more time for both data production and training when the power grid was expanded.

Regardless of the utilization of diverse approaches, including MLPs and Kohonen maps, adequate generalisation was not achieved. Future research should concentrate on expanding the features, testing multiple artificial neural networks and integrating load flow calculations with AI. It is suggested that the methodology and approach to data cleaning should be re-evaluated and further research should explore contemporary optimisers or additional artificial neural network models.

Kurzfassung

Titel: KI-basierte Analyse der statischen Spannungsstabilität von Stromnetzen

Schlüsselwörter: KI (Künstliche Intelligenz), ML (Maschinelles Lernen), künstliche neuronale Netze, Stromnetze, Smart Grid, Spannungsstabilitätsanalyse, dV/dQ Sensitivitätsanalyse, PowerFactory, Keras

Das Ziel dieser Arbeit bestand in der Entwicklung einer KI zur Analyse der statischen Spannungsstabilität von Stromnetzen sowie der Bewertung ihrer Anwendbarkeit, um im Vergleich zu bisherigen Ansätzen schnellere Rechenzeiten zu erreichen.

Zu diesem Zweck wurde ein Trainingsdatensatz für ein 5-Bus-Netz mit zufällig generierten Fällen erstellt, wobei das Programm PowerFactory für die Lastflussberechnung und die dV/dQ-Sensitivitätsanalyse verwendet wurde. Verschiedene künstliche neuronale Netze wurden mithilfe des Pruning-Verfahrens im Programm DataEngine optimiert und mit der Python-Bibliothek Keras implementiert. Der Schwerpunkt lag auf der Bewertung ihrer Anwendbarkeit, dem Einfluss der Merkmalsreduktion und Datensatzbereinigung sowie der Analyse der Variation der Netzwerkgröße.

Die Untersuchungen ergaben, dass ein Training mit hoher Genauigkeit erfolgreich durchgeführt werden konnte. Das Modell zeigte jedoch eine unzureichende Generalisierungsfähigkeit. Die Erweiterung des Datensatzes verbesserte diese Fähigkeit nicht. Zudem benötigte das Modell mehr Zeit für die Datengenerierung und das Training, wenn das elektrische Netzwerk erweitert wurde.

Trotz der Verwendung verschiedener Ansätze, wie MLPs und Kohonen-Karten, konnte keine ausreichende Generalisierungsfähigkeit erreicht werden. Zukünftige Forschung sollte sich auf die Erweiterung der Merkmale, den Test weiterer künstliche neuronale Netze und die Integration von Lastflussberechnungen mittels KI konzentrieren. Es wird empfohlen, die Methodik und den Ansatz der Datenbereinigung zu überdenken und moderne Optimierer oder weitere künstliche neuronale Netzwerkmodelle in zukünftigen Untersuchungen zu berücksichtigen.

Table of Contents

1 Introduction

1.1 Background and Motivation

Against the backdrop of climate change and geopolitical dependencies, the transition to renewable energy, known as the energy transition, is a critical issue of this century. It is altering how energy is produced, distributed and consumed. Power grids[1] hold a pivotal role in this regard as they form the backbone of our energy infrastructure.

Traditionally, electricity grids were primarily supplied by large, centralised power plants. This allowed for relatively simple control and monitoring of the grid through grid control centres and grid control technology. Skilled personnel, supported by technical support tools, ensured that the present grid was coordinated and reliably kept stable in the high and medium voltage (HV & MV) grids.

The rapid and continuing expansion of renewable energy, particularly in the form of distributed generation, has changed this landscape. These power plants, which are highly dependent on weather, season and time of day, also feed into the low voltage (LV) grid. This situation has caused a shift in the traditional energy flow, where excess energy from the distribution grid (LV & MV) has to be fed back into the transmission grid (HV). This paradigm shift poses significant challenges to the LV grid, particularly in terms of grid stability. The previous method of using grid control centres cannot be replicated in every LV grid or coordinated properly. If the number of uncoordinated generators increases excessively, the potential for unstable grid conditions increases [29].

Complex algorithms have been developed for managing the grid and preventing unstable situations. [6] As the amount of uncontrollable generators rises, the computational time needed for running these algorithms becomes an essential resource.

In this context, the term "smart grids" is gaining importance. These are grids that utilise information and communication technologies (ICT) to interconnect all types of energy systems on the grid. [29] Smart meters collect data while management systems provide "intelligence" by coordinating electricity supply with demand. In this context, data-based and data-driven applications, such as artificial intelligence (AI), can significantly contribute by reducing computing times and accurately predicting the state of the grid. [15]

Overall, the energy transition is in a critical phase and there are still challenges to overcome. In addition to expanding smart infrastructure in MV and LV grids, another hurdle is integrating and developing AI into the grid. This digitalisation and AI could prove vital in overcoming these challenges and ensuring a sustainable, reliable and efficient energy supply for the future.

[1] In this thesis, the word "network" is deliberately avoided in connection with electricity, in order to make it clearly applicable to AI.

1.2 Research Objectives

The main research objective of this thesis is to develop an artificial intelligence (AI) approach for assessing the static voltage stability of power grids and to improve the computational time compared to existing approaches. Initially, a case generator produces synthetic data to establish the training data required for the artificial neural networks (ANN) as no dataset is currently obtainable. Numerous load flow calculation methods are employed to determine the static voltage stability of the generated cases. On the basis of this load flow calculation, the dV/dQ sensitivity is used for the evaluation of the static voltage stability, which is the target.

An important aspect of this thesis is the creation and improvement of different ANN and architectures adapted by the pruning function. At the same time, the influence of data cleaning and feature reduction methods on the ANN output and accuracy is analysed. Additionally, scalability is appraised as a prominent theme. This incorporates evaluating how adjustments to the ANN size impact the training and testing phases' efficacy. It will also establish whether the suggested methodology can be utilised in more extensive grids.

Finally, the effectiveness and practicality of the developed ANN for assessing static voltage stability will be evaluated through a thorough analysis of the results and conclusions.

1.3 Structure of the Thesis

The structure of the thesis is shown in the flowchart in Fig. 1.1. This flowchart visualises the connections between the chapters and sections.

At the beginning of the thesis, a literature review is conducted in chapter 2 to examine previous advancements and ascertain their relevance to the present thesis. Chapter 3 presents fundamental theories and principles of load flow calculation, stability indicators and artificial intelligence and ANNs as sections, drawing on various sources.

Chapter 4 describes the data generation. A case generator is used to generate different cases for the load flow calculation in section 4.2 and calculate them in section 4.3. This produces a dataset with voltage contingencies. This dataset is analysed, compared and cleaned in section 4.4 to establish a database for use in ANN for training, validation and testing in chapter 5. As part of the AI approach, section 5.1 considers data with and without conflicts and with a reduced number of features. Two types of ANN are developed and considered in this thesis (see section 5.2 & section 5.3).

The aspect of scaling is considered in chapter 6. It includes data size scaling in section 6.1 and additional datasets are generated for grid size scaling analysis in section 6.2. The limitations of the presented approach will be discussed and highlighted in section 6.3. Finally, chapter 7 summarises the results of the thesis and presents potential future developments.

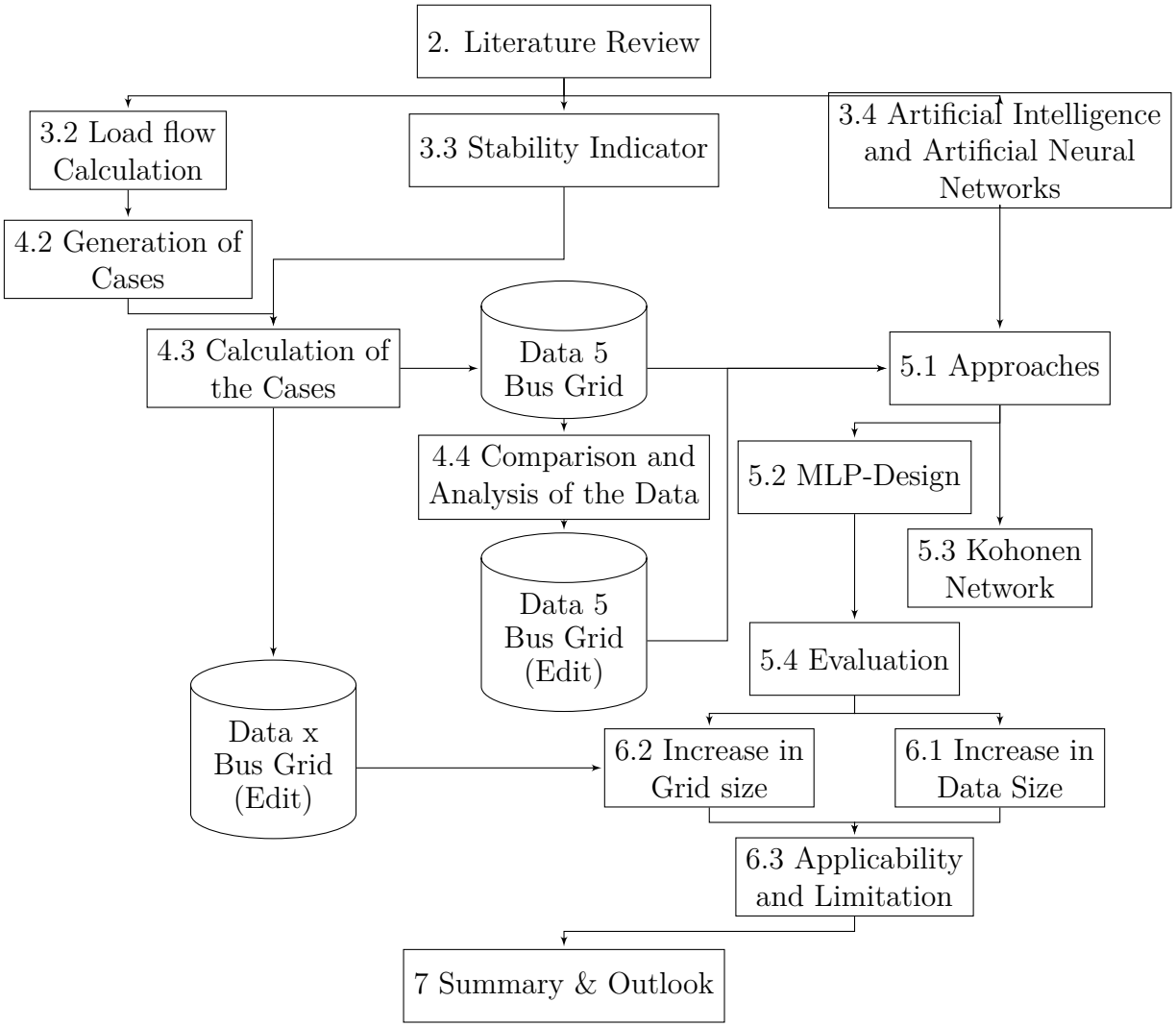


Fig. 1.1: Flowchart of the Thesis

It is important to note that the flow chart does not contain all the information presented in the appendices. The appendices provide supplementary information that supports and explains the main part of the work.

2 Literature Review

Extensive literary research was conducted in preparation for this master thesis for familiarization with the topic. The following sources, arranged by related topics, are the most relevant to this thesis. Their content and impact are briefly summarised.

2.1 Load Flow Calculation

Book: Analiza systemów elektroenergetyczny [26]
(in English: Analyse of electro energy systems)
Kremens & Sobierajski, Wydawnictwa Naukowo-Techniczne Warszawa, 1996

In this book, the authors describe, among other things, the load flow calculation according to the Newton-Raphson and Gauss-Seidel methods. In the book, exemplary calculations for both procedures were performed, so that the correct implementations could be evaluated. This was done with a 5 bus grid which is different from the Institute of Electrical and Electronics Engineers(IEEE) 5 bus grid. The grid used there does not contain a PV bus, only PQ buses and a slack bus. In a later chapter of the book, the implementation is roughly discussed. In section 3.2 the procedures will be described in detail.

2.2 Existing Methods for Static Voltage Stability Analysis

Expert report: P2021 Gutachten zur NEMO VIII, Los 1 - Stabilität [25]
(in English: Expert Report for NEMO VIII - Stability)
Schmieg, DIgSILENT GmbH, 2022

The expert report on system stability in the distribution grid commissioned by the German Federal Network Agency (Bundesnetzagentur) describes the requirements for the evaluation of system stability and the testable NEP (in German Netz Entwicklungspläne - grid development plans) criteria. For future developments in the distribution grid as conventional power plants are shut down and converter-based generators increase, the following criteria remain: static voltage and static angle stability, dynamic voltage stability, transient stability and frequency and voltage stability. Static voltage stability is the first mentioned criterion. According to the source, the static voltage stability is given if the dV/dQ sensitivity at each grid bus is positive for all conditions under observation. These values can be provided by the corresponding program in PowerFactory, also from DIgSILENT GmbH.

Lecture: Dynamik und Stabilität von Energieübertragungssystemen [22]
(in English: Dynamics and stability of energy transfer systems)
Rehtanz, TU Dortmund, 2022

In the lectures, Rehtanz discusses the stability criteria and presents their calculations. Among others, Rehtanz covers the criteria for static voltage stability, which differ from those in the report by DIgSILENT GmbH. The criterion by Rehtanz for static voltage stability is that the eigenvalues of J_R are positive, which describes the procedure for modal analysis. Rehtanz uses for the determination of matrix J_R the individual differentials, which is formed by the Jacobian matrix. The Jacobian matrix required for the static voltage stability evaluation is generated in the Newton-Raphson load flow calculation. The relation $dV/dQ = J_\mathrm{R}^{-1}$ for the static stability criterion of Rehtanz provides an option for using PowerFactory with this criterion, as the programme does not output the Jacobian matrix.

Analysis of the Impact of Reactive Power Control on Voltage Stability in Transmission Power Grids [6]
Rengin İdil Cabadağ, PhD-Thesis, 2019

In the PhD-Thesis from Cabadağ, the criterion from [25] is confirmed, specifies the statement as sensitivity analysis and refers to the original source [20]. There is also confirmation of the conversion of dV/dQ to J_R. In addition to this approach to determining the stability criterion, the possibility of determining the criterion using the P-V and P-Q curves or using modal analysis is also mentioned. In order to optimise the static voltage stability, particle swarm optimisation (PSO) is also applied to an IEEE 24 RTS. PSO is used to determine optimal settings for controller parameters of power factor correction equipment. Each agent in PSO is optimizing a specific objective function using information based on the swarm in a multidimensional search space. One result of the thesis is that the use of PSO results in increased grid stability during voltage changes. The PSO approach is not relevant to this thesis, but to the prospect of a possible application (see chapter 7).

Master Thesis: Analyse des Einflusses verschiedener Arbeitspunkte auf die Wirkleistungsabhängige Spannungssensitivität an einem Mittelspannungsnetzknoten [12]
(in English: Analysis of the influence of different operating points on the active power-dependent voltage sensitivity at a medium-voltage grid bus)
Hartweg, Ostbayerische Technische Hochschule Regensburg, 2021

In the master thesis from Hartweg, the influence of the active power dependent voltage sensitivity dV/dP is shown. Hartweg gives the derivation of this sensitivity and gives an example of the use of the PowerFactory API (Application Programming Interface) to access the sensitivity via Python. Both were an important base for the understanding as well as for the implementation of the dV/dQ sensitivity.

2.3 Application of Artificial Intelligence in Load Flow Calculation and Stability Analysis

Conference: Application of artificial intelligence in power grid state analysis and -diagnosis [23]
Winter Et al., NEIS Conference Hamburg 2020, 2022

The authors present the basics of the application of the new technology artificial intelligence (AI) based on machine learning (ML) for load flow calculation and grid state diagnosis. The described AI model considers different load states in an LV grid. Synthetic generated cases are created by a case generator and solved by the load flow calculation according to the Newton-Raphson method with the program ATPDesigner. Load profiles of assumed houses, photovoltaic systems and electric vehicles in a LV grid are assumed. The input is PQ or slack bus depending on the grid. Additionally, a grid state diagnosis is performed. This procedure generates training data. In the paper the basic algorithms of the AI/ ML are described, the structure of the neural networks (ANN). The next section describes how a multi-layer-percepton (MLP) was trained with the feed forward network using the Python library PyTorch with this data. The MLP is supposed to estimate the load flow calculation and grid state and was validated on a test grid. In the end, the potential for optimization in hyperparametrization is noted.

In the following years, the same authors published further papers, in which the handling of missing data for the AI model [24] was investigated. In another paper, the methodology used for the hyper parametrization [30] was described. In [1], an algorithm for grid optimization is presented, which represents another application of AI.

2.4 Summary and Identification of Research Gaps

The presented documents show that these are known problems that can be solved with previous approaches. To distinguish this work from the previous publications from Winter Et al. [23], there has been no development that deals with static voltage stability. However, the publications have provided a useful base for this thesis to estimate the amount of training data required and which methods could be successful. For simplicity, Keras is used instead of PyTorch, as it is part of TensorFlow and offers therefore a wide range of integrated features.

In addition, the grid topology is different as it uses a PV bus. The LV grid used by Winter et al. did not include PV buses, as PV buses are not common in LV grids.

3 Fundamentals of the Topic

3.1 Electrical Grids

The electricity power grid is used to transmit and distribute electrical energy from generators to consumers. This is generally done in alternating current (AC) grids and since direct current (DC) grids occur only in specific applications, there are not look further in this thesis.

Generators describe devices that convert non-electrical energy into electrical energy. These have different primary energy sources, which are fossil or renewable. In the context of the German and global energy transition, there is a strong increase in renewable energy. The motivation for this is climate change, which is accelerated by the emissions from fossil energy usage. This has also led to changes on the consumer side. Examples are the installation of heat pumps or the substitution of fossil powered vehicles by electrical powered vehicles, which need corresponding loading facilities, as a connecting element to the grid. This action will replace fossil energy sources. The whole process is facilitated by the installation of energy storage systems in various forms of energy. This can subsequently be transformed into electrical energy and distributed to consumers. The transmission of the electrical energy takes place in several voltage levels depending on the distance to be covered and the power quantity. For long distances and large amounts of energy, HV is selected to minimize losses. Below this level, MV is used, f.e. in regional or urban areas. Consumers such as households are connected to the LV grid. This voltage level was chosen because, among other things, the electrical flashover is lower at LV. Transformers are utilised to connect grids with varying voltage levels, resulting in a voltage increase or decrease through coiling differences. Based on the amount of power being supplied and consumed in the area, a load flow is established within the interconnected power system. Consequently, certain lines may become more heavily loaded, hence the necessity for grid regulation to prevent overloading.

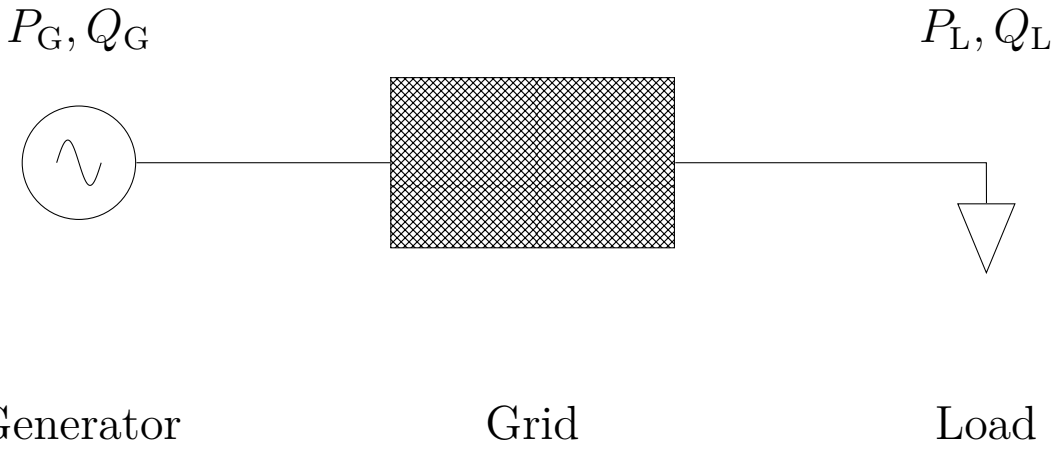


Fig. 3.1: Strongly Simplified Structure of a Power Grid

Overloading is indicated by exceeding the rated current of a line. High currents cause the line to heat up more and consequently lead to higher losses in the distribution. The current can be determined by applying Ohm's law to alternating current. For the complex notation Eq. 3.1 is given. Furthermore, this can be determined with the correlation between impedance and admittance.

$$\underline{Z} = \underline{V}/\underline{I} = 1/\underline{Y} \tag{3.1}$$

The impedance can be expressed in terms of power, in cases where the voltage, resistance or impedance/admittance of the line is not measured (see Eq. 3.2).

$$\underline{S} = \underline{V} \cdot \underline{I}^* = \underline{Z} \cdot |\underline{I}|^2 \tag{3.2}$$

The distribution grid can be well represented according to the graph theory with its connections in an admittance matrix (or nodal admittance matrix, Y bus matrix) $\underline{\boldsymbol{Y}}_{\mathrm{M}}$. The size of the quadratic matrix (n x n) is influenced by the number of buses n. An example of the form of a Y bus matrix is given in section G.3. This is formed by the diagonal (Eq. 3.3) and non-diagonal (Eq. 3.5) components. The diagonal component are determined by the sum of all admittances of the lines connected to the bus and the sum of the susceptance $B^{'}$ of the 1,2 sequence (or according to Kremens et al. [26] $\underline{y}_{ij}^{'}/2$) of the connected lines. The notation x_j describes the bus connected to the bus i. In summary, $\underline{Y}_{\mathrm{M}_{ii}}$ is given by Eq. 3.3 with Eq. 3.4. The non-diagonal components are determined according to Eq. 3.5. $\underline{Y}_{ij}$ describes the admittance of the connected line. For non-connected components, zero is inserted.

$$\underline{Y}_{\mathrm{M}_{ii}} = \underline{Y}_i^{'} + \sum_{k=x_j} \underline{Y}_{ik} \tag{3.3}$$

$$\underline{Y}_i' = \sum_{k=x_j} \frac{\underline{y}_{ik}^{'}}{2} \tag{3.4}$$

$$\underline{Y}_{\mathrm{M}_{ij}} = -\underline{Y}_{ij} \tag{3.5}$$

The basis for the admittance values are the values for the lines given in section 4.1. In the further course of the master thesis, the angle or the magnitude of the admittance is required, which is determined according to Eq. 3.6.

$$\underline{Y}_{ij} = Y_{ij} \cdot \mathrm{e}^{\mathrm{j} \cdot \mu_{ij}} \tag{3.6}$$

With the admittance matrix and the current, the voltage can be determined. However, the load flow calculation is necessary to determine the currents, as the current is not present in the synchronised state, which is explained in the following section section 3.2.

3.2 Load Flow Calculations

Load flow calculation can be used to determine voltages and possible overloads in the grid (see section 3.1). Applying the Gauss-Seidel or Newton-Raphson method presented below, the unknown values can be determined iteratively. In this way, overloads in the grid or critical voltages can be determined before the operating control is changed. Furthermore, there are a number of accompanying measures to adjust grid stability, as these must also be ensured for reliable grid operation (see section 3.3) [25].

The core function of PowerFactory is performing a load flow calculation using the Newton-Raphson method. Additionally, PowerFactory can be customized for grid planning, operation management, fault analysis and optimizing grid usage.

The load flow calculation is necessary as not all values are given. Depending on the input at the buses, different adoptions in the Gauss-Seidel and Newton-Raphson methods are necessary. The inputs used are the real power P and the reactive power Q, the magnitude of the voltage V and the voltage angle δ. With the real power P and the reactive power Q the apparent power can be determined by Eq. 3.7. The complex voltage is defined by the magnitude of the voltage and the voltage angle δ (see Eq. 3.8). Based on the given quantities, the buses are named correspondingly. The different bus types are listed in Tab. 3.1.

$$\underline{S} = P + \mathrm{j} \cdot Q \tag{3.7}$$

$$\underline{V} = V \mathrm{e}^{\mathrm{j} \cdot \delta} \tag{3.8}$$

Tab. 3.1: Types of Buses for Load Flow Calculation [26, 12]

label	**given variables**	**missing variables**	**application**	**note**
slack	V,δ	P,Q	reference bus	could also work as external grid,which in this case provides the frequency stability
PQ bus	P,Q	V,δ	generator or consumer	
PV bus	P, V	Q,δ	generator or consumer	Bus with high reactive power reserve or receiving bus for reactive power compensation

The benefit of load flow calculation based on a grid with only PQ buses is that there is no need for time-consuming synchronisation of measured values. Numerous buses lack synchronised measurements, particularly in LV grids, where the generators can be designated as PQ buses as they typically lack voltage control.

3.2.1 Gauss–Seidel Method

The Gauss-Seidel method is available in 2 variations, which are used to calculate to a defined convergence limit: Firstly with the impedance and secondly with the admittance. According to Kremes et. al. [26] the calculation time for one step is faster with the admittance variant than the impedance variant, depending on the number of buses. This is also the case for the total calculation time, i.e. the calculation of the convergence limit according to the method and buses. [26]

For this reason, only the admittance variant will be discussed in the following section. The principles for a grid with a PQ bus and a slack bus will be described first.

Grid without PV bus:
A complex voltage vector $\underline{V}$ is created initially, which is considered to be the starting value. Then a new voltage is determined with the voltage vector, the given real and reactive power P and Q and the admittance matrix $\underline{Y}_{\mathrm{M}}$ according to Eq. 3.9. This process is repeated for all voltages, except the slack voltage, where the voltage remains constant, prior to commencing the subsequent iterative phase. The iteration continues until convergence is achieved or a convergence criterion is fulfilled. In the absence of convergence, the case may be deemed unstable.

$$\underline{V}_i^{(v+1)} = \frac{(P - \mathrm{j}Q)^*}{\underline{Y}_{ii} \cdot \underline{V}_i^{(v)*}} - \sum_{j=1, j\neq i}^{w} \left(\frac{\underline{Y}_{\mathrm{ji}} \cdot \underline{V}_i^{(v)}}{\underline{Y}_{ii}} \right) \tag{3.9}$$

Grid with PV bus:
In the case of one or more PV buses, a value for the reactive power Q determined according to Eq. 3.10 must be determined in advance. For this, the magnitude of the voltage V_k of the PV bus must first be determined according to the control of the PV bus.

$$Q_k^{(v+1)} = -\sum_{j=1}^{w} V_k \cdot V_j \cdot Y_{kj} \cdot \sin\left(\mu_k + \delta_{\mathrm{j}} - \delta_k\right) \tag{3.10}$$

Subsequently, the calculated reactive power Q is inserted into Eq. 3.9, which corresponds to Eq. 3.11. The index k describes the position of the PV bus.

$$\underline{V}_k^{(v+1)} = \frac{(P - \mathrm{j} \cdot Q_k^{(v+1)})^*}{\underline{Y}_{ii} \cdot \underline{V}_k^{*(v)}} - \sum_{j=1, j\neq i}^{w} \left(\frac{\underline{Y}_{\mathrm{ji}} \cdot \underline{V}_k^{(v)}}{\underline{Y}_{ii}} \right) \tag{3.11}$$

Finally, only the angle δ of the complex voltage on the PV bus is updated. The magnitude of the voltage V remains according to the defined initial value. The determination of the reactive power of the PV bus is repeated in each iterative step.

3.2.2 Newton-Raphson Method

Compared to the Gauss-Seidel method, the Newton-Raphson method admittance variant is slower per iteration step but requires a fraction of the iteration steps until convergence, i.e. it requires a shorter total computing time [26]. On the other hand, the creation of the Jacobian matrix is significantly more complex and consequently requires more memory.

Creation of the Jacobian matrix:
The Jacobian matrix $\boldsymbol{J}$ consists of the differentials of the values necessary for the calculation (cf. Gauss-Seidel method). The structure of the matrix can be seen in Eq. 3.12. To make these differentials and their placement in the matrix more understandable, these are described as $\boldsymbol{J}_1$ to $\boldsymbol{J}_4$ (see Eq. 3.13). The iteration counter v is not included. The Jacobian matrix $\boldsymbol{J}$ is also called a functional matrix $\mathbf{F}$ in other sources [12, 27].

$$\boldsymbol{J} = \left[\begin{array}{ccc|ccc} \frac{\partial P_1}{\partial \delta_1} & \cdots & \frac{\partial P_1}{\partial \delta_{\mathrm{n}}} & \frac{\partial P_1}{\partial V_1} & \cdots & \frac{\partial P_1}{\partial V_{\mathrm{n}}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial P_{\mathrm{n}}}{\partial \delta_1} & \cdots & \frac{\partial P_{\mathrm{n}}}{\partial \delta_{\mathrm{n}}} & \frac{\partial P_{\mathrm{n}}}{\partial V_1} & \cdots & \frac{\partial P_{\mathrm{n}}}{\partial V_{\mathrm{n}}} \\ - & - & - & - & - & - \\ \frac{\partial Q_1}{\partial \delta_1} & \cdots & \frac{\partial Q_1}{\partial \delta_{\mathrm{n}}} & \frac{\partial Q_1}{\partial V_1} & \cdots & \frac{\partial Q_1}{\partial V_{\mathrm{n}}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Q_{\mathrm{n}}}{\partial \delta_1} & \cdots & \frac{\partial Q_{\mathrm{n}}}{\partial \delta_{\mathrm{n}}} & \frac{\partial Q_{\mathrm{n}}}{\partial V_1} & \cdots & \frac{\partial Q_{\mathrm{n}}}{\partial V_{\mathrm{n}}} \end{array}\right] \tag{3.12}$$

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial P}{\partial \delta} & \frac{\partial P}{\partial V} \\ \frac{\partial Q}{\partial \delta} & \frac{\partial Q}{\partial V} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_1 & \boldsymbol{J}_2 \\ \boldsymbol{J}_3 & \boldsymbol{J}_4 \end{bmatrix} \tag{3.13}$$

The differentials are calculated as below. The index w describes the number of buses and for comprehensibility, the index k was used instead of j in the summation signs. The calculation of the diagonals is described first and then the non-diagonals.

$\frac{\partial P}{\partial \delta}$ corresponds to $\boldsymbol{J}_1$

$$\frac{\partial P_i}{\partial \delta_i} = -\sum_{k=1, k\neq i}^{w} Y_{ik} \cdot V_k \cdot V_i \cdot \sin(\delta_i - \delta_k - \mu_{ik}) \tag{3.14}$$

$$\frac{\partial P_i}{\partial \delta_j} = Y_{ij} \cdot V_j \cdot V_i \cdot \sin(\delta_i - \delta_j - \mu_{ij}) \tag{3.15}$$

$\frac{\partial P}{\partial V}$ corresponds to $\boldsymbol{J}_2$

$$\frac{\partial P_i}{\partial V_i} = 2 \cdot V_i \cdot Y_{ii} \cdot \cos(\mu_{ii}) + \sum_{k=1, k\neq i}^{w} Y_{ik} \cdot V_k \cdot \cos(\delta_i - \delta_\mathrm{k} - \mu_{\mathrm{ik}}) \tag{3.16}$$

$$\frac{\partial P_i}{\partial V_j} = Y_{ij} \cdot V_i \cdot \cos(\delta_i - \delta_i - \mu_{ij}) \tag{3.17}$$

$\frac{\partial Q}{\partial \delta}$ corresponds to $\boldsymbol{J}_3$

$$\frac{\partial Q_i}{\partial \delta_i} = \sum_{k=1, k\neq i}^{w} Y_{\mathrm{ik}} \cdot V_\mathrm{k} \cdot V_i \cdot \cos(\delta_i - \delta_\mathrm{k} - \mu_{\mathrm{ik}}) \tag{3.18}$$

$$\frac{\partial Q_i}{\partial \delta_j} = -Y_{ij} \cdot V_i \cdot V_i \cdot \cos(\delta_i - \delta_j - \mu_{\mathrm{ik}}) \tag{3.19}$$

$\frac{\partial Q}{\partial V}$ corresponds to $\boldsymbol{J}_4$

$$\frac{\partial Q_i}{\partial V_i} = -2 \cdot V_i \cdot Y_{\mathrm{ii}} \cdot \sin(\mu_{\mathrm{ik}}) + \sum_{k=1, k\neq i}^{w} Y_{\mathrm{ik}} \cdot V_\mathrm{k} \cdot \sin(\delta_i - \delta_k - \mu_{\mathrm{ik}}) \tag{3.20}$$

$$\frac{\partial Q_i}{\partial V_j} = Y_{ij} \cdot V_i \cdot \sin(\delta_i - \delta_i - \mu_{ij}) \tag{3.21}$$

Grid without PV bus:
First, the vector of the current $\underline{\boldsymbol{I}}$ according to Eq. 3.22 is determined with the existing Y bus matrix $\underline{\mathbf{Y}}_\mathrm{M}$. The current can then be used to determine the matrix of the calculated apparent power $\underline{\boldsymbol{S}}_{\mathrm{cal}_i}^{(v)}$ according to Eq. 3.23.

$$\underline{\boldsymbol{I}}^{(v)} = \underline{\mathbf{Y}}_\mathrm{M} \cdot \underline{\boldsymbol{V}}^{(v)} \tag{3.22}$$

$$\underline{\boldsymbol{S}}_\mathrm{cal}^{(v)} = \mathrm{diag}(\underline{\boldsymbol{V}}^{(v)}) \cdot \underline{\boldsymbol{I}}^{(v)^*} \tag{3.23}$$

From the given apparent power, which is determined according to Eq. 3.7, the calculated apparent power $\underline{\boldsymbol{S}}_\mathrm{cal}^{(v)}$ is subtracted, which gives the difference of the apparent power $\Delta\underline{\boldsymbol{S}}^v$ according to Eq. 3.24.

$$\Delta\underline{\boldsymbol{S}}^{(v)} = \underline{\boldsymbol{S}}^{(v)} - \underline{\boldsymbol{S}}_\mathrm{cal}^{(v)} \tag{3.24}$$

The columns and rows that have the notation x of the slack bus are removed from the Jacobian matrix, resulting in a modified Jacobian matrix. The rows and columns to be removed are x and $x + w$. Thus the modified Jacobian matrix $\boldsymbol{J}_{\mathrm{M}}$ has the difference of the apparent power $\Delta\underline{\boldsymbol{S}}^{(v)}$ off-settable dimension, since it does not contain the slack bus.

In the next step, the difference of the voltage $\Delta\underline{\boldsymbol{V}}^{(v)}$ is determined according to Eq. 3.25. In the notation in which matrices are multiplied, the active and reactive power (P & Q), along with the magnitude of the voltage (V) and the voltage angle (δ) are separate (see Eq. 3.8) and stacked on top of each other, it looks like in Eq. 3.26.

$$\Delta\underline{\boldsymbol{V}}^{(v)} = \boldsymbol{J}_{\mathrm{M}}^{(v)^{-1}} \cdot \Delta\underline{\boldsymbol{S}}^{(v)} \tag{3.25}$$

$$\begin{pmatrix} \Delta\delta^{(v)} \\ \Delta V^{(v)} \end{pmatrix} = \boldsymbol{J}_{\mathrm{M}}^{(v)^{-1}} \cdot \begin{pmatrix} \Delta P^{(v)} \\ \Delta Q^{(v)} \end{pmatrix} \tag{3.26}$$

The voltage of each PQ bus is calculated using Eq. 3.27. This equation results in the development of the Taylor series to the linear element on which the Newton-Raphson method is based. The Taylor series is an infinite series that approximates a function at a specific point. In load flow analysis, the power inequalities are linearised using the first-order Taylor series. [27] The voltage angle δ and the magnitude of the voltage V are determined for the voltage $\underline{V}$ and the voltage difference $\Delta\underline{V}$ according to Eq. 3.8.

$$\underline{V}_i^{(v+1)} = (V_i^{(v)} + \Delta V_i^{(v)}) \cdot \mathrm{e}^{\mathrm{j}\cdot\delta_i^{(v)} + \mathrm{j}\cdot\Delta\delta_i^{(v)}} \tag{3.27}$$

Finally, the complete Jacobian matrix J must be determined with the new values of the voltage $\underline{V}$ for the next iteration step, if this is not done at the beginning of each step. The iteration process is continued until a convergence criterion is reached, which corresponds to the linear equilibrium of the Taylor series. [27] In addition to these methods, there are other variations of the Newton-Raphson method, for instance, using a simplified Jacobian matrix or a constant for acceleration.

Grid with PV bus:
The calculation is performed as described above. In addition to the slack, the row and column of the magnitude of the voltage V_x of the PV bus are also removed. The index x represents the position of the PV bus. This means that no value is determined for $\Delta\mathbf{V_x}$. This is set as zero, which does not adjust the magnitude of the voltage, but the angle of the voltage at the PV bus. According to Eq. 3.27, the following Eq. 3.28 results for the PV bus.

$$\underline{V}_x^{(v+1)} = V_x \cdot \mathrm{e}^{\mathrm{j}\cdot\delta_x^{(v)} + \mathrm{j}\cdot\Delta\delta_x^{(v)}} \tag{3.28}$$

3.3 Stability Indicator

The operation of a power grid follows certain requirements to avoid instabilities. In addition to the operation of the grid, instabilities could also affect the consumers who do not receive electricity and the electricity producers who have to adjust their generation. The following requirements for stability are named by Rehtanz [22]:

1. primary requirement
    - frequency stability
    - generation equals load
2. secondary requirement
    - $V_{\min} \leq V \leq V_{\max}$ (voltage range)

The main focus of primary stability is to maintain a narrow frequency range, typically around 50 Hz in the continental European synchronous grid (UCTE grid), in order to ensure that the power generated is equal to the load. The UCTE grid contributes to compliance with this requirement, as it can absorb or compensate for major over and under capacities. [5] The secondary requirement is the maintenance of the voltage in a certain voltage range. DIN EN 50160 requires that 95 % of the time in a weekly interval shall not deviate more than 10 % of the nominal voltage V_{n} (see 3.29). If the voltage exceeds this value, it is referred to as over voltage. This violation of this regulation, is called voltage range deviation [21, 9] or as in this thesis voltage contingencies [19].

$$V_{\mathrm{n}} \cdot 0.9 < V \leq V_{\mathrm{n}} \cdot 1.1 \tag{3.29}$$

Strong deviations from this voltage range indicate instability. The power feed-in and the grid are regulated according to the first and second requirements.

In order to provide an exact definition of the term instability, the following general definition of stability by Rehtanz is quoted here in translated form, which is the opposite of the term:

> "An electrical power system is stable for a given disturbance if, after a finite time, it reaches a steady state that is acceptable from an operational point of view.
>
> Note 1: The new state reached need not necessarily be identical to the initial state.
> Note 2: In this context, acceptable means that none of the operating limits may be violated, so that no chain reaction can be triggered and a sometimes slow subsequent disturbance can develop." [22]

A system remains stable as long as the current state does not give rise to any unstable developments. In this case, the system dynamics remain insufficient to lead to an unstable state. [25]

When examining the stability of the original stable state, it is important to take into account structural or parametric modifications that may also impact the stable state, according to the citation. Instances of this include (n-1) situations, changes in operating conditions that push them beyond their permissible ranges or changes in the parameters of operational equipment. [25]

Another example of instability arises from faults, like the failure of a line. Such faults lead to a violation of the stability requirements, which consequently prompts the implementation of safeguards and control mechanisms to restore stability.

In order to analyse this multitude of influencing factors that contribute to stability or instability more precisely, the term stability has been divided into suitable categories. According to Rehtanz, there are the following types of stability [22]:

1. frequency stability
2. resonance stability
3. inverter-related stability
4. rotor angle stability
    - static (small disturbance)
    - tranisent (large disturbance)
5. voltage stability
    - static (small disturbance)
    - dynamic (large disturbance)

The frequency stability corresponds to the criteria mentioned above, i.e. maintaining the nominal frequency, especially during periods of fluctuations in load behaviour or outages. In the event of deviation, it is influenced by the primary control, whereby the control generators available in the grid intervene. In the secondary control, deviations from the nominal value in the affected control region are influenced in accordance with the secondary requirement. [5]

The inverter-related stability and resonance stability are new types of stability that were added in 2020 [22]. The inverter-related stability describes the instability caused by power converters due to the incorrectly designed control or due to unpredictable rebound effects with the grid. Resonance stability describes the damping or overlapping of the nominal frequency of the grid with the frequency from disturbances or excitations.

The rotor angle stability describes the ability of the synchronous machines in the grid to remain in a synchronous state. A distinction is made between the static rotor angle stability with a small disturbance and the transient rotor angle stability with a large disturbance. In contrast to the static disturbance, the transient disturbance is a state that can only be kept stable by a control process and not by the grid itself. [5]

Voltage stability is divided into two sub-types. The static small disturbance and the dynamic large disturbance. These large disturbances can be caused, f.e. by load shedding, loss of a generator or fault in the grid which causes the voltage to rise or fall uncontrollably. This again requires major intervention by the control system. Small disturbances are f.e. the load changes. Furthermore, according to [5], the short-term and long-term voltage stability is also discussed. It describes the time required to restore the voltage. In the case of short-term, immediately by regulation and in the case of long-term, in a period of 30 s to 30 min. Based on this, the grid can be assumed to be stable if the static stability is fulfilled, but can still not lead to instability if it is not fulfilled. However, a steady state is to be attempted.

In [27] the description of the static voltage stability and the static angular stability is used under the generic term static stability. However, a clear distinction must be made between this and the static stability of generators. In this context, it describes the range in which synchronous generators work well.

In [27] an indicator for the static voltage instability and the static rotor angle instability is also introduced. This is the determinant of the Jacobi/functional matrix which approaches zero.

In [25] the necessary test criteria for the grid development plan are mentioned. In order to fulfil all criteria, a dynamic grid model is also necessary in addition to the dynamic grid model. The NEP analysis basically concentrates on the grid characteristics resulting from the established grid connection rules with regard to frequency and voltage stability. The source states that the static voltage stability is fulfilled if the dV/dQ - sensitivity is positive in each grid bus. [25]

In [6] it is explained in more detail what is intended by the term "in each grid bus", taken from the diagonal of the dV/dQ - sensitivity matrix. This type of analysis s described as a sensitivity analysis. Furthermore, [20] states:

> "A positive V-Q sensitivity is indicative of stable operation; the smaller the sensitivity, the more stable the system. As stability decreases, the magnitude of the sensitivity increases, becoming infinite at the stability limit. Conversely, a negative V-Q sensitivity is indicative of unstable operation" [20]

The dV/dQ - sensitivity can be determined according to Eq. 3.34. The term dV/dQ describes the ratio $\Delta V/\Delta Q$. The derivation of this equation is described in Eq. 3.30 to Eq. 3.32.

$$\begin{pmatrix} \Delta\boldsymbol{\delta} \\ \Delta\boldsymbol{V} \end{pmatrix} = \boldsymbol{J}^{-1} \cdot \begin{pmatrix} \Delta\mathbf{P} \\ \Delta\mathbf{Q} \end{pmatrix} \tag{3.30}$$

$$\boldsymbol{J}^{-1} = \frac{1}{\det(\boldsymbol{J})} \cdot \begin{pmatrix} \boldsymbol{J}_4 & -\boldsymbol{J}_2 \\ -\boldsymbol{J}_3 & \boldsymbol{J}_1 \end{pmatrix} \tag{3.31}$$

$$\det(\boldsymbol{J}) = \boldsymbol{J}_1 \cdot \boldsymbol{J}_4 - \boldsymbol{J}_2 \cdot \boldsymbol{J}_3 \tag{3.32}$$

The active power variation is assumed to be $\Delta P = 0$. This results Eq. 3.33 or Eq. 3.34.

$$\Delta V_i = \frac{J_{1_{ij}}}{J_{1_{ij}} \cdot J_{4_{ij}} - J_{2_{ij}} \cdot J_{3_{ij}}} \cdot \Delta Q_j \tag{3.33}$$

$$\frac{\Delta V_i}{\Delta Q_j} = J_{R_{ij}}^{-1} = [J_{4_{ij}} - J_{3_{ij}} \cdot J_{1_{ij}}^{-1} \cdot J_{2_{ij}}]^{-1} \tag{3.34}$$

$$\frac{\Delta V}{\Delta Q} = J_{\mathrm{R}}^{-1} = [J_4 - J_3 \cdot J_1^{-1} \cdot J_2]^{-1} \tag{3.35}$$

Rehtanz states in [22] as a criterion for static voltage stability that the eigenvalues of the matrix J_{R} must be greater than zero (see Eq. 3.36). This describes the procedure for modal analysis. The document in section G.4 shows that modal analysis and dV/dQ - sensitivity analysis are contradictory. Problems with non-diagonal J_{R}-matrices in modal analysis were also noted in [6]. For this reason, the criterion according to PowerFactory [25] (dV/dQ - sensitivity analysis) is used in this thesis.

$$\text{eigenvalues}(J_{\mathrm{R}}) > 0 \tag{3.36}$$

This thesis deals with the previously mentioned area, partly because the fundamental calculation for the load flow calculation only allows this static consideration. Furthermore, the frequency stability is influenced by the overlying grid, which here corresponds to the slack bus.

The assessment of system stability necessitates a model-based analysis of the system. Since the control system relies on fast analysis, this work investigates the extent to which AI can be trained to evaluate static voltage based on model-based data.

3.4 Artificial Intelligence and Neural Networks

Artificial intelligence (AI) has made great progress in recent years. The application of AI is becoming more widespread. In this thesis, the term AI is often used, although the more precise and AI-subordinate term machine learning (ML) would be more appropriate, as it describes the ability of machines to learn using statistical methods.

A new mathematical relationship is applied to behavioural models based on input and output, in contrast to system models where a mathematical model exactly reproduces the real steps of the system. A typical behavioural model, in addition to fuzzy logic, is the artificial neural network (ANN). This modelling of the behaviour model is called soft computing and is particularly suitable for the modelling of non-linear complex multivariable systems. The system must be trained to perform this task. [18]

The focus is on the fact that the model does not learn the information by heart, but learns the rules behind it, so-called competence learning or eager learning, in order to be able to generalise with these learned rules so that they can also be reliably applied to unknown values. The output can be either a classification (f.e. an assignment to a class) or an approximation of real numbers (f.e. an assignment to an infinite number of classes). Methods based on memorization with generalization ability are referred to as lazy learning or memory-based learning. [14]

3.4.1 Single-layer Perceptron

To understand how AI works and how (machine) learning works, the observation of a single neutron in a single-layer perceptron (SLP) can be used to provide an explanation. An SLP describes a simplified ANN, see Fig. 3.2. The perceptron was developed in 1957 by Frank Rosenblatt and describes an electrical device that works according to the biological model of a neuron. [4]

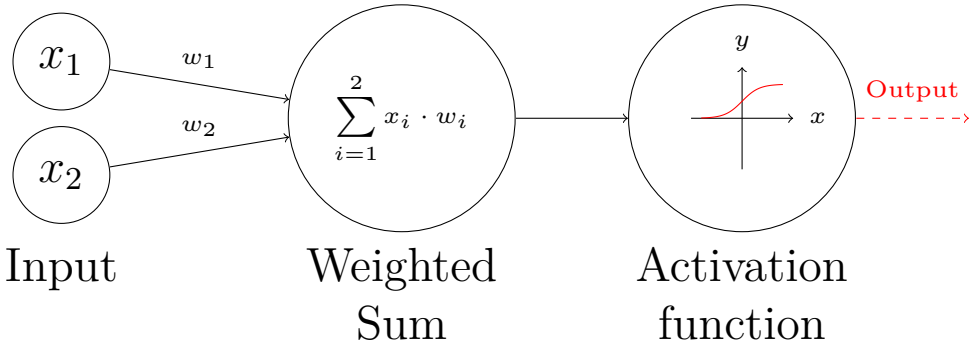


Fig. 3.2: Single-Layer Perceptron (SLP) with Sigmoid Activation Function

Initially, the neuron receives a signal as input. The values with which the model is trained are called features (feature vectors).

Then a weighted sum is formed from the feature vector and the initialised random weights w, as shown in Eq. 3.37 for two input neurons. The result is called the activity a_j. This activity serves as input for an activation function f_{act}. Depending on the selected function, this function outputs a signal o_j (see Eq. 3.38). Furthermore, by adding a threshold ϑ_k(see Eq. 3.39) the activity can be shifted (bias). This is an alternative to changing the threshold value in the unit step activation functions, which is not possible with the other activation functions.

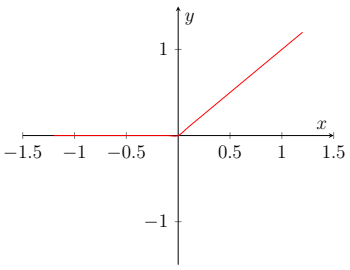
$$a_j = \sum_{i=1}^{2} x_i \cdot w_i \tag{3.37}$$

$$o_j = f_{\text{act}}(a_j) \tag{3.38}$$

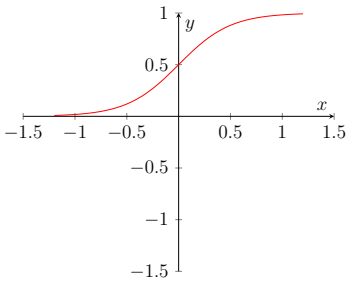
$$a_j = \left(\sum_{i=1}^{2} x_i \cdot w_i \right) + \vartheta_j \tag{3.39}$$

An example of an activation function is the "sigmoid" activation function, which produces a binary output between zero and one. This signal can be excitatory, i.e. it activates the following neuron or inhibitory, i.e. it is less likely to activate the neuron and produce a signal. Whether a signal is excitatory or inhibitory depends on the upstream activating function. When a neuron experiences low activation, the application of the sigmoid function causes the neuron's output to be close to zero.

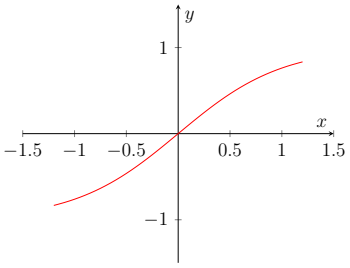
An overview of the activation functions is given in Fig. 3.3. The equation of the functions is shown in Tab. 3.2.



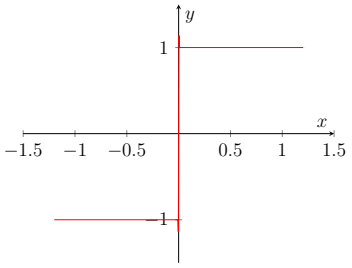
(a) ReLU (Rectified Linear Unit)



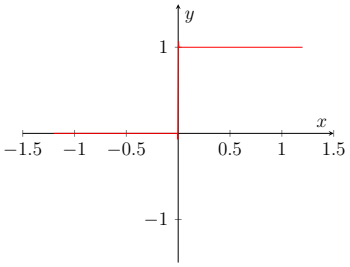
(b) Sigmoid



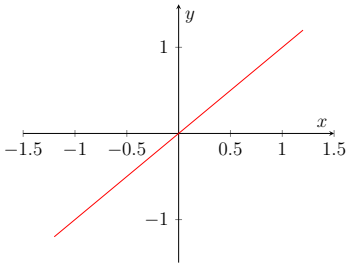
(c) Tanh (Hyperbolic Tangent)



(d) Unit Step Function (Threshold)



(e) Binary threshold function (Threshold=0)



(f) Lineare Funktion

Fig. 3.3: Activation Functions

The activity a_j is then compared to the target (classification) t_j of the feature vector. The feature vector is either binary or converted to binary. The learning rule specific to the perceptron is then applied, which only allows for a linear separation. If the target and the output signal match, the weights are not changed. The weight is changed if the prediction is incorrect. The weight is incremented, i.e. increased step by step, if the output signal is falsely zero (0). If the output signal is incorrectly one (1), the weights are decremented, i.e. they are minimised step by step. The adjustment is made in the direction of the steepest gradient of the function. [28] The new weight $w_{ij}^{(v+1)}$ follows after Eq. 3.40. The size of the step Δw_{ij} must be determined each time and follows from Eq. 3.41, whereby the value, i.e. the intensity of this change, can be adjusted by the learning rate α. The learning rate is $\alpha > 0$.

$$w_{ij}^{(v+1)} = w_{ij}^{(v)} + \Delta w_{ij} \tag{3.40}$$

Eq. 3.41 is also referred to as the Hebb rule.

$$\Delta w_{ij} = \alpha \cdot (t_j - o_j) \tag{3.41}$$

In this way, linear separable feature vectors can be classified into two classes with the SLP. Non-linear and differentiable activation functions are used for the MLP, as it is intended to solve non-linear problems.

Tab. 3.2: Activation Functions and Their Derivative [18]

name	**function**	**derivative**
ReLU (rectified linear unit)	$f_{\text{act}}(a) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$	1
sigmoid	$f_{\text{act}}(a) = \frac{1}{1+e^{a}}$	$f'_{\text{act}}(a) = f_{\text{act}}(a) \cdot [1 - f_{\text{act}}(a)]$
tanh (hyperbolic tangent)	$f_{\text{act}}(a) = \tanh(a) = \frac{e^{a}-e^{-a}}{e^{a}+e^{-a}}$	$f'_{\text{act}}(a) = \frac{1+f_{\text{act}}(a)}{1-f_{\text{act}}(a)}$
unit step function (threshold)	$f_{\text{act}}(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ -1 & \text{otherwise} \end{cases}$	0
binary threshold function	$f_{\text{act}}(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases}$	0
linear	$f_{\text{act}}(a) = a$	1

3.4.2 Multi-layer Perceptron

The multi-layer-percepton (MLP) consists of several layers, in accordance to the name. It consists of an input layer, an output layer and at least one hidden layers in between. The Fig. 3.4 illustrates the structure. Each layer consists of at least one artificial neutrons. These are connected to every neuron in the previous layer. The topology of the resulting ANN is fixed and, unlike its biological counterpart, there is no growth or decay of neurons. However, to optimise the topology, it is possible to use the pruning function in a separate step. Pruning is a method used to eliminate unimportant neurons in the network, i.e. to eliminate connections that are below a defined relevance. [18] The reason for this is that reduction avoids overfitting and the ability of the ANN to increase generalisation decreases with the number of connections. [28]

Relevance refers to the extent to which a connection of a specific weight influences the weighted sum formation (see Eq. 3.39). Weights with values near zero and those multiplied by close-to-zero activity have low relevance. The relevance can be calculated using the formula presented in Eq. 3.43 [28].

$$r_{\text{ij}}(v) = \frac{\tau_{\text{P}} \cdot r_{\text{ij}}(v-1) + s_{\text{ij}}}{\tau_{\text{P}} + 1} \tag{3.42}$$

$$s_{\text{ij}} = |w_{\text{ij}}| \tag{3.43}$$

The symbol τ_P refers to a time constant in epochs of the low pass filter. If the defined relevance threshold is not reached, connections below this value are eliminated. [28]

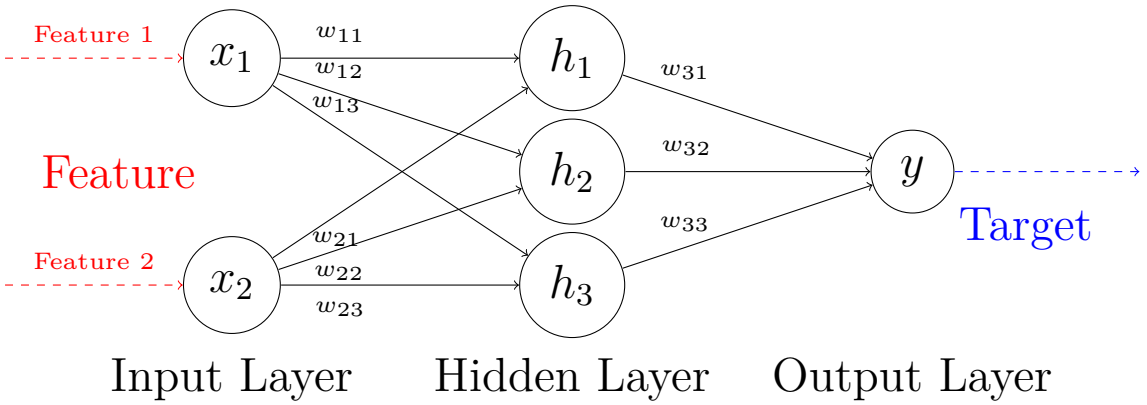
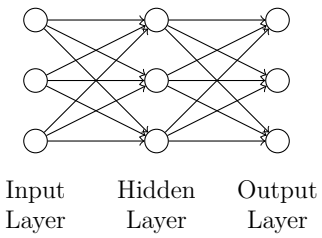


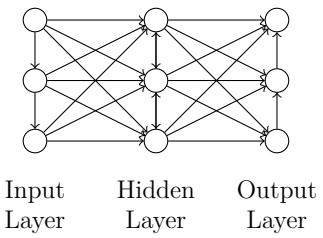
Fig. 3.4: ANN with Two Input Neurons, One Hidden Layer with Three Perceptrons and an Output Layer with One Neuron. The Weights Shown on the Connections.

There are two phases in the training of an MLP. The first is the forward phase, in which the initialised weights are used in the analogue to the perceptron and the signal is propagated in layers through the ANN in dependence on the activation function. Finally, an output is determined. [3]

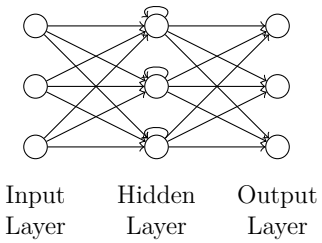
Next to the forward phase, various connection directions are distinguished (see Fig. 3.5). The transfer to the next higher layer is achieved through feed-forward, which proceeds from the input layer to the output layer. Lateral feedback denotes connections within layers, such as within hidden layers. Direct feedback, on the other hand, utilises the input value of a neuron directly as input again. Indirect feedback also links neurons in the previous layer. Additionally, there exist networks where all neurons are interconnected.



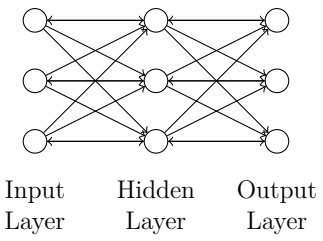
(a) Feed Forward Network



(b) Lateral Network



(c) Direct Feedback Network



(d) Indirect Feedback Network

Fig. 3.5: Different Network Architectures for Layered ANNs [18]

The measure of the difference between prediction and target is called loss, which denotes the deviation in the prediction. It is represented in numerical terms, with a high value indicating a poor model. Loss can be determined using a number of methods, one of which is a mean squared error (MSE). It calculates the average squared difference between the target and the predicted outcome. The MSE is an example of regression loss. It also includes the mean absolute error (MAE). In addition to regression loss, there are also classification losses. For binary classifications, binary cross entropy can be used. For more than two classifications, the categorical cross entropy is recommended.

In the second phase, the backward phase (backpropagation), the error signal from the output-to-target comparison is propagated back to the input layer, for each layer, adjusting the weights [3]. More specifically, this is achieved using an optimised algorithm to minimise the gradient between loss and weights. This process is iterated over multiple epochs leading to the reduction of loss. An epoch describes the use of the entire training dataset to improve the ANN by applying backpropagation. Stochastic gradient descent (SGD) is an instance of such an algorithm. It can be combined with a momentum term that considers past gradients. This improves optimization by overcoming the local minima, resulting in faster convergence. Similarly, there is RMSprop (Root Mean Square Propagation), which utilises a moving average of the previous squared gradients to dynamically adjust the learning rate for each weight. By incorporating concepts from Momentum and RMSprop, an optimizer called Adam (Adaptive moment estimation) is obtained. Adam calculates an exponential moving average of both the gradients and the squared gradients.[16]

The basic prerequisite for the gradient descent method is that the activation function (refer to Tab. 3.2) has an easily differentiated capability. [14]

Due to the limited interpolation capability of the MLP model, the trained MLP can only work in the already learned feature space. [18]

After introducing the general functions and terms of the SLP and MLP, the general process of an AI application can be described. This consists of these four steps:

1. Training/ learning
2. Validation
3. Test
4. Recall

1. Training/ learning:
Training or learning has already been described in this chapter, but the process is repeated over a number of epochs. The number of epochs can be fixed or the training can be terminated by a convergence criterion, like an error limit. A training dataset with classified data is used. The training accuracy or the RMSE of the training is used to evaluate the training. For classifications of two classes, accuracy describes the ratio of the sum of assigned true-negative and true-positive classifications to the sum of assigned true-positive, true-negative, false-positive and false-negative classifications. These values are often presented in the form of a confusion matrix. From this, further precision and recall can be evaluated. Since this type of evaluation requires a definition of the acceptable deviation, the RMSE is often used. This is the average of the deviations between prediction and actual value for the target of all patterns.

2. Validation:
A validation dataset is usually used in addition to the training dataset. It is important to note that the validation dataset does not include any data from the training dataset. The validation dataset size is typically 20% of the for the training dataset. During training, the validation accuracy of the applied AI with respect to the current epoch is reported based on the validation datasets. The validation accuracy is defined in the same way as the training accuracy, only applied to the validation dataset. This dataset is kept separate and is not included in the training process. Thus, it can be directly verified whether the model is memorising or effectively utilising interpolation/ generalization ability. In addition, it is possible to use the loss function of the validation dataset as a termination criterion if the loss increases. In some programs, this step is omitted (f.e. DataEngine), so that the test also fulfils this task.

3. Test:
The test dataset represents the first test of the artificial intelligence, since they are unknown datasets that have not indirectly influenced the learning process of the artificial intelligence (f.e. during validation, this indirect influence can be premature termination). After the evaluation, the model can be the subject of an iterative adaptation process. The dataset consists of classified data. [18]

4. Recall:
Recall describes the practical application of AI. It is often referred to as the online phase of AI. The used dataset consists of unclassified data, meaning no target value.

3.4.3 Other Neural Network Models

There are alternative ANNs to MLP/SLP that follow different learning procedures or alter the training process. In the following, examples are presented based on the classification of learning methods.

In supervised learning, the ANN gets trained on the feature vectors, with each feature having a target or being classifiable. For this reason, the dataset is referred to as a classified dataset. The goal is to learn a mapping between inputs and outputs to make accurate predictions for the target. [18] One of the most common methods of supervised learning is MLP as a feed-forward network (FFN) and backpropagation, which was introduced earlier in this chapter. [28]

Unsupervised learning is used when the ANN is presented with only features to train. The data has not been labelled, classified or categorised, thus requiring the ANN to independently determine the classification criteria based on patterns and structures within the data. [28] Consequently, the weights are modified accordingly. Two frequently employed methods of unsupervised machine learning are the Kohonen network and the fuzzy Kohonen network. The Fuzzy Kohonen Network is a variant of the Kohonen Network that permits the utilisation of fuzzy clustering. The Kohonen network is employed to reduce dimensions and cluster data. The procedure for the Kohonen network differs from the previous described AI application, therefore the procedure is described in more detail here. The neurons are arranged in a chain (1D), map (2D) or grid (3D). In contrast to MLP, the weights are formed between the input variables and not between neurons themselves. In this feature space, each neuron is mapped as a function of the weight in the feature space. The neuron that has the smallest distance to the input vector is called the "winning neuron". The goal of this fitting process is to minimise the distance between the winner neuron's weight vector and the input vector. By reducing these distances, dependencies are created in the feature space. Because of this self-organisation, Kohonen networks are also called self-organising maps (SOM). The learning process can be monitored by the average distance and the maximum distance depending on the iterations of this distance minimization. After the learning process is completed, the target is introduced by labelling. These targets are classifications and not approximations of values. In the map/grid overview, areas may become visible in which feature characteristics can be clearly assigned to a target. This can be evaluated by a test. [18]

In reinforcement learning, the ANN acts as an agent that receives rewards or punishments based on its strategy (policy). It adjusts its weights to learn a strategy that maximises cumulative rewards and improves its decision making over time.

Stochastic strategies involve using randomness to minimisation of an energy function. [28] The Boltzmann Machine (Hopfield Stochastic Network) is an example. [18] The Boltzmann machine is a type of ANN with indirect and direct feedback. It uses probability activation to model complex relationships in the data. Often used for optimisation.

4 Data Generation

4.1 Description of the 5 Bus System

The IEEE 5 bus grid (see Fig. 4.1) is comparable for benchmarking purposes and has been implemented and viewed many times elsewhere.

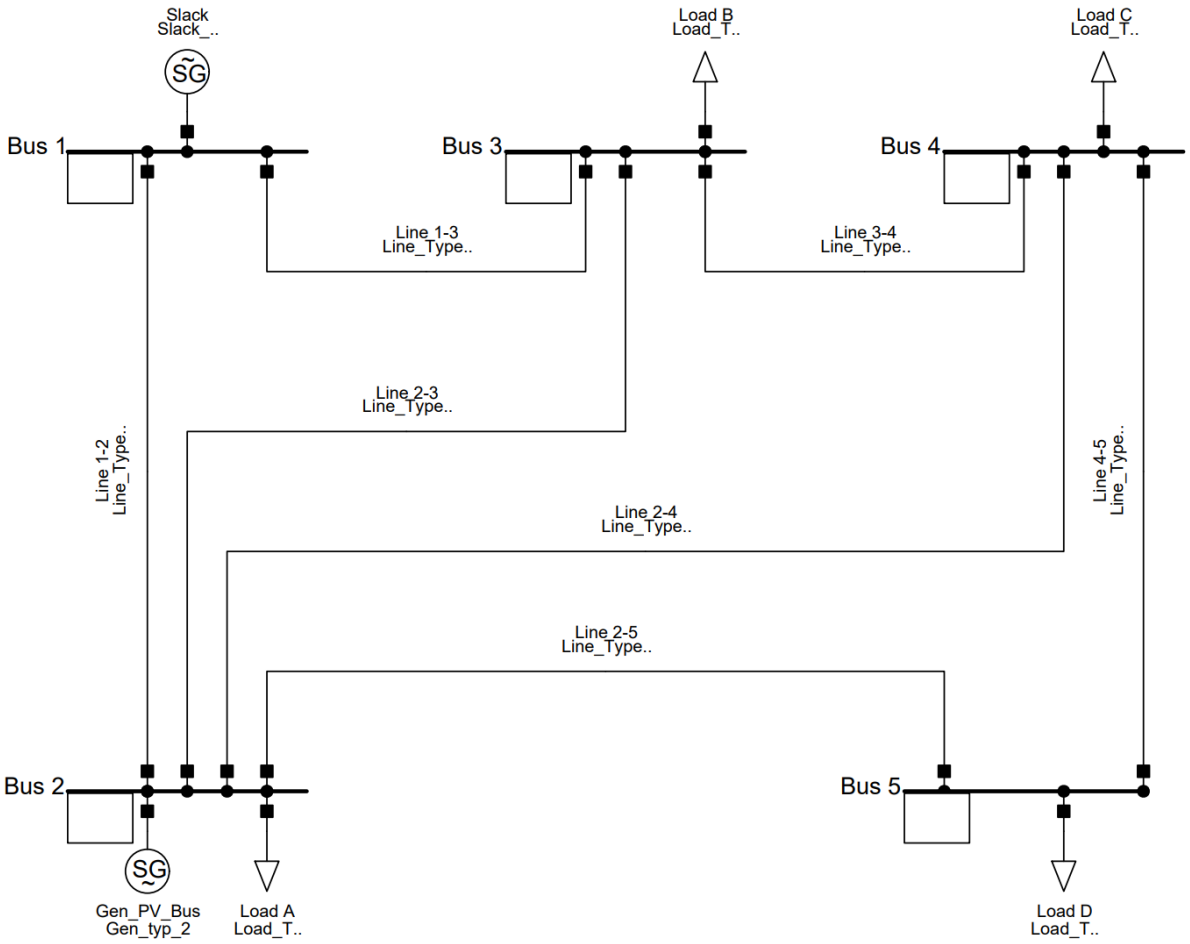


Fig. 4.1: IEEE 5 Bus Grid According to [2] in PF

Based on [2] the base of $S_{\text{Base}} = 100\text{MVA}$ and $V_{\text{Base}} = 230\text{kV}$ were chosen. In section G.1, the conversion of p.u. values from Tab. 4.1 into values with units is reproduced. Derived from this informations, the Y bus matrix $\underline{\boldsymbol{Y}}_{\text{M}}$ for this grid can be determined as described in section 3.1. The matrix can be found in section G.3 for comparison purposes, in order to check a correct implementation.

According to [2], the grid has the following input parameters from Tab. 4.2. The table also contains the bus type, which is relevant for the load flow calculation.

Tab. 4.1: Data of Lines in the IEEE 5 Bus Grid Based on 100 MVA [2]

from	**to**	r **in [p.u.]**	x **in [p.u.]**	$b/2$ **in [p.u.]**
1	2	0.02	0.06	0.06
1	3	0.08	0.24	0.05
2	3	0.06	0.18	0.04
2	4	0.06	0.18	0.04
2	5	0.04	0.12	0.03
3	4	0.01	0.03	0.02
4	5	0.08	0.24	0.05

Tab. 4.2: Buses in the IEEE 5 Bus Grid Based on 100 MVA [2]

bus	**typ**	V **in [p.u.]**	V **in [°]**	$\boldsymbol{P_G}$ **in [MW]**	$\boldsymbol{Q_G}$ **in [MW]**	$\boldsymbol{P_L}$ **in [MW]**	$\boldsymbol{Q_L}$ **in [MW]**
1	slack	1.06	0	0	0	0	0
2	PV	1.00	0	40	0	20	10
3	PQ	1.00	0	0	0	45	15
4	PQ	1.00	0	0	0	40	5
5	PQ	1.00	0	0	0	60	10

Since the IEEE 5 bus grid under consideration is recommended for $V_{\text{Base}} = 230kV$, an HV grid can be assumed. For the application in the LV grid, it is recommended to use an adapted model, as there is actually no PV bus in the LV grid.

For simplification, the grid was adapted compared to the IEEE5 bus grid in that the load on bus two is considered without reactive and active power load. The reason for this is that this only gives additional feature vectors, which either added together ($P_L + P_G; Q_L + Q_G$) with loss of information or separately ($P_L; P_G; Q_L; Q_G$) with a number of values not assigned causes an increase of the feature vector.

As input, the description of the real (active) and imaginary parts (reactive) of the power is used instead of the angle and the magnitude of the power. The reason for this is that the angle would otherwise be very dependent on the magnitude of the power and should be avoided in relation to a possible feature value that is independent of each other (see section 4.4).

4.2 Generation of Cases

Different values for P_L, P_G, Q_L and Q_G have been chosen in comparison to Tab. 4.2. This aims to generate random syntactical values for these quantities with which the ANN can be trained in chapter 5.

The decision to implement this approach using random functions allows scalability of the training data. If the variants were applied systematically, a significant number of variants would have to be calculated, resulting in longer computation times. Therefore, the static voltage stability criterion cannot be determined using the P-V curve or Q-V curve method, as is done in [6].

In this method, the active power P or the reactive power Q of a bus would be increased until the load flow calculation no longer converges. This signifies static voltage instability. However, when trimming the data, bias or non-learning of a certain range may occur.

As a result, this course of action must be repeated for every bus while varying the other active power P or the reactive power Q values at the different buses. In addition, it should be noted that the majority of ANN training cases are characterized as static voltage stable, as non-convergence is usually the termination criterion for this method.

The value for the load has been randomly generated for bus 3 to bus 5 for power P_{L} and reactive power Q_{L} from 0 MW to 99 MW using the passive sign convention. The order of magnitude of the power values is based on the active power values selected in section 4.1.

Since there is no observation over time to determine the static voltage stability, there are no load profiles that are assumed accordingly. Reducing the state space of the feature by adjusting the order of the power magnitude results in the ANN being able to interpolate more accurately in the reduced space, assuming ideal training, but not beyond the expected range. Since the AI of the ANN should go beyond known conditions, a buffer was added to the maximum value from Tab. 4.2. For the variation of the active power of the generator in PV bus 2, values from 0 MW to 199 MW were generated. Since the slack is calculated, as well as the reactive power Q at the PV bus, these data do not have to be given.

For all synthetically generated values, rounding to MW was chosen. This reduces the accuracy with which the boundary between unstable and stable can be determined. Since this is not the requirement, but only a safe range is to be determined, rounding was used for the purpose of improved classification.

The slack bus represents the higher-level grid and can be considered as a stable grid, i.e. the frequency stability is given and no deviation of the initialised voltage is assumed.

The corresponding program was implemented in Python 3.9 and is available via GitHub. More information can be found in section G.5.

4.3 Calculation of the Cases

The Gauss-Seidel method described in section 3.2.1 and the Netwon-Raphson method described in section 3.2.2 was used to calculate the load flow with the input data. The implementation is based on a program for both methods with three individual phases in MatLab by Prof. Dzienis. It was adapted for Python and the PV bus calculation was added. In addition, DIgSILENT's PowerFactory (PF) 2023 was used via its Python API.

A parent programme (main) has been created in the spirit of universal implementation. This calls up the child method used, which is stored in another program. This was necessary in order to easily provide the programs with comparative calculation values (e.g. for Fig. 4.4), since the method was used in several main programs. Once the method had been adapted, the changes were automatically made in all the main programs in which the method was used. On the other hand, multiprocessing was implemented for the Gauss-Seidel and Newton-Raphson methods. Since Python itself is not capable of multiprocessing, "multiprocessing" had to be implemented via the Python library. This requires outsourcing of functions anyway, so outsourcing the method to a separate program seemed appropriate. With regard to multiprocessing, the number of CPUs (Central Processing Unit) to be used can be specified.

The main programme, f.e, was used to generate cases and store the data. Data storage differed for each method in order to record separate results. The data were stored in the 'parquet' file format. This allows structured data to be stored in a compressed, efficient form. Unlike the typical CSV format, the parquet file format stores data in columns. It also allows columns to be added and can be used in Python with the Pandas library. As complex values are disadvantaged for further use, they are split into their real and imaginary parts. The program also checks that the file in which the data is to be stored exists and creates it if necessary. It also shows the percentage of states that it considers unstable, the number of lines and stops saving when it exceeds a certain size.

Furthermore, the Y bus matrix was also defined in this main programme (see section G.3), as well as the start voltages, bus types and the number of cases that are carried out and added to the dataset.

The main file also used an executer file, as memory is heavily used if data is not saved regularly. This meant that the program was called repeatedly in the background and a large dataset could be created. Another reason for this is the temporary cache problem, as discussed in section 4.3.3.

The corresponding program was implemented in Python 3.9 and is available via GitHub. More information can be found in section G.5.

4.3.1 Calculation with Gauss-Seidel

Gauss-Seidel methods are used for determination according to section 3.2.1. Since the programme created has been programmed for general usage, adjustments have to be made. In the first step, the location of the slack bus is determined and associated values are not required in the calculation. Likewise, if a PV bus is identified, the active power Q is not considered. The necessary calculation is carried out at the location of the PV bus (see Eq. 3.10), which determines an active power Q of the corresponding PV bus. The calculation according to Eq. 3.9 follows. In this way, all voltage values and power values can be determined.

Fig. 4.2 shows the development of an exemplary calculation with Gauss-Seidel and Newton–Raphson for 25 iterations. V_1 describes the slack and V_2 the PV bus, which is why the magnitude of the voltage is constant. In order to speed up the calculation, it was implemented that the program stops early if the last three values change below $\Delta V < 0.0005$ kV. The number of iterations is limited to 200 iteration steps as a second stop criterion.

If there is no convergence within this limit, then the case is unstable and there will be an oscillating voltage. If it converges during this limit, a further analysis is carried out according to section 3.3. Since no Jacobian matrix is created for the Gauss-Seidel method, which is necessary for the determination of the static voltage stability according to the criterion, the matrix is created as described in section 3.2.2 and evaluated accordingly.

4.3.2 Calculation with Newton–Raphson

The calculation is done as described in section 3.2.2. Similar to section 4.3.1, columns are removed at the beginning. These are values related to the slack bus, but this time also row and column in the Jacobian matrix. The Jacobian matrix was then edited in relation to the PV bus and the corresponding reactive power Q was removed as this value is not provided.

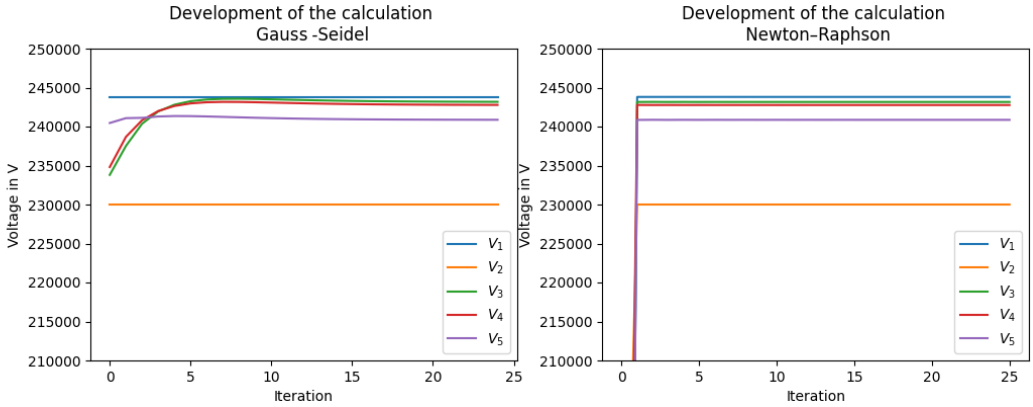


Fig. 4.2: Comparison of the Development of the Calculation with the Gauss-Seidel and the Newton-Raphson Method, for the Comparison Without Convergence Criterion for the Case from Tab. 4.2

Fig. 4.2 shows the development of an exemplary calculation with Gauss-Seidel and Newton–Raphson methods for 25 iterations. It is visible that the calculation with Newton–Raphson requires significantly less iteration steps (3) compared to Gauss-Seidel (20). The same early stop condition was used as in the previously described Gauss-Seidel implementation. In contrast to Gauss-Seidel, the number of iterations is limited to 100 iteration steps, since a faster convergence can be achieved with this method.

The corresponding program was implemented in Python 3.9 and is available via GitHub. More information can be found in section G.5.

4.3.3 Setting Up the Calculation with PowerFactory

The implementation of PowerFactory 2023 from DIgSILENT GmbH proved to be the most time-consuming. It is based on two steps, the necessary setup in PowerFactory and the implementation of the API in Python.

To construct the grid, a graphical interface must first be established. This involves placing the elements like buses, generators and loads in the graphical interface. Next, all lines need to be created. Each created element must be assigned a specific type. For instance, the converted values from Tab. G.1 must be entered for the lines. Additionally, the slack bus must be activated within the generator element. The PV bus's generator has been assigned a V-controller. The basic settings of PowerFactory remain unchanged. The voltage in p.u. from Tab. 4.2 was added at each terminal, along with the nominal voltage of the relevant type. This process is prone to error and requires careful checking to ensure that all values are entered correctly. Finally, the following overview is obtained (see Fig. 4.3).

The calculation of the synthetic cases is done via a Python API, which differs between Python versions and must be entered at the beginning. The project must then be activated by its name, while ensuring that PowerFactory is closed to avoid any error messages. Finally, *.ElmSym* can be utilised with the *GetCalcRelevantObjects* command to determine the location of the generator. Values can be set in PowerFactory using the command "SetAttribute" utilizing the obtained data and assigned names. This command enables the variation of synthetically generated cases as depicted in Tab. 4.3. The state of all other values remains unchanged from the previous alteration in PowerFactory.

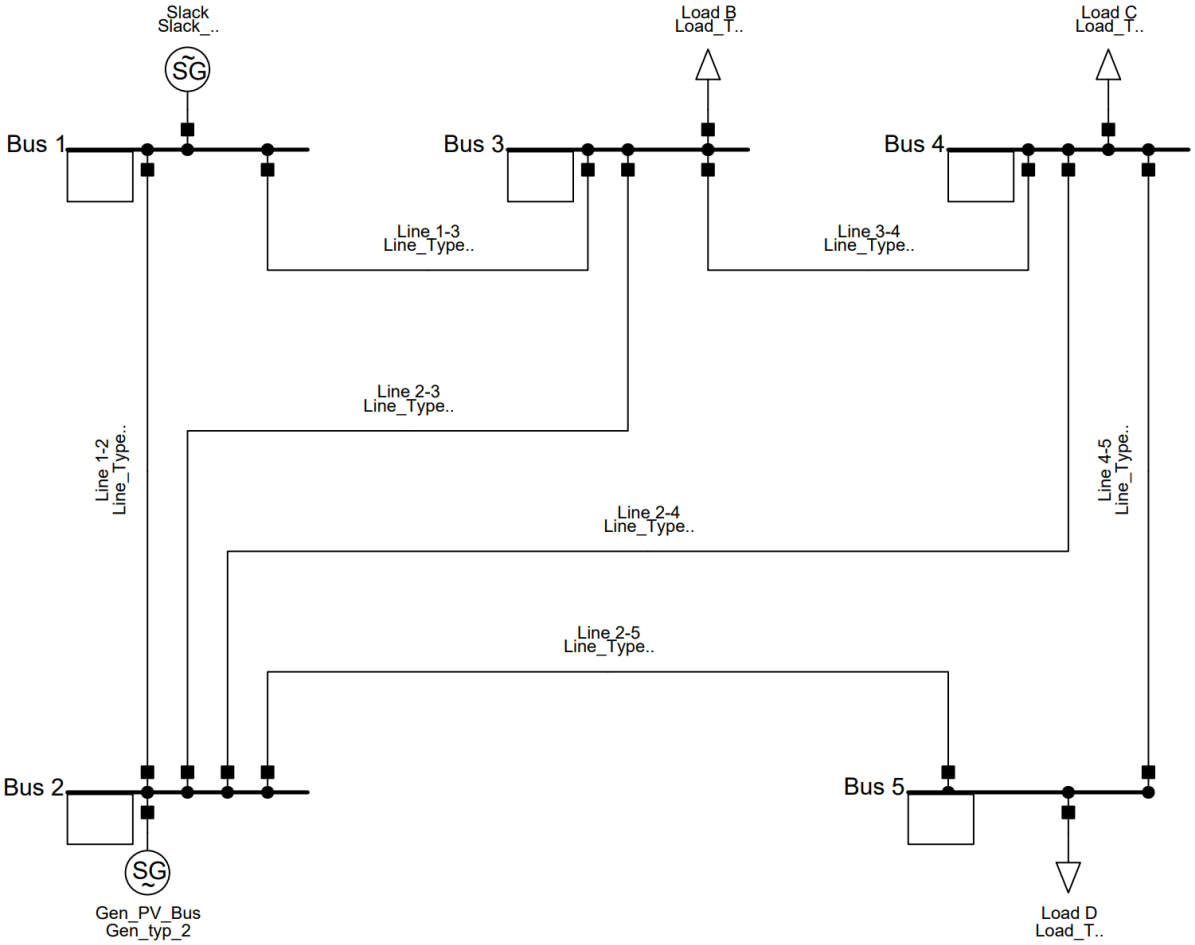


Fig. 4.3: Modified IEEE 5 Bus Grid in PF

Tab. 4.3: Changing Values in PowerFactory

name	**value**	**unit**	**PF value name**	**site**	**PF site name**
active power	P	MW	pgini	generator	$.ElmSym$
reactive power	Q	MVar	qgini	generator	$.ElmSym$
voltage	V	in p.u.	usetp	generator	$.ElmSym$
active load	P	MW	plini	load	$.ElmLod$
reactive load	Q	MVar	qlini	load	$.ElmLod$
voltage	V	in p.u.	u0	load	$.ElmLod$

Subsequently, the load flow calculation *ComLdf* can be initialised via the command *GetFromStudyCase* and executed with *Execute*. If the output value is zero, the calculation was successful, so there is a converged result. In the next step, all values of the load flow calculation are obtained via the command *GetAttribute* (see Tab. 4.4).

As the Jacobian matrix cannot be obtained in PowerFactory, this must be done via the sensitivity analysis.

Tab. 4.4: Get Values in PowerFactory from Load-Flow Calculation

name	**value**	**unit**	**PF value name**	**site**	**PF site name**
active power	P	MW	m:P:bus1	generator	$.ElmSym$
reactive power	Q	MVar	m:Q:bus1	generator	$.ElmSym$
name	Q	MVar	loc_name	Bus	$.ElmTerm$
voltage amplitude	V	in p.u.	m:u1	bus	$.ElmTerm$
nominal voltage	V	in kV	GetUnorm()	bus	$.ElmTerm$
voltage angle	δ	rad	m:phiu	bus	$.ElmTerm$

Similar to the load flow calculation, this must be accessed in an additional step using the command *ComVstab*. Then the values for each bus i must be determined in dependence on the other buses j to create a matrix. This can be done using two loops, with the calculation being performed each time after the first loop, since the values can only be read for each bus ij. The sensitivity can be called with *m:dvdQ* of the corresponding bus. In addition, the first column and row have been removed as they correspond to the slack bus where there is no voltage change (ΔV).

For the selection of the criterion, however, the dV/dQ sensitivity must first be activated via the selection variables[1] and all thresholds needed to be switched off. Using the criterion of the sensitivity analysis for static voltage stability criterion, all diagonal values of the dV/dQ sensitivity matrix created were checked to be greater than or equal to zero.

Multiprocessing is not supported by the PowerFactory API, but users can modify the number of processors in PowerFactory in the parallelization settings[2]. Nevertheless, there is no disparity in computation time when calling through the API using either one or twenty cores. However, the calculation period is impacted by the clock speed and type of the CPU.

It was observed that PowerFactory's performance deteriorated as the number of calculations increased. The optimal step duration of 36 ms per 1000 calculations rises to 2.8 s per case before aborting, which is unacceptable calculation time. Initially, loading a previous backup resolved the issue and restored the previous calculation speed.

After examining the case, an option was found not to cache data[3]. The issue was resolved by utilising the executer program, which clears the cache at every program start.

The corresponding program was implemented in Python 3.9 and is available via GitHub. More information can be found in section G.5. A tool has been developed for PowerFactory to read the Y-matrix of grids that do not contain transformers and do not contain disconnected lines.

[1] Calculation -> Sensitivities -> Results -> Variable selection -> Terminals -> Sensitises -> m:dvdQ to selected variables

[2] Navigate to user settings > Parallelisation > Enable all processors

[3] Project manager -> select project -> right click -> settings -> memory -> enable automatic cleanup -> 0d

4.4 Comparison and Analysis of the Data

A powerful workstation was used to calculate the data [4]. This results in the following calculation times per method (see figure 4.4).

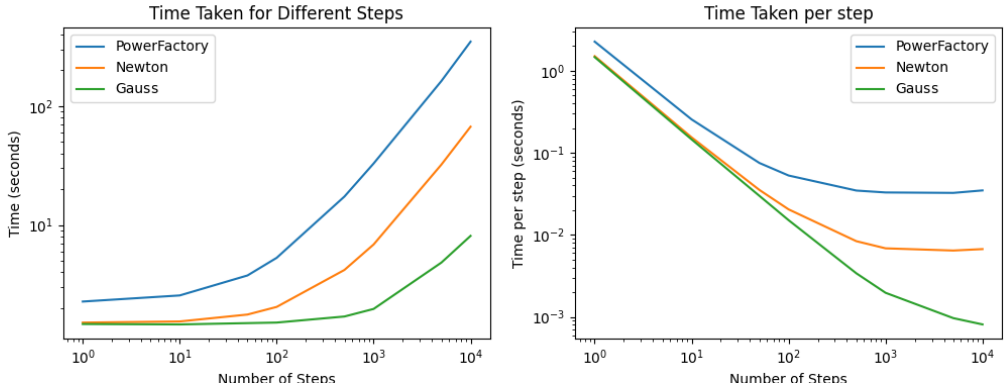


Fig. 4.4: Computing Time of the Models

It turns out that the Gauss-Seidel method has by far the best calculation time. This is due to the simple algorithm and was already described by [26]. This is significantly influenced by the choice of the limit of iteration steps or stop condition.

The Newton-Raphson method improves computational time by implementing multiprocessing. In contrast, PowerFactory experiences considerable computation time, partly attributed to the API's inability to support multiprocessing in this format, which requires the loading of a large program in the background. For this reason, PowerFactory is slower in one step where the number of processors used should have no influence.

The advantage of PowerFactory over the two calculation programs is that it has been validated in a wide range of publications (f.e. [11]) and is therefore reliable. In the implementations using the Gauss-Seidel and Newton-Raphson methods, there is a small deviation in the voltage (see Fig. 4.5 to Fig. 4.7). In the diagrams, the investigated methods for load flow calculation were examined with regard to the deviation from each other in the absolute value, in the imaginary part and in the real part of the voltage. For the voltage, the unit *per unit* (p.u.) was chosen, so that the realizations remain absolute, but the value remains classifiable. When comparing the results of PowerFactory with the own results of the implementation, a maximum deviation of approx. 0.0075 p.u. was found.

The error is small for the imaginary proportions and the absolute error is dominated by the real proportion. Between the two own implementations, there is a very small deviation of approx. max. $5 \cdot 10^{-9}$ p.u., although these work after different methods. So PowerFactory seems to use different calculations or slightly different input data here, which could be provided in the default settings e.g. for the generator. This allows the own implementations to keep up and get better computation times, eventually by choosing a different stop criterion.

The implementation of the voltage stability criterion (refer Eq. 3.35) is problematic. Both the Gauss-Seidel and Newton-Raphson methods identified only 9.6% of 1000 test calculations as stable, while PowerFactory identified stable cases in 42.8% of 1000 test calculations.

[4] 13th Gen Intel(R) Core(TM) i5-13500 2.50 GHz (number of threads 20, number of cores 14),DDR4 64 GB RAM, GeForce GTX 1060 6 GB

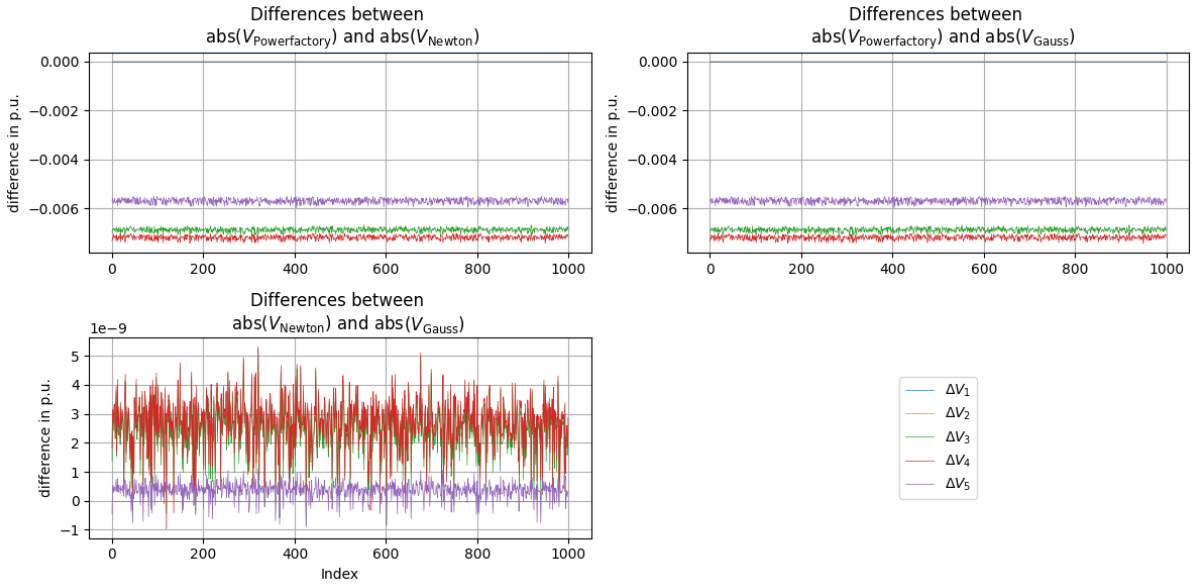


Fig. 4.5: Difference Between the Models Results for abs(V) for Load Flow Calculation

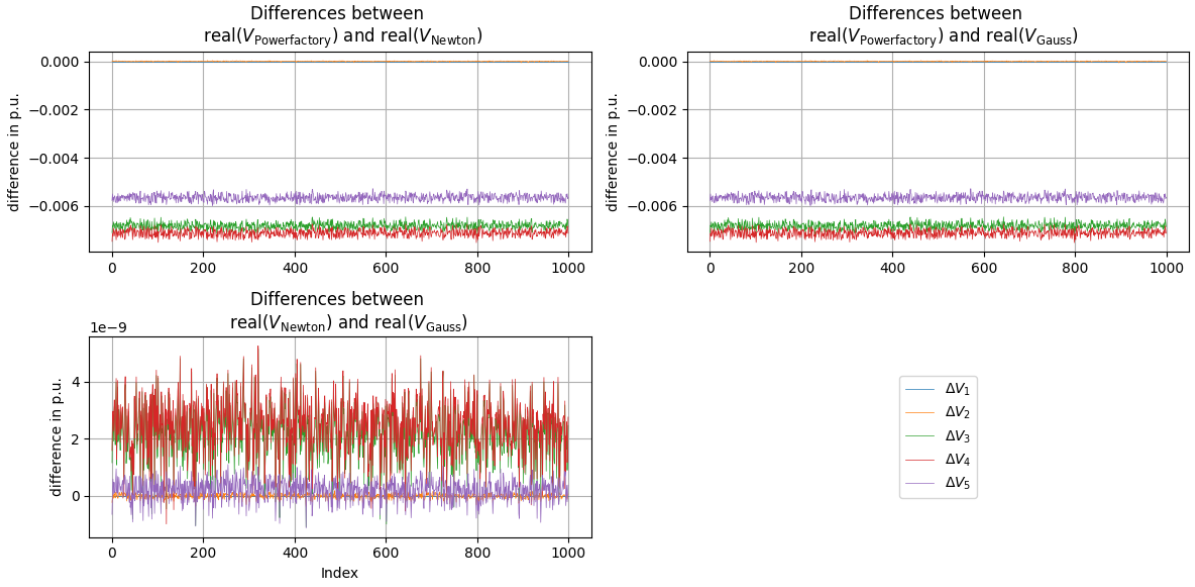


Fig. 4.6: Difference Between the Models Results for real(V) for Load Flow Calculation

Assuming that PowerFactory is also the correct value, there were 46 false positives for both methods. It is presumed that either the implementation of dV/dQ is incorrect or PowerFactory is calculating differently.

The PV bus has no effect on this, as in another tested 5 bus grid according to [26] without a PV bus, no match could be found using this criterion. The basic Jacobian matrices can be assumed to be correct, since the load flow calculations between PowerFactory and the implementations produce almost identical results. PowerFactory and the implementations also use the passive sign convention. There may also be another stopping criterion that causes different Jacobian matrices to be generated. To clarify how the dV/dQ sensitivity was imported in PowerFactory, it might be necessary to consult the developer DIgSILENT.

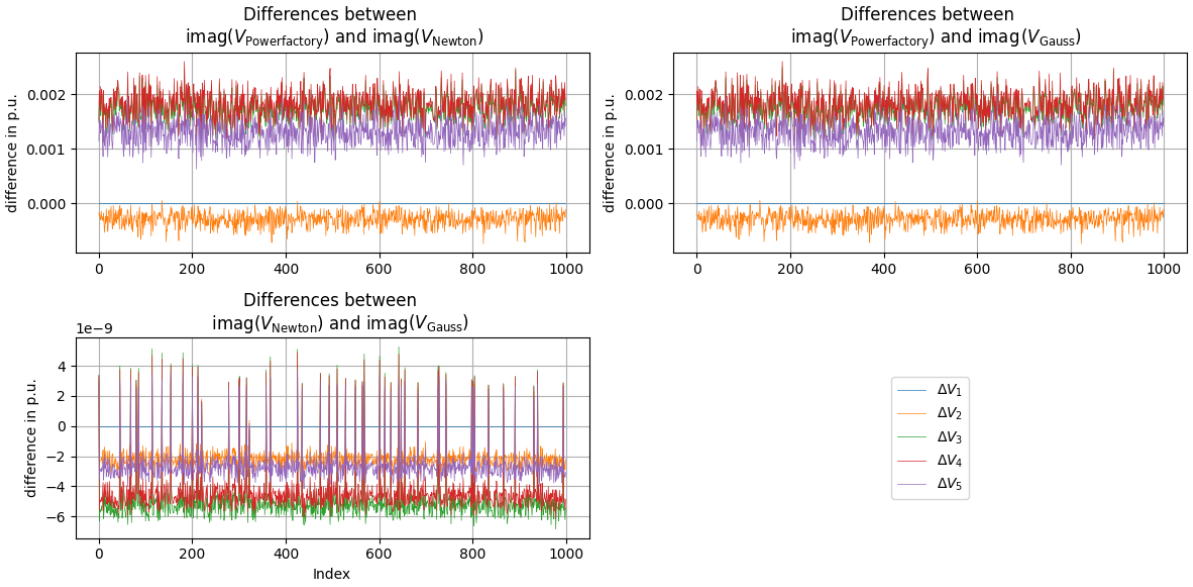


Fig. 4.7: Difference Between the Models Results for imag(V) for Load Flow Calculation

Due to the unclear situation in the correct application of the stability criterion of static voltage stability, the creation of the data was realized with PowerFactory, using the criterion of the sensitivity analysis [25] with dV/dQ sensitivity from PowerFactory.

Based on this calculation, a dataset with 789000 cases was created. To understand the size of the dataset, it requires about 8 hours of computing time and is 108.96 MB in size. This dataset contains the active and apparent power of the load and the generators, the voltage in the real and imaginary part, the diagonal matrix of the dV/dQ sensitivity as well as the evaluation according to the criterion. The percentage of feature vectors that do not meet the criterion is 42.0 % and those that do meet it is 58.0 %. The large dataset was created because it was not clear at the time of creation how large the dataset would need to be for the ANN training.

An analysis of the correlation results in Fig. 4.8 between the features and Fig. 4.9 between the features and the target.

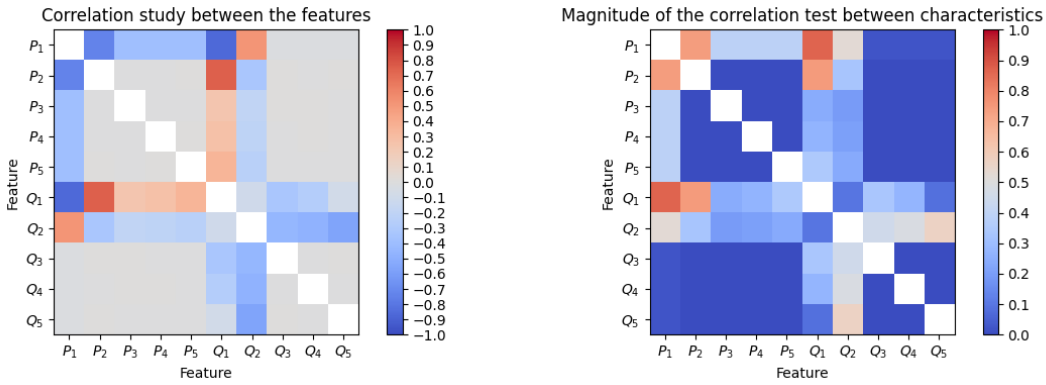


Fig. 4.8: Correlation of the Feature Vector from the Entire Dataset

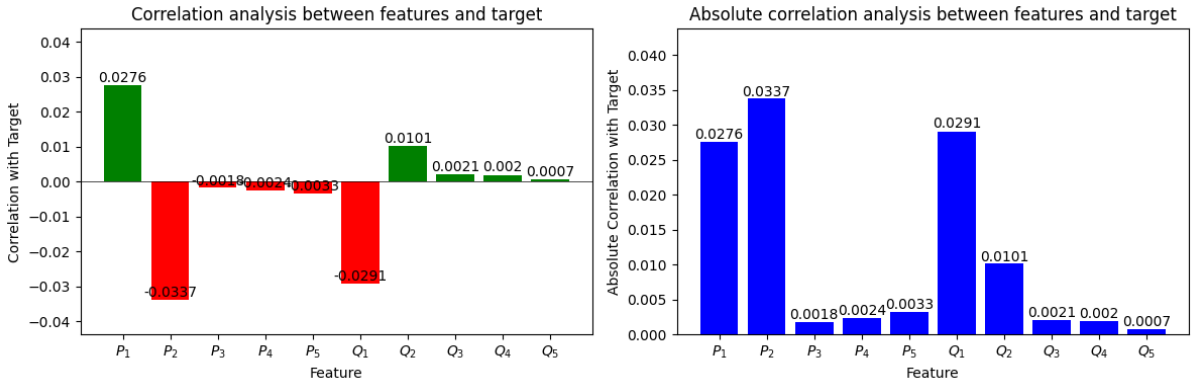


Fig. 4.9: Correlation of the Target and the Feature from the Entire Dataset

Correlation analysis can be used to evaluate the dataset before training with an MLP. Correlations indicate how linear the features are to each other and to the target. If there is a high correlation between the features, this is called multicollinearity. This can lead to unreliable estimates in static models such as MLPs. In this case, it can be difficult to isolate the influence of each feature and train the model. This can lead to poor training results as it is difficult for the ANN to find relative relationships between the features and the target.

Low correlation, on the other hand, means that the features are complementary and contain different information that the ANN can use for good training.

From Fig. 4.8 it can be seen that the correlation between reactive power Q_1 and active power P_1 is high. There is also a correlation between P_1, P_2 and Q_1 which can be related to the PV bus. A small correlation between Q_2 and Q_5 has the same effect. These values represent operating parameters based on the calculation and therefore represent a correlation. In general, it can be assumed that the datasets are less correlated with each other.

In addition, the correlation between the features and the target can provide further insight. A strong correlation with the target indicates that the features are relevant for predicting the target. In this scenario, the MLP can make better use of the relevant features and make more accurate predictions. Conversely, if there is no correlation, the number of features can be reduced. The correlation between feature and target is low, see Fig. 4.9. The correlation between the reactive power Q_3 to Q_5 and the active power P_3 to P_5 is low. This can be explained by the fact that these values were generated randomly. The exception is PV bus 2, for which P_2 was given and the correlation is high, while for the calculated Q_2 the correlation is low. In comparison, the values of the slack bus show a high correlation. In this respect, it can be assumed that the dataset has a low correlation between the target and the features.

The stored voltage values are used to assess whether the magnitude of the voltage meets the requirements of 0.9 to 1.1 V_n (= 230000 kV) from Eq. 3.29 (voltage contingencies).

As these states lead to a generator/load adjustment in the grid and this adjustment leads to a valid voltage state range, it can be assumed that these values can be filtered for another observation state. In the following, this dataset is always referred to as the dataset without voltage contingency. The number of values is reduced to feature vectors with a length of 203639, of which 57.1 % (116269) meet the static voltage stability criterion and 42.9 % (87370) do not. This is similar to the previous ratio of 42 % for meeting the criterion and 42.9 % for not meeting the criterion.

This results in the following analysis of the correlation results in Fig. 4.10 between the features and Fig. 4.11 between the features and the target.

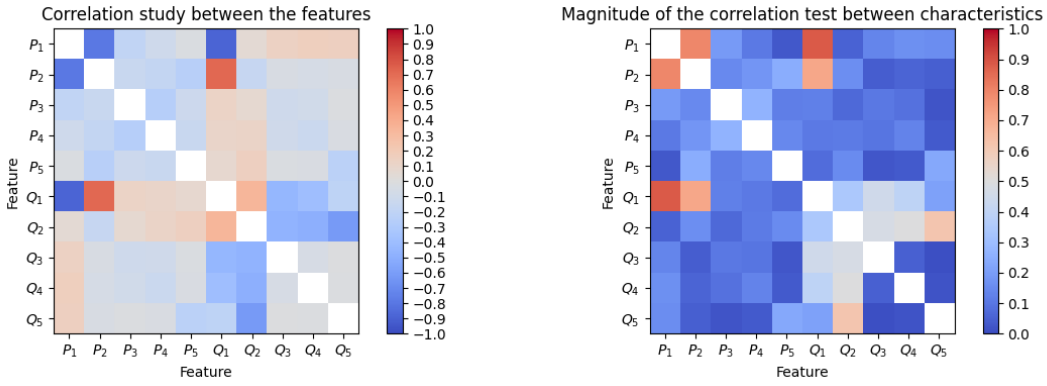


Fig. 4.10: Correlation of the Feature for Dataset Without Voltage Contingency

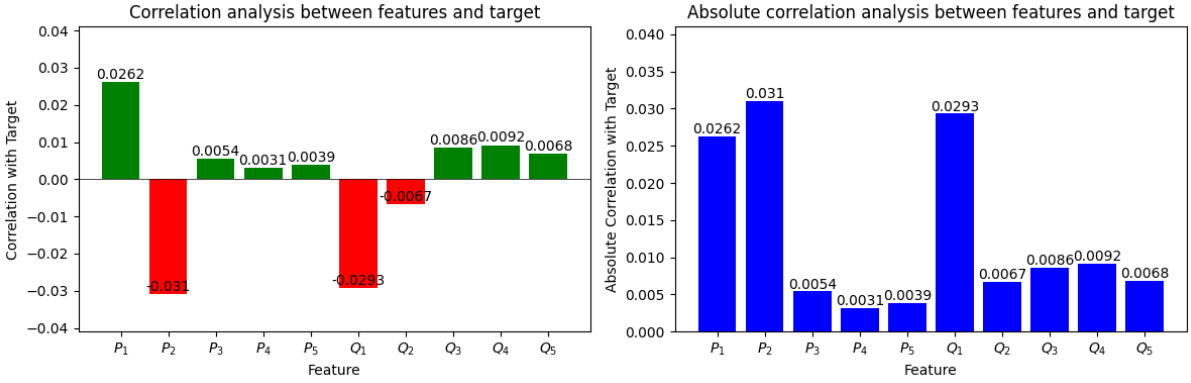


Fig. 4.11: Correlation of the Target and the Feature for Dataset Without Voltage Contingency

Through this processing, the correlation changes slightly (see Fig. 4.10), but the sign of the correlation between the target and the features P_3 to P_5 and Q_2 changes (see Fig. 4.11). However, since these values are very low anyway, they cannot be based on a central correlation. They remain lower than the correlations among the features.

To investigate the effect of the ratio of positive to negative according to the voltage stability criterion on the correlation, the proportion of those classified as positive according to this criterion was randomly reduced to 87370 feature vectors. In this way, the same ratio is obtained. The correlation changes, as shown in Fig. 4.12 and Fig. 4.13. There are no other correlations resulting from this and the correlations remain identical except for minor changes.

In the following, reduced datasets were created with which the training of the ANN was carried out. The analysis of the datasets can be found in the appendix in section G.6. It was ensured that the ratio between the two target classes is equal. The datasets created are based on the approaches described in section 5.1, corresponding to the datasets described here.

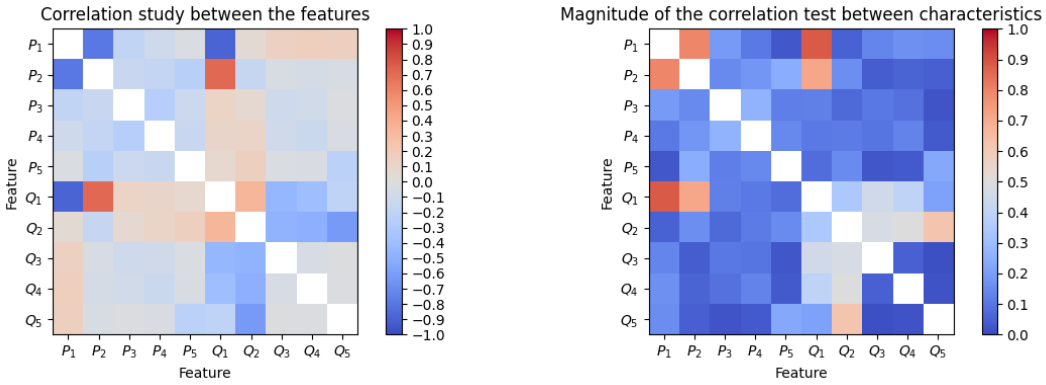


Fig. 4.12: Correlation of the Feature for Dataset Without Voltage Contingency with Same Ratio/ False and True Criteria

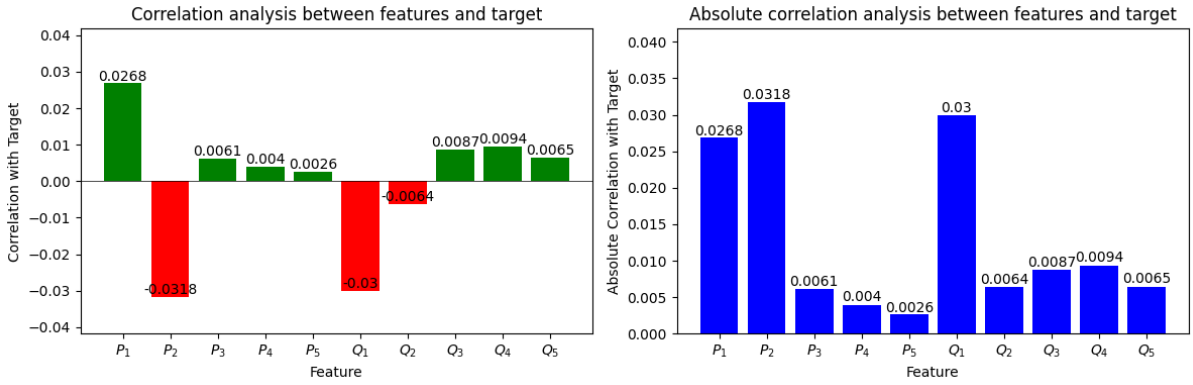


Fig. 4.13: Correlation of the Target and the Feature for Dataset without Voltage Contingency with Same Ratio/ False and True Criteria

Summary of the Chapter

The load flow calculation based on synthetically generated cases was performed using a 5 bus grid using PowerFactory via its Python API. In addition to the calculation with PowerFactory, Gauss-Seidel and Newton Raphson load flow methods were considered. The dataset was then evaluated using PowerFactory's static voltage stability criterion. An implementation of this criterion in the other method did not produce consistent results. The result was a large dataset required for training, validation and testing of the ANN. The dataset was adjusted for voltage contingency with voltages outside a required range to investigate the impact on ANN training.

5 Design and Implementation of a Artificial Neural Network

5.1 Approaches

The chosen approach is the creation of AI on the basis of an artificial neural network (ANN) which predicts the evaluation of the stability criterion according to [25] with the input of the active and reactive power at the buses of the modified IEEE5 bus grid.

The following variants are approaches to be realised via the ANN:

1. Training with the dataset with voltage contingency according to Eq. 3.29 and data of PowerFactory in a multi-layer-percepton (MLP) (10 feature vector) called **PowerVcon10F**.
2. Training with the dataset with voltage contingency according to Eq. 3.29 and data of PowerFactory in a MLP with reduced input (7 feature vector) **PowerVcon7F**.
3. Training with the dataset without voltage contingency according to Eq. 3.29 and data from PowerFactory in a MLP (10 feature vector) **Power10F**.
4. Training with the dataset without voltage contingency according to Eq. 3.29 and data from PowerFactory in a MLP with reduced input (7 feature vector) **Power7F**.
5. Training in a Kohonen network.

Keras, a Python library designed for AI, was used to develop the AI. The search for a suitable ANN architecture for hyperparameter optimisation using the Python library Tuner in Keras was unsuccessful, so the design and implementation were done using the pruning function of DataEngine from MIT GmbH. Successful MLP designs from DataEngine were then transferred to Keras. The pruning function significantly reduces many degrees of freedom, like the number of hidden layers, number of connections and activation functions.

For Keras, the GPU was used instead of the CPU via Nvidia CuDa (interface for GPU use, Compute Unified Device Architecture) and CuDNN (Deep Neural Network). The computer already used for data generation was also used and is equipped with a Nividea graphics card (GeForce GTX 1060). The training is limited by the 6 GB of graphics card memory.

In addition, based on the dataset with voltage contingency, an ANN was tested for AI-based load flow calculation in section G.9, called **PQtoV**. The target is better suited for MLP creation with DataEngine due to the approximation of values.

5.2 Multi Layer Perceptron Design

DataEngine was used in the design of the MLP as it provides a very useful pruning function. Pruning requires the number of connections n_c to be greater than the number of training patterns n_t (see Eq. 5.1). This is called "overfitting", which leads to an over-dimensioning of the architecture.

$$n_c > n_t \tag{5.1}$$

The number of training data/patterns is therefore also relevant for the design of the MLP. For a dataset of, f.e., 1000 training patterns, the following condition must be met for a two-layer dataset according to Eq. 5.2. The Index A describes the number of input neurons. Index B is the number of output neurons and M_x is the number of neurons in each hidden layer. A generalised function is given in Eq. 5.3.

$$n_c = A \cdot M_1 + M_1 \cdot M_2 + M_2 \cdot B \tag{5.2}$$

$$n_c = A \cdot M_1 + B \cdot M_n + \sum_{i=1}^{n} M_i \cdot M_{i+1} \tag{5.3}$$

Another equation for the design of the MLP is to estimate the number of neurons in the hidden layers M_n according to Eq. 5.4) from [13].

$$M_n = 2/3 \cdot A + B \tag{5.4}$$

This results in a number of neurons in the hidden layer that should correspond to the results of the pruning process. From $A = 10$ (for 10 features) and $B = 1$ (one target), $M_n = 8$. In order to allow for possible adaptation, a higher number of neurones can be chosen $M_x = 10$. This means that the design of the MLP with one layer must have a dataset of less than 110 training patterns. For DataEngine, only a maximum of two layers is possible. Another hidden layer only increases the number of training patterns by $M_x^2 \cdot x$, i.e. 100 training patterns to 210.

Pruning removes connections with low activity over a period of time (default time constant setting in DataEngine is 10 epochs), using a release threshold (default is 0.02) in a distinct training process. The successful result is a reduced number of connections, which should be less than the number of training patterns n_t. If the size is reduced, a learning effect can be ensured, whereas if the size remains higher than the training patterns n_t, the target may be memorised depending on the input.

To accommodate the pruning result, the architecture of the ANN should be adjusted to have at least the same number of connections. In addition, the selection of neurons in MLPs should decreases towards the output. Pruning was performed according to the settings listed in section G.7.

1. PowerVcon10F:
A dataset of 100 patterns/ cases was used for the pruning process. This is described in section G.6. An examination was carried out for one and two hidden layers with a neuron number M_x= 10 per layer. For the pruning process in DataEngine, the settings from section G.7 were used with an oversizing of the ANN, with 110 connections with one hidden layer or 210 connections with two hidden layers. The activation functions were changed in order to test several variants. The results are summarized below in the Tab. 5.1.

Tab. 5.1: Examination of the Architecture for PowerVcon10F (Pruning)

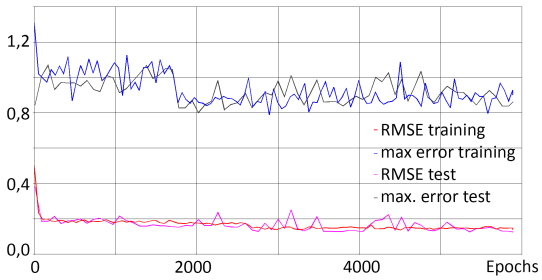
variant	M_1	$f(x)$ **from** M_1	M_2	$f(x)$ **from** M_2	n_c	n_c **after pruning**	**RMSE**	ε_{max}
1	10	sigmoid			110	19	0.3587	**0.6529**
2	10	tanh			110	8	**0.2960**	0.9994
3	10	sigmoid	10	sigmoid	210	32	0.2760	**0.8479**
4	10	tanh	10	sigmoid	210	47	**0.1251**	0.8621
5	10	sigmoid	10	tanh	210	26	0.2360	0.9852
6	10	tanh	10	tanh	210	31	0.1352	1.0020

The goal is to have the smallest possible root-mean-square error (RMSE) and secondly the smallest possible maximum error. The RMSE should differ from the target distribution (50% stable/ 50% unstable), since this ratio represents the limit precisely, indicating that no learning has occurred. The RMSE is calculated according to Eq. 5.5, while the maximum error ε_{max} is the maximum difference between target t_i and prediction o_i (see Eq. 5.6).

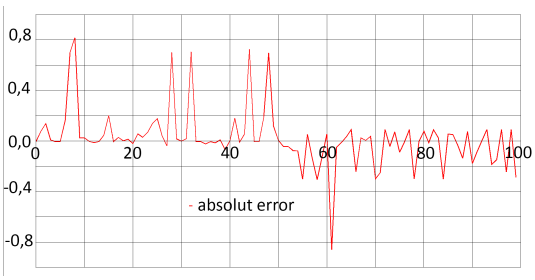
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_i - o_i)^2} \tag{5.5}$$

$$\varepsilon_{\max} = max_i|t_i - o_i| \tag{5.6}$$

Other measures of training quality are outlined in section G.8.4. This depicts the learning curve and error rates for test values. These diagrams for variant C-2 are further explained in the main section below.



(a) Learning Curve



(b) Error of the Train Data

Fig. 5.1: PowerVcon10F Variant 4 (10 linear -10 tanh - 10 sig - 1 linear)

Fig. 5.1a shows the learning curve, which includes the RMSE and the maximum error.

During the pruning process, the test dataset is assigned to the training dataset. It is evident that the root mean square error decreases and then converges. Although the maximum error remains high, further examination with the test of the ANN after pruning is necessary.

This is illustrated in Fig. 5.1b, which reveals that only 7 values have deviations greater than ± 0.5. As DataEngine does not have a threshold function to choose from, this should be considered for implementation in Python with Keras. This would then allow for threshold based classification with a value of 0.5, resulting in the precise assignment of the values. With this approach, the test error rate would be around 7%. If the maximum error during file input is less than 0.5, it indicates that all values were assigned correctly.

Since the same dataset was used, it can be checked whether similar maximum errors exist. As this is not the case, the outliers cannot be attributed to strong deviations from certain values. The aspect of the physically explainable outliers is examined more closely based on the dataset without violating the voltage contingencies.

This leads to the further investigation of the variants, the results of which are shown in bold in the table, with an adapted architecture with a reduced number of connections.

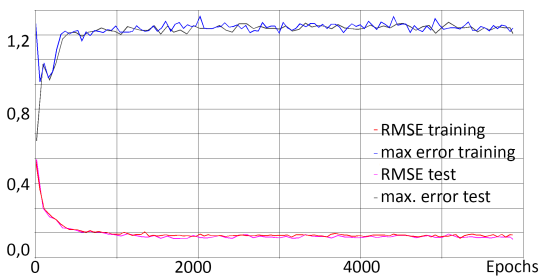
Tab. 5.2 presents the additional variants that were examined. If the learning curves show strong fluctuations, it is recommended to reduce the learning rate. This effect is apparent in variant c-2, which is presented here with a reduced learning rate of 0.01 as opposed to 0.1 in variant c (see Fig. G.13a). The learning rate for variant d-2 was therefore also set to 0.01 for the output layer. The calculated error ε_{cal} (see Eq. 5.7) takes into account potential errors associated with the threshold of 0.5.

$$\varepsilon_{\text{cal}} = \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 1, & \text{if } t_i - o_i < 0.5 \\ 0, & \text{else} \end{cases} \tag{5.7}$$

Tab. 5.2: Examination of the Architecture for PowerVcon10F

variant	$\boldsymbol{M_1}$	$\boldsymbol{f(x)}$ **from** $\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{f(x)}$ **from** $\boldsymbol{M_2}$	$\boldsymbol{n_c}$	**RMSE**	$\boldsymbol{\varepsilon_{max}}$	$\boldsymbol{\varepsilon_{cal}}$
a	2	sigmoid			22	0.3482	0.8522	0.210
b	1	tanh			11	0.2982	1.0092	0.270
c	3	sigmoid	2	sigmoid	38	**0.2117**	0.8199	0.120
c-2	3	sigmoid	2	sigmoid	38	**0.1747**	1.0066	**0.090**
d	4	tanh	3	sigmoid	55	0.1939	0.7911	0.130
d-2	4	tanh	3	sigmoid	55	**0.1496**	0.6427	0.100

The most suitable variants are c, c-2 and d-2 and are therefore included in Keras. A thorough error analysis of the training data illustrates that the errors are below 12%. Figure (5.2a) shows a consistent RMSE. Upon inspecting the error depicted in figure (5.2b), a lower RMSE is observed, however, a higher maximum error exists. The corrected error is about half as large (9% instead of 17%) as during pruning in variant 3 (see Fig. G.8).



(a) Learning Curve

(b) Error of the Train Data

Fig. 5.2: PowerVcon10F Variant c-2 (10 linear - 3 sigmoid- 2 sigmoid - 1 linear)

2. PowerVcon7F:
Since the input variables were reduced to 7, to avoid using another dataset, the number of neurons in the hidden layer had to be increased to 15 for the variants with only one hidden layer. The input variables were chosen based on the input of the load flow calculation, which included 1 PV bus and 3 PQ buses. The voltages can be disregarded as they were all the same at initialization.

Tab. 5.3: Examination of the Architecture for Power7F (Pruning)

variant	M_1	$f(x)$ **from** M_1	M_2	$f(x)$ **from** M_2	n_c	n_c **after pruning**	**RMSE**	ε_{max}
1	15	sigmoid			120	17	0.4176	**0.8204**
2	15	tanh			120	8	**0.3210**	0.9730
3	10	sigmoid	10	sigmoid	180	28	0.3680	**0.8829**
4	10	tanh	10	sigmoid	180	24	0.2703	1.0003
5	10	sigmoid	10	tanh	180	0	0.5000	**0.5013**
6	10	tanh	10	tanh	180	23	**0.1903**	0.9973

It turns out that the reduction of the feature vector and its modification results in different results. The result of the pruning is a lower number of connections than the previous one. This could be due to the reduced number of features. It was also tried the configuration variant 5 as variant d, with the pruning number resulted from PowerVcon10F variant 5.

Variant d proved to be the most successful with an error rate of only 3%. This outcome, however, was not entirely expected, as pruning the variant 5 was unsuccessful. The maximum error was also relatively low compared to the other variants, making it the preferred option. The increased number of connections proved beneficial in this instance, but it is not yet approaching the point of overfitting.

3. Power10F:
In the following, the training is conducted with the adapted dataset Power10F, without the voltage contingency. This dataset is different from the initial one and, consequently, cannot be compared. However, the process remains constant.

Tab. 5.4: Examination of the Architecture for PowerVcon7F

variant	M_1	$f(x)$ from M_1	M_2	$f(x)$ from M_2	n_c	RMSE	ε_{max}	ε_{cal}
a	2	sigmoid			22	0.3421	0.8522	0.190
b	1	tanh			11	0.2982	1.0091	0.280
c	3	sigmoid	2	sigmoid	29	0.2187	0.7842	0.130
d	5	sigmoid	2	tanh	47	0.0617	1.0066	0.030
e	3	tanh	2	tanh	29	0.1601	1.0026	0.140

Tab. 5.5: Examination of the Architecture for Power10F

variant	M_1	$f(x)$ from M_1	M_2	$f(x)$ from M_2	n_c	n_c after pruning	RMSE	ε_{max}
1	10	sigmoid			110	15	**0.4283**	0.7579
2	10	tanh			110	0	0.5000	0.5000
3	10	sigmoid	10	sigmoid	210	0	0.5000	0.5001
4	10	tanh	10	sigmoid	210	40	**0.0653**	0.9996
5	10	sigmoid	10	tanh	210	0	0.5000	0.5001
6	10	tanh	10	tanh	210	29	0.1935	0.9990

Overall, the training using this dataset appears to be ineffective, as only 2 variations exist where pruning results in non-zero connections. Based on the test data errors, it is evident that the initial 50 values belong to one classification and the other 50 to the other and the ANN consistently predicted a value near to 0.5, as the RMSE indicates. Nevertheless, the RMSE for variant 4 is the lowest in the whole study and it is therefore analysed further (refer to Tab. 5.6).

Tab. 5.6: Examination of the Architecture for Power10F

variant	M_1	$f(x)$ from M_1	M_2	$f(x)$ from M_2	n_c	RMSE	ε_{max}	ε_{cal}
a	2	sigmoid			22	0.4134	0.6309	0.36
b	3	tanh	3	sigmoid	42	0.3060	0.6374	0.22
b-2	3	tanh	3	sigmoid	42	0.1943	0.8623	0.12

Variant B-2 seems to be the best variant here, but the results are worse than previously determined variants.

4. Power7F:
Again, the number of neurons must be increased for variants with one hidden layer (see Vcon-Power7F).

Tab. 5.7: Examination of the Architecture for Power7F

variant	M_1	$f(x)$ **from** M_1	M_2	$f(x)$ **from** M_2	n_c	n_c **after pruning**	**RMSE**	ε_{max}
1	15	sigmoid			120	11	**0.4604**	0.7656
2	15	tanh			120	0	0.5000	0.5001
3	10	sigmoid	10	sigmoid	180	0	0.5000	0.5013
4	10	tanh	10	sigmoid	180	20	**0.4015**	0.6503
5	10	sigmoid	10	tanh	180	0	0.5000	0.5013
6	10	tanh	10	tanh	180	0	0.5000	0.5013

It shows in Tab. 5.7 comparable results as in Power10F, so that only two variants are successful. Variant 2 and variant 4 will be further evaluated. The number of connections decreases after pruning due to feature reduction.

Tab. 5.8: Examination of the Architecture for Power7F

variant	M_1	$f(x)$ **from** M_1	M_2	$f(x)$ **from** M_2	n_c	**RMSE**	ε_{max}	ε_{cal}
a	2	sigmoid			16	0.4128	0.6499	0.360
b	2	tanh	2	sigmoid	20	0.3393	0.8241	0.270

It turns out that no variant has been found that can keep up with the previous results, so the results have, f.e., a high RMSE.

Conclusion:
The pruning function provide a systematic approach to finding the optimal ANN structure and activation functions for different datasets and requirements. This approach provides an initial network architecture with significantly fewer connections. Models were identified for both datasets (with and without voltage contingencies). However, those that include voltage contingencies achieve better results. The reason for this may be that areas are removed for which the AI based on the ANN is unable to denormalise. Instead of removing data, it would be possible to introduce and train an additional evaluation criterion. Alternatively, the criterion could be combined and the target accepted as positive only if the voltage contingencies are not violated. There is also evidence that models with two hidden layers do better than models with one hidden layer. It is not possible to make a plausible statement about the reduction in features on the basis of the data, but the influence of the reduction will be discussed again at a later chapter.

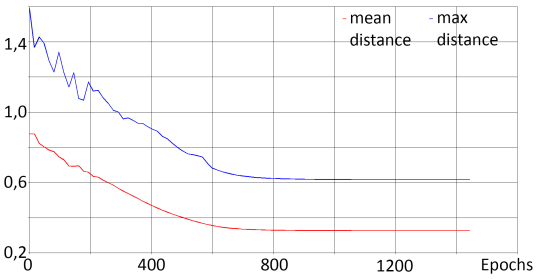
5.3 Kohonen Network

DataEngine also provides the option to utilize the Kohonen network, which was briefly described in Section section 3.4. The Kohonen network utilises classes rather than approximations like MLPs, making it particularly well-suited for differentiating between stable and unstable classifications. The data must be preprocessed to facilitate the presence of these two classes in the dataset, with the feature pattern assigned a value of stable (1) if it matches the class and unstable (0) otherwise. For the design of the Kohonen network, it is crucial to ensure that the number of neurons is less than the patterns of the dataset and that the number of target classes (i.e. stable/unstable) is less than the number of neurons. Additionally, the initialized learning radius should be roughly equal to the number of neurons on the diagonal. This number is equivalent to the number of neurons for quadratic networks, whereas a different number must be used for non-quadratic networks. [17]

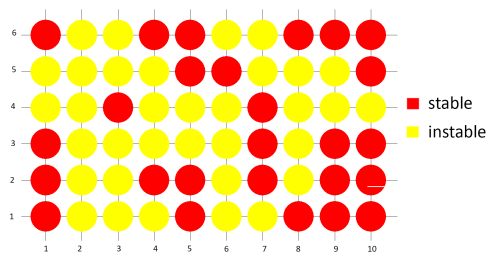
Accuracy is crucial to the evaluation, also in comparison to the MLPs. The dataset of 100 lines without voltage contingencies was the choice. Eight different variants were tested (see Tab. 5.9). See section G.10 in the appendix for the graphs. The settings that are not described are identical and are listed in Tab. G.7. The values for Tab. 5.10 are determined according to the calculated variants.

Tab. 5.9: Variants

variant	neuron dimension 1	neuron dimension 2	number of connections	learning radius	dataset
1	5	5	25	10	100Power7F
2	5	5	25	10	100Power10F
3	8	8	64	8	100Power7F
4	8	8	64	8	100Power10F
5	8	8	64	16	100Power7F
6	8	8	64	16	100Power10F
7	10	6	60	12	100Power7F
8	10	6	60	12	100Power10F



(a) Learning Curve



(b) Kohonenmap

Fig. 5.3: Kohonen Network Variant 8

It is evident that all variants exhibit a decrease in both mean and maximum distances, eventually converging for each variant. This is exemplified in Fig. 5.3a.

Tab. 5.10: Traning,Labelling & Test

variant	mean distance	max. distance	actual learning rate	actual learning radius	labelling error
1	0.3428	0.5393	4.975 E-07	0.0072	0.33
2	0.4103	0.7179	4.925 E-07	0.0072	0.35
3	0.2370	0.4496	4.975 E-07	0.0058	0.25
4	0.2684	0.5492	4.975 E-07	0.0058	0.24
5	0.2835	0.5347	4.975 E-07	0.0115	0.23
6	0.3450	0.7375	4.975 E-07	0.0115	0.31
7	0.2631	0.4684	4.975 E-07	0.0086	0.27
8	0.3261	0.6161	4.975 E-07	0.0086	0.21

The Kohonen map affords a visualisation of the topology of the Kohonen network. The labels are coloured to facilitate the identification of the neuron's respective class. The map also shows whether certain classes are concentrated in neighbourhoods. In Fig. 5.3b, it is evident that no distinct boundary exists and the labels are distributed, rather than being concentrated. None of the Kohonen map variants exhibits such a neighbourhood, which would signify excellent classification by the Kohonen network.

It is not clear from the data obtained that the reduction of the data has an influence on the accuracy due to the variation in the number of features.

Generally speaking, it can be stated that the Kohonen method results in inaccurate labelling errors. These are greater than the errors determined with the MLPs. As a result, this method is not investigated further in this thesis.

5.4 Implementation of MLPs with Keras

The best MLPs designed by DataEngine were used. It is important that the pruning is done with the same optimiser/learning method and settings. This affects the design of the Architecture. For this reason, the DataEngine methodology had to be emulated.

Methodically, a result was reproduced in Keras by transferring the weights determined by DataEngine. However, through this replication, it was possible to determine that DataEngine was actually still training the bias connection. This is relevant as Keras allows turning off the bias, so the number of connections was different. In the case of PowerVcon7F, instead of 59 connections, 69 trainable connections are determined by this analysis.

The data would also need to be normalised. DataEngine has done this for each column (e.g. all values for P_3) by determining the maximum and minimum values and then assigning the values between unstable (0) and stable (1). The normalisation method used here is called interval scaling. However, it should be noted that with a limited amount of data, setting the maximum and minimum values can be problematic. This can result in inputs that exceed the maximum value in the training dataset or fall below the minimum value not being normalised correctly.

Furthermore, there is a correlation between the data as they all describe the same unit/values and influence each other. For this reason, it would also be possible to determine the maximum and minimum values of all columns and use these for normalisation. Alternatively, the values can be set when using synthetically generated data and the limits used there can be used.

After these steps, the transfer was partially successful. Despite the same dataset, the exact accuracy of the training could not be reproduced. There are further steps in the background of the DataEngine that could not be found yet.

Nevertheless, the learning process was subsequently transferred to Keras. Since the pure implementation of the architecture and the learning function was not successful with another optimiser like f.e. "Adam", an attempt was made to copy the learning process of DataEngine. This could not be copied exactly, but similar results were achieved, as no batch size is given in DataEngine, so the best batch size had to be optimised.

The batch size describes the subdivision of the training dataset, the split parts of which are used to improve the MLP and apply backpropagation. An epoch, on the other hand, describes the run through of the entire dataset. Thus, for a dataset of 100 values with a batch size of 50, the backpropagation process is performed twice with the two parts. The composition of the dataset is randomized. This has the advantage of using less memory, bypassing limits and training the MLP faster. Such a limit was detected for the graphics card used (GeForce GTX 1060 - 6 GB) with a batch size of 1 million data. The disadvantage is that with small batch sizes, the estimation of the gradient in the gradient descent process becomes less accurate.

If no batch size is given in Keras, the size of the dataset is divided by four and set as batch size. DataEngine works with the *SGD* optimiser. This allows the learning rate to dynamically decrease as the number of epochs increases, resulting in improved learning. *Mean squared error* is used as the loss function. With other settings, the good results could not be achieved, which means that pruning has to be done for each of these settings. As DataEngine is very old (2002), newer optimizer and loss functions are now available. Keras also provides a pruning library which could be included in the future, f.e. to use the newer *relu* activation function which is not available in DataEngine.

Regarding the threshold for converting into binary classifications, no additional settings are necessary as Keras automatically performs this task. Nevertheless, the implementation may permit another deviation (f.e 0.2) of unstable (0) or stable (1) in theory.

It is noticeable that in Keras significantly longer training times are necessary compared to DataEngine. The training times are strongly influenced by the batch size and the fact that no parallelizable process is possible with the MLP training by Python. Attempts to improve these times have been made, including using the GPU instead of the CPU.

The results of the training in Keras are shown in Tab. 5.11, where the accuracies are listed. The training was performed with the identical datasets as with DataEngine, in order to be able to make a comparison. The investigation was performed in the same process as the evaluation. Graphics can therefore be found in section 5.5 and in section G.11, as they are directly related to the validation.

Tab. 5.11: Accuracy of Keras vs. DataEngine MLP Training

variant	training accuracy in DataEngine	training maximum accuracy in Keras	training time for 10^5 epochs in Keras
PowerVcon10F c	0.88	0.88	2731 s
PowerVcon10F c-2	0.91	0.70	2702 s
PowerVcon10F d-2	0.90	0.84	2643 s
PowerVcon7F d	0.97	0.97	2722 s
Power10F b-2	0.88	0.81	2727 s

High accuracies of up to 97% were achieved in Keras training. A number of cases that appeared good in Dataengine could not be reproduced (see PowerVcon10F, PowerVcon10F d-2 & Power10F b-2). This discrepancy cannot be attributed to the number of epochs, as evidenced by the 58700 epochs required for PowerVcon10 c-2 to achieve this result. One possible explanation is that fluctuations occurred due to the random value used in the Keras training process.

The training process took approximately ¾ hours and was extensively influenced by the batch size selection. The impact of batch size is prominently demonstrated in section 5.5. Here, the different versions are compared based on the number of connections.

5.5 Evaluation

In the previous chapters, the accuracy of the models developed and considered was evaluated using only the training data. In this section, this evaluation is performed using a validation dataset with 20 values that are not present in the training dataset and compared to the evaluation of the training dataset. The basis for this section is the training with the implemented Keras MLP, which performs the validation in each epoch in parallel with the training. For this reason, the confusion matrix (f.e. Fig. 5.5) can also be presented for each epoch of the training. The variants from section 5.4 were considered and the results are summarized in Tab. 5.12.

Tab. 5.12: Accuracy of Keras MLP Training

variant	maximum val accuracy	training accuracy at maximum val accuracy	val accuracy at maximum training accuracy
PowerVcon10F c	0.60	0.61	0.5
PowerVcon10F c-2	0.55	0.50	0.40
PowerVcon10F d-2	0.65	0.51	0.45
PowerVcon7F d	0.65	0.44	0.40
Power10F b-2	0.55	0.50	0.45

It seems that the generalization ability of the trained MLP is low, because although a well trained model is available based on the training data, it cannot respond correctly to unknown values. Another characteristic of this is that at the maximum accuracy of the training dataset, the accuracy of the validation data is less than 50%.

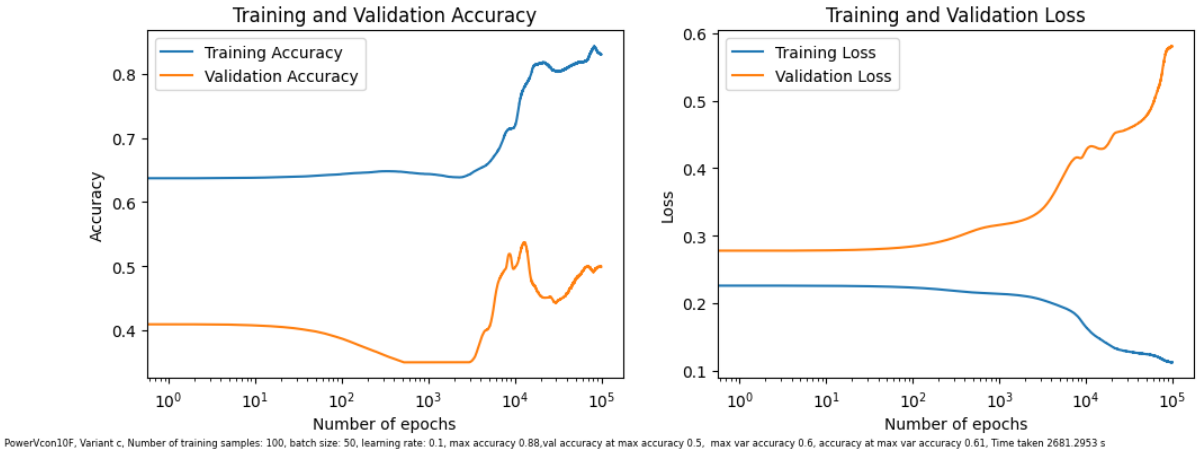


Fig. 5.4: Accuracy and Loss Curve of Keras Training of PowerVcon10F Variant c

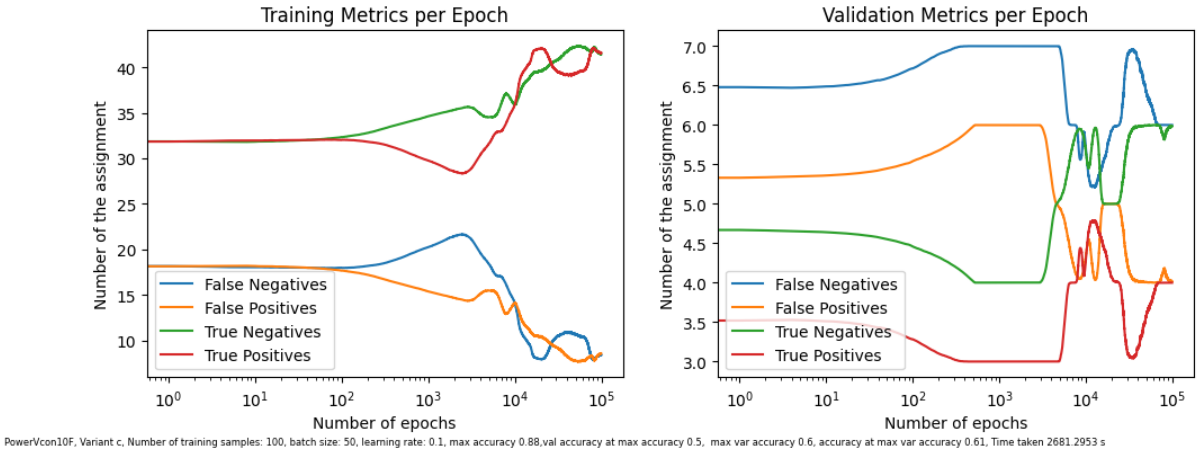


Fig. 5.5: Confusion Matrix Progression During Keras Training of PowerVcon10F Variant c

Fig. 5.4 shows the development of the accuracy and the loss of the training and the validation over the epochs. A logarithmic scale was chosen for the x-axis to show the influence of the number of epochs. For presentation reasons, the values were adjusted with a moving average of 1000 in order to compensate for strong fluctuations and to show the trend. From this graph, it is evident that the accuracy of the training increases, but at a certain point the accuracy of the validation no longer increases with the accuracy of the training. It is notable that the loss of validation does not correspond to the loss of training at any point, indicating a low generalisation ability. The confusion matrix plotted over time, including the moving average and logarithmic x-axis, also shows a noticeable decrease in model accuracy after this point when examining the validation metric (see Fig. 5.5). The training accuracy of PowerVcon10F variant c is currently at 61%. Training with a higher number of epochs beyond this point seems impractical, as it is based on the current training data. Increasing the size of the training dataset may enhance the generalization ability of the ANN, as discussed in section 6.1.

All other images of the other variants can be found in section G.11.

Summary of the Chapter

Since hyper-parameter optimisation in Keras, in which the variable sizes in the MLP are varied, did not lead to successful results, several MLPs were designed using the DataEngine program, which includes the pruning function. This method reduces the number of connections to limit the infinite possibilities, but also limits the study of configurations to a maximum of two hidden layers. Up to four hidden layers are used in other papers like [23]. The study evaluated the impact of the dataset cleaning on voltage contingencies and feature reduction. The next step was to implement an MLP using the Keras Python AI library. The model was optimised using the *mean squared error* loss function and the *SGD* optimiser with impulse and learning rate decay fitting to mimic the learning method from DataEngine. The result showed a low generalisation ability. The current state of the implemented AI based on the MLP is not applicable in practice. Using the MLP with these settings on a binary target did not produce acceptable results. For this reason, the Kohonen map, which is optimised for binary targets, was chosen. However, the investigation of another ANN with the Kohonen map did not lead to more accurate AI models in the training.

6 Scalability and Applicability

6.1 Increase in Data Size

Since section 5.5 has shown that with a training dataset of 100 cases, the model is poorly generalizable, the dataset is subsequently increased. The dataset of 100 cases is based on the MLP design. The increase in the training dataset results in an increase in the training time. Subsequently, the learning process was repeated based on the PowerVcon10F variant c with varying batch sizes. A training dataset of size 1000 (10 times more) was used and a validation dataset of size 200, i.e. 20% of the size of the training data. The results are shown in Tab. 6.1.

Tab. 6.1: Examination of the Architecture for Power7F

batch size	**learning rate**	**maximum training accuracy**	**val. accuracy at maximum training accuracy**	**maximum val accuracy**	**training accuracy at maximum val. accuracy**	**duration**
50	0.1	0.603	0.485	0.570	0.496	10501 s
50	0.01	0.566	0.495	0.565	0.494	10476 s
100	0.1	0.606	0.515	0.575	0.549	6594 s
500	0.1	0.554	0.520	0.560	0.503	2688 s

The impact of batch size on training duration is evident with larger batch sizes showing shorter training times. Conversely, for small datasets, high accuracy is not achieved with the training data even after a certain number of epochs (10^5), so the training can be considered incomplete. Furthermore, maximum validation accuracies with small datasets are in line with those obtained from larger datasets. The table indicates that the validation accuracy is once again low during the epoch with the highest training accuracy. However, it is important to note that the difference between the maximum validation accuracy and the training accuracy during this epoch is not significant, being just over 50%.

Figures 6.1 and 6.2 replicate the plots presented in Section 5.5, with only the number of cases being altered. It is evident from the charts that the selected number of epochs was insufficient. The validation loss also starts to decrease at about $4 \cdot 10^4$. This behaviour can not be determined in diagrams for other variants (see section G.12). The unsteady decrease in training and validation loss indicates a poor generalisation ability.

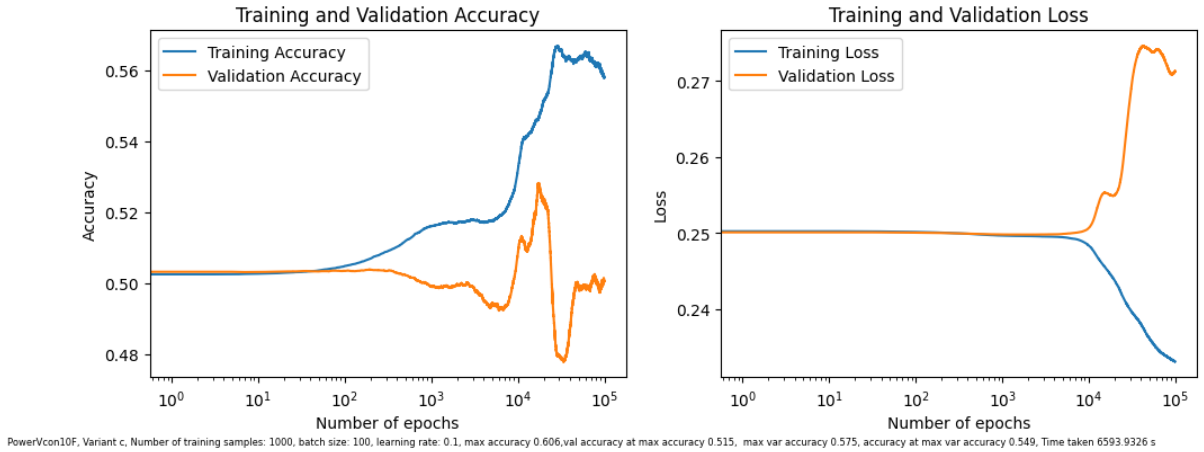


Fig. 6.1: Accuracy and Loss Curve of Keras Training of Power10F Variant c with Batch Size 100, Dataset 1000

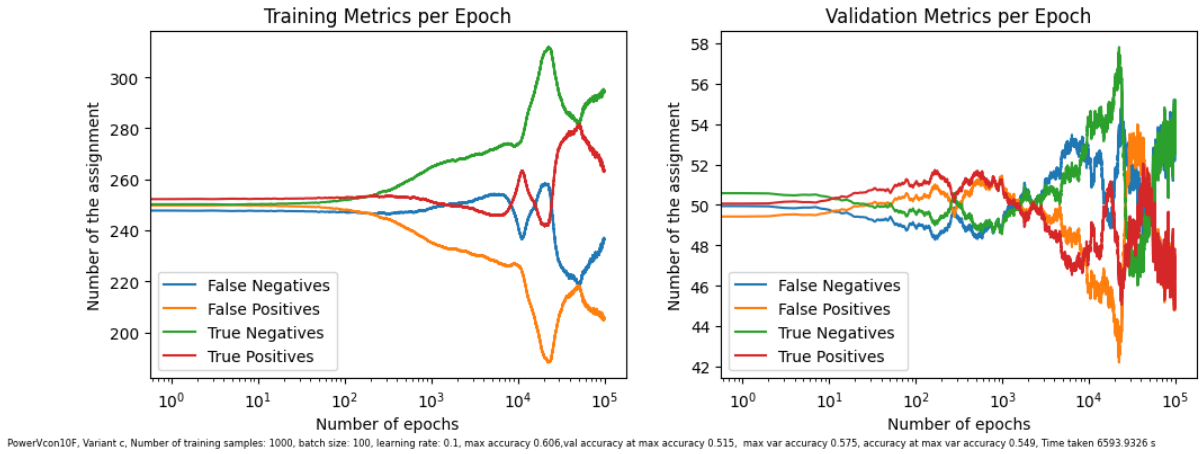


Fig. 6.2: Confusion Matrix Progression During Keras Training of Power10F Variant c with Batch size 100, Dataset 1000

Therefore, and also due to tracking duration, versions were computed with a number of epochs of 10^6. The training stops at a number of epochs of 724967 with an error message [1]. At this point, the accuracy of the training is 0.5730 with a validation accuracy of 0.5, which means that there are no improvements.

The dataset was also scaled to 10000 values as shown in Fig. G.71 and 100000 values as shown in Fig. G.69. Due to training time constraints, the dataset was not scaled beyond 10^3. However, scaling does not show any improvement in generalisation.

Since successful data scaling is not achieved, a considerably large dataset ought to be utilized when designing the MLPs to enhance their generalization ability. Multiple hidden layers might be required or the increased dimensionality of features might need to be considered. To employ the pruning function with a larger dataset, an exceedingly high number of neurons or an increase in the number of features is required.

[1] "F tensorflow/c/logging.cc:43] HRESULT failed with 0x887a0001: dml_device_->GetDeviceRemovedReason()"

6.2 Increase in Power Grid Size

To analyse the influence of grid size, grids were selected first. This was done by selecting grids that were as common as possible, which had been researched previously. Compared to the 5 bus grid considered in this thesis, the smaller 3 bus grid by Gonzalez-Longatt [10] was chosen and implemented in PowerFactory. Settings and the grid itself can be found in section G.2. The IEEE 9 [8] and IEEE 14 [7] buses, available as sample models in PowerFactory, were used for the larger grids. Settings for both grids can be found in the PowerFactory documentation[2].

The next step was to determine the calculation times for the grids (see Tab. 6.2) for the load flow calculation and the stability assessment [25] in order to be able to compare them. For this purpose, 1000 steps were determined for the average calculation time. All calculation times were measured using PowerFactory. The number of processors had an effect on the calculation. The deviation from single step time to total time for 1000 steps is caused by starting PowerFactory via the API. This has already been investigated in Fig. 4.4.

Tab. 6.2: Computing Times for Different Bus Systems

bus system	**nr. of PQ buses**	**nr. PV buses**	**varied values for P_G, Q_G, P_L & Q_L**	**avg. computing time per step**	**computing time per 1000 step**
3 bus [10]	1	1	3	18.9 ms	24.170 s
5 bus	3	1	7	32.2 ms	34.244 s
9 bus [8]	2	3	8	54.0 ms	59.140 s
14 bus [7]	4	11	26	104.1 ms	109.546 s

The minimum number of neurons was then determined from the inputs required for the load flow calculation according to Eq. 5.4. The finally chosen number of neurons in the hidden layers is greater than the minimum number, to provide adjustment of the number during the pruning process. This results in a dataset with at least the minimum number of connections, but less than the chosen number of connections. Hence it allows for overfitting, but also makes the dataset large enough to use for evaluation. In general, an MLP with two hidden layers was assumed. The size of the dataset and the number of hidden layer neurons for pruning were determined according to Tab. 6.3. According to the distribution of the data between stable and unstable conditions, further calculations were performed over the 1000 steps to obtain a uniformly distributed dataset. This is not adjusted for the voltage contingency (see Eq. 3.29).

Because the optimal ANN architecture cannot be found for every process in the form of a variation of the activation function, the best result of the architecture from section 5.2 with tanh in the first hidden layer and with sigmoid in the second hidden layer was chosen. To roughly estimate the learning time of the AI, the duration of the pruning was determined (see Tab. 6.5) in DataEngine. It was set to run for 100000 epochs and an average of 3 runs was taken. Starting from 5 buses, the system does not learn properly, which is not relevant for this consideration.

[2]The files are automatically added during a PowerFactory installation and can be found at `\ProgramFiles\DIgSILENT\PowerFactory2023SP1\localisation\en-GB\examples\Benchmark`

Tab. 6.3: Information About the Buses

bus system	varied values for P& Q	M_n	min. number of connections	chosen M_n	used number of connections	size of the dataset
3 bus	3	3	21	6	60	50
5 bus	7	6	84	10	180	100
9 bus	8	6	90	10	190	150
14 bus	23	17	756	21	945	900

Tab. 6.4: Training Times for Different Bus Systems in DataEngine

bus system	time for 100000 epochs	max. error	rms test error
3 bus	4.967 s	0.4081	0.7480
5 bus	15.293	0.5	0.6492
9 bus	17.960	0.5000	0.6037
14 bus	240.000	0.5000	0.5000

From these results, it can be derived how much more or less time is needed compared to the 5 bus grid. Influences on this are the computing time for the data and the computing time for the training. It was assumed that the number of values corresponds to four times the minimum number of connections, which results in the calculation times shown in Tab. 6.2. The number of epochs also influences the training time.

Tab. 6.5: Comparison for Different Bus Systems with the 5 Bus System

bus system	computing time vs. 5 bus	training time vs. 5 bus	estimated total time for 100 epochs	estimated total time for 100000 epochs
3 bus	59%	32%	15%	25%
9 bus	168%	117%	180%	143%
14 bus	324%	1569%	1535%	1555%

In addition, it is useful to consider the values as a function of the number of buses by dividing the values by the number of buses (see Tab. 6.6).

It can be seen that the computation time and the training time increase with the number of buses, but in an approximately linear relationship. It is noticeable that, f.e., the training time per bus is less for the 9 bus grid than for the 5 bus grid. The reason for this could be the number of varied values. This can also be seen in Tab. 6.6, where the training times were divided by the number of varied values showing similar ratios.

Tab. 6.6: Comparison for Different Bus Systems in Relation to the Number of Buses/ Used Values

bus system	**computing time/ n_B**	**training time/ n_B**	**varied values n_v for P& Q**	**computing time/ n_v**	**min. number of connections**	**training time/ $n_{c_{min}}$**
3 bus	$630 \cdot 10^{-5}$	$1.66 \cdot 10^{-5}$	3	$630 \cdot 10^{-5}$	21	$237 \cdot 10^{-8}$
5 bus	$644 \cdot 10^{-5}$	$3.06 \cdot 10^{-5}$	7	$460 \cdot 10^{-5}$	84	$182 \cdot 10^{-8}$
9 bus	$600 \cdot 10^{-5}$	$2.00 \cdot 10^{-5}$	8	$675 \cdot 10^{-5}$	90	$200 \cdot 10^{-8}$
14 bus	$744 \cdot 10^{-5}$	$17.14 \cdot 10^{-5}$	23	$453 \cdot 10^{-5}$	756	$317 \cdot 10^{-8}$

For the computation time, the number of minimum connections has been used, since in the ideal case the values are close to this value. This also leads to a comparable ratio. In reality, the number of connections may be lower, which affects this result (compared to the pruning results of section G.8). It is necessary to evaluate whether these relations are generally valid, e.g. in Keras or with other programs to compute the criterion.

An analysis of the accuracy could not be performed due to the complexity of the search for the best ANN. The selected configurations from Tab. 6.5 with comparable activation functions were not successful.

6.3 Applicability and Limitation

The developed ANN requires further optimisation as it currently has a limited generalisation ability and has difficulties reacting to unknown values. The increase in the training dataset did not lead to a significant improvement in the generalization. Most importantly, the learning process is not successful, even though the same MLP structure as with the low dataset was used.

Nevertheless, limitations and the application chain will be further discussed in this section.

Regarding the applicability of the AI of the ANN in terms of prediction time, PowerVcon10F variant c was used after 10000 epochs of training with the prediction of different numbers of calculations. It is important to note that the predictions are not accurate, but with successful training (different weights) it is possible to generate comparable times. If necessary, the architecture would have to be adapted. The outcome is visible in Fig. 6.3. It can be seen that the AI based on the ANN takes significantly less time to compute than the previously considered computational methods. In addition, the model requires a specific startup time/load time, as the required time is higher with only one step compared to per step. For 10^4 calculations, the AI is approximately 18 times faster than the Gauss-Seidel implementation, 147 times faster than Newton–Raphson and 768 times faster than PowerFactory.

In its current form, the AI was designed to be used in a grid where no switching operations are performed. However, this would be easy to implement in the data generation via the PowerFactory API, but a suitable way has to be found for the transfer of this information as a feature. Likewise, the limitations of the lines were not taken into account. This can also be used as a target in addition to the static voltage stability criterion and can be easily accessed via the PowerFactory API.

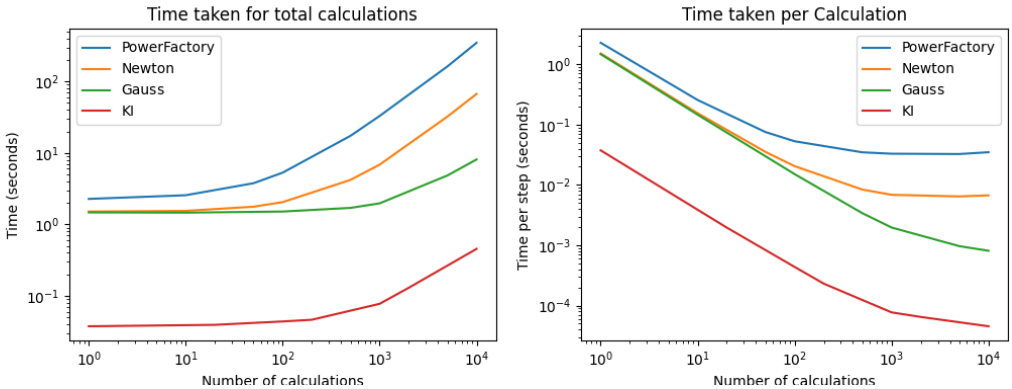


Fig. 6.3: Required Time for the Prediction of the AI Model Variant PowerVcon10F c

Another limitation for an AI with high accuracy in the validation would be the limits of the syntactically generated inputs. Here, they were deliberately oversized so that they did not have to be adapted to the normalization. This would, however, have to be taken into account for other grids.

In addition, the dataset used only considers a balanced grid. An unbalanced grid would have more characteristics as it uses a 3-phase input that can be adjusted. When calculating with PowerFactory, the user can set a balanced or unbalanced grid.

Summary of the Chapter

When the training dataset was expanded, no successful ANN training could be identified. This means that the number of features is insufficient and needs to be increased. Reducing the number of features is therefore not recommended. The effect of grid scaling on the computation and training time for MLPs was also analysed. Based on this analysis and approach, it is assumed that a general AI covering all grid scales will require a very high computation and training time. Furthermore, limitations of the current approach were identified, as well as the great advantages in computing time.

7 Summary & Outlook

In this thesis, the development of an artificial intelligence (AI) based on an artificial neural network (ANN) to evaluate grid conditions in terms of static voltage stability was investigated. The aim of this approach is to reduce the computation time for this purpose compared to existing methods.

The merit of this thesis is not only in working with an existing dataset, but also in considering all the steps necessary to create the dataset and then implement the ANN.

To accomplish this task, the initial step involved performing load flow calculation on a 5 bus grid with synthetic cases generated using PowerFactory via its Python API. In addition to the PowerFactory calculation, the Gauss-Seidel & Newton-Raphson methods were examined. These cases were than evaluated using the static voltage stability criterion. This procedure created a dataset essential for ANN training, validation and testing purposes.

Several MLPs were designed using the DataEngine program, which included the pruning function. The study assessed the impact of dataset cleaning for voltage contingencies and feature reduction. In the final step, an MLP was implemented by using the Keras Python AI library. The model was optimized with the *mean squared error loss* function and the *SGD* optimizer with momentum and learning rate decayed fitting to mimic Dataengin's learning method. The result showed a low generalisation ability. With the expansion of the training dataset, no successful training of the ANN could be identified. This implies that the count of features is insufficient and needs to be increased. Therefore, reducing the examined features is not recommended. The current state of the implemented ANN is not applicable in practice.

The increase in features would also have the advantage that the number of neurons in the hidden layers, estimated according to the usual formulas, would be larger and the datasets used for the pruning process could be larger. Nevertheless, there must be a subsequent check as to whether there really is a greater capacity for generalisation. Another possibility is to predict the dV/dQ-sensitivity and then evaluate whether all values are greater than or equal to zero.

Using the MLP with this setting for a binary target did not produce acceptable results. For this reason, the Kohonen map, which is optimised for binary targets, was chosen. However, the investigation with the Kohonen map did not lead to a more accurate AI model.

The effect of power grid scaling on computation and training time for MLPs was also analysed. It is assumed that based on this analysis and approach a general AI, which covers all grid scales, requires a very high computation and training time.

Another way that can lead to an AI-based assessment of static voltage stability is to perform the load flow calculation with AI. The determined voltages and load flows are then used to evaluate the grid state with the correct implementation of the static voltage stability criterion. The implementation of such a load flow calculation AI has been investigated (section G.9). This approach has the advantage that a considerable amount of data can be used for the design of the ANN and the accurate estimation of the voltages.

In addition to searching for the best ANN configuration via the pruning algorithm, hyperparameter optimization can also be used, in which the variable sizes in the ANN are varied. This involves a lot of computational effort, but can also lead to successful results. This approach was used before pruning, but was not successful and therefore has not been described here.

The generated dataset will be made open-access so that other researchers can analyse this problem with the proposed variants, more modern optimizers or other ANNs.

The utilization of AI provides the opportunity to ascertain a stable condition based on a target function through PSO, like minimizing load changes, whilst considering load fluctuations or battery injection to a stable state space. Instead of performing a complicated load flow calculation, AI can determine the target and work with a larger population of particles, which results in faster state determination.

A Bibliography

[1] Peter Schegner Andreas Winter Michael Igel. "Machine learning based grid optimization algorithm for real-time applications". In: *CIRED Rome* (2023).

[2] Anjali Atul Bhandakkar and Lini Mathew. "Real-Time-Simulation of IEEE-5-Bus Network on OPAL-RT-OP4510 Simulator". In: *IOP Conference Series: Materials Science and Engineering* 331.1 (2018), p. 012028. ISSN: 1757-8981. DOI: 10.1088/1757-899X/331/1/012028. URL: https://dx.doi.org/10.1088/1757-899X/331/1/012028.

[3] Stefan Bischoff. "MPL Lecture". In: 2019.

[4] Stefan Bischoff. "Perceptron lecture". In: 2019.

[5] Agron Bislimi. "Einfluss von Spannungsstabilitätsproblemen auf die Sicherheit elektrischer Energienetze". PhD thesis. TU Wien, 2012.

[6] Rengin İdil Cabadağ. "Analysis of the Impact of ReactivePower Control on Voltage Stability inTransmission Power Grids". PhD thesis. TU Dresden, 2019.

[7] DIgSILENT. *14 Bus Network*. System description a example, 2023.

[8] DIgSILENT. *9*. System description as example, 2023.

[9] German Standards Institute/ Deutsche Institut für Normung e.V. (DIN). "DIN EN 50160". In: (2020). Part 1: General principles (IEC 62305-1:2010, modified).

[10] Francisco M. Gonzalez-Longatt. *FGL's 3 bus Test System*. 2007. URL: https://fglongatt.org/OLD/Test_Case_FGL's3bus.html.

[11] Ahmad Harb Hamza Alnawafah. "Modeling and Validation of Jordanian Power Grid in DIgSILENT PowerFactory Toward Implementing a Smart Grid Scenario". In: *IREC 2021* (2021).

[12] Johanna Hartweg. "Analyse des Einflusses verschiedener Arbeitspunkte auf die wirkleistungsabhängige Spannungssensitivität an einem Mittelspannungsnetzknoten". MA thesis. Ostbayerische Technische Hochschule Regensburg, June 2021.

[13] J. Heaton. *Introduction to Neural Networks for Java*. 2nd ed. 1. Heaton Research, Inc., 2008. ISBN: 978-1-60439-008-7.

[14] Johann Jäger. *Wissensbasis Maschinelles Lernen – Neuronale Netze*. FAU. 2023.

[15] Kristina Jurczyk, Sebastian Wende von Berg, and Kurt Brendlinger. *Datenanalyse und Künstliche Intelligenz im Stromverteilnetz*. Tech. rep. Joseph-Beuys-Straße 834117 Kassel: Fraunhofer-Institut für Energiewirtschaft undEnergiesystemtechnik (IEE), Aug. 2022. URL: https://future-energy-lab.de/app/uploads/2023/01/Fraunhofer_Gutachten_KI_im_Stromnetz.pdf.

[16] *Keras Wiki -Optimizers*. https://keras.io/api/optimizers/.

[17] W. Kästner. "Lecture: KNN – Kohonen Netzwerk". In: *Klassifikation mittels Kohonen Netzwerk*. 2021.

[18] W. Kästner. “Lecture: Künstliche Neuronale Netze”. In: *Modellierung mittels Multilayer Perzeptron*. 2021.

[19] James Matthew Kulaga. “Methodologies for Voltage Contingency Ranking”. MA thesis. Purdue University, 1991.

[20] Prabha Kundur. *Power System Stability and Control*. New Edition. Elektric Power Research Institue (EPRI), 1994. ISBN: 978-0070359581.

[21] Oliver Lang. “Spannungsqualität, ein Durchblick”. In: *Device GmbH* (2009).

[22] Prof. Dr.-Ing. Christian Rehtanz. “Powerpoint from lecture”. In: *Module: Dynamics and stability of energy transfer systems (dt. Modul: Dynamik und Stabilität von Energieübertragungssystemen)*. TU Dortmund, 2022.

[23] Andreas Winter; Michael Igel; Peter Schegner. “Application of artificial intelligence in power grid state analysis and -diagnosis”. In: *NEIS Conference Hamburg 2020*. htwsaar and TU Dresden. VDE, 2020.

[24] Andreas Winter; Michael Igel; Peter Schegner. “Supervised Learning Approach for State Estimation in Distribution Systems with missing Input Data”. In: *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)* (2021).

[25] Dr. Martin Schmieg. “P2021 Gutachten zur NEMO VIII, Los 1 - Stabilität (Kurzfassung)”. In: *DIgSILENT GmbH* (2022).

[26] Kremens; Sobierajski. *Analiza systemów elektroenergetyczny (in English: Analyse of electro energy systems)*. Wydawnictwa Naukowo-Techniczne Warszawa, 1996.

[27] G. Theil. “Statische Stabilität von Stromnetzen - Erfahrungen bei Analyse realer Systeme”. In: *Alternativen für die Energiezukunft Europas* (2012), pp. 1–10.

[28] *Tutorials und Grundlagen*. MIT GmbH, 1998.

[29] Umweltbundesamt. “Was ist ein „Smart-Grid“?” In: *UBA fragen* (2013).

[30] Andreas Winter et al. “Monitoring in Niederspannungsnetzen mit Verfahrender künstlichen Intelligenz – Validierung derMethodik in einem realen Stromnetz”. In: *17. Symposium Energieinnovation* (2022).

B List of Figures

C List of Tables

D Formula Directory

E Symbol Director

A

A	number of input neurons

B

B	number of output neurons
$B^{'}$	susceptance

E

$\varepsilon_{\mathrm{cal}}$	calculated error
$\varepsilon_{\mathrm{max}}$	max. error

I

$\underline{I}$	current (complex)
$\underline{\boldsymbol{I}}$	current matrix (complex)

J

j	index describes the bus connected to the bus
j	imaginary part
J_{R}	reduced Jacobi matrix

M

M_{x}	number of hidden layer neurons

N

n_{c}	number of connections
n_{t}	number of training patterns

O

o	output signal

P

P	active power
P_{G}	active power generator
P_{L}	active power load

Q

Q	reactive power
Q_{G}	reactive power generator
Q_{L}	reactive power load

R

r	relevance
R	resistance
$RMSE$	root mean square error

S

S	apparent power (magnitude)
S_{Base}	complex power basis when using p.u.
$\underline{S}$	complex power
$\underline{\boldsymbol{S}}$	complex power matrix

T

t	target

V

v	iteration index
V	voltage (magnitude)
V	voltage (complex)
V_{Base}	voltage basis when using p.u.
V_{n}	nominal voltage
$\underline{V}$	voltage (complex)
$\underline{\boldsymbol{V}}$	voltage matrix (complex)

W

w	number of buses
w	weights
Δw	weight difference

X

x	reactance

Y

Y	admittance (magnitude)
Y	admittance (complex)
Y_{M}	admittance matrix
$\underline{Y}$	admittance (complex)
$\underline{y}_{ij}^{'}/2$	susceptance

Z

Z	impedance (magnitude)
Z_{Base}	impedance basis when using p.u.
$\underline{Z}$	impedance (complex)
$\underline{\boldsymbol{Z}}$	impedance matrix (complex)

Greek

α	learning rate
δ	voltage angle
μ_{ij}	admittance angle
ϑ_k	threshold (Bias)
τ_{P}	time constant in epochs of the low pass filter

F List of Abbreviations

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
AC	alternating current
AI	artificial Intelligence
ANN	artificial neural network
API	application programming interface
CPU	central processing unit
CUDA	compute unified device architecture
CuDNN	CUDA deep neural network
DC	direct current
DIN	German Institute for standardisation registered association
FFN	feed-forward network
f.e.	for example
f.i.	for instance
GPU	graphics processing unit
HV	high-voltage
HSZG	University of Applied Sciences Zittau/Görlitz
ICT	information and communication technologies
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
LV	low-voltage
MAE	mean absolute error
MSE	mean squared error
ML	machine learning
MLP	multi-layer perceptron
MV	medium-voltage
NEMO	nominated electricity market operator
NEP	network development plan
p.u.	per unit
PF	PowerFactory
PQ bus	active and reactive power bus
PSO	particle swarm optimization
PV	photovoltaic
PV bus	active power and voltage bus
Q-Q plot	quantile-quantile plot
rms	root mean square
SGD	stochastic gradient descent
SLP	single-layer perceptron
SOM	self-organizing maps
UCTE grid	synchronous grid of continental Europe

G Appendix

G.1 Conversion p.u. to Values for IEEE5 Bus Grid Based on Tab. 4.1

Use of p.u. in tree phase grid:
$Z_{\text{Base}} = V_{\text{Base}}^2 / S_{\text{Base}}$

$Y_{\text{Base}} = 1/(V_{\text{Base}}^2 / S_{\text{Base}})$

r in p.u. to R:
$R = Z_{\text{Base}} \cdot r$

in this case $V_{\text{Base}} = 230kV$ and $S_{\text{Base}} = 100\text{MVA}$
for $r = 0.01p.u.$ transformed $R = 5.29\Omega$

x in p.u. to X:
$X = Z_{\text{Base}} \cdot x$
for $r = 0.085p.u.$ transformed $R = 44.965\Omega$

$b/2$ in p.u. to B:
$B' = (b/2 * 2)/1/(V_{\text{Base}}^2 / S_{\text{Base}})$ or $b = B'/Y_{\text{Base}}$
for $b/2 = 0.088p.u.$ transformed $B = 332.7\mu\text{S}$

Tab. G.1: Data of Lines in the IEEE 5 Bus Grid in the PowerFactory Model

Line	**From**	**To**	**R in [Ω]**	**X in [Ω]**	**B in [Ω]**
Line 1-2	1	2	10.58	31.74	113.42
Line 1-3	1	3	42.32	126.96	94.58
Line 2-3	2	3	31.74	95.22	75.61
Line 2-4	2	4	31.74	95.22	75.61
Line 2-5	2	5	21.16	63.48	56.71
Line 3-4	3	4	5.29	15.87	37.81
Line 4-5	4	5	42.32	126.96	94.52

G.2 FGLs 3 Bus Grid Based on [10]

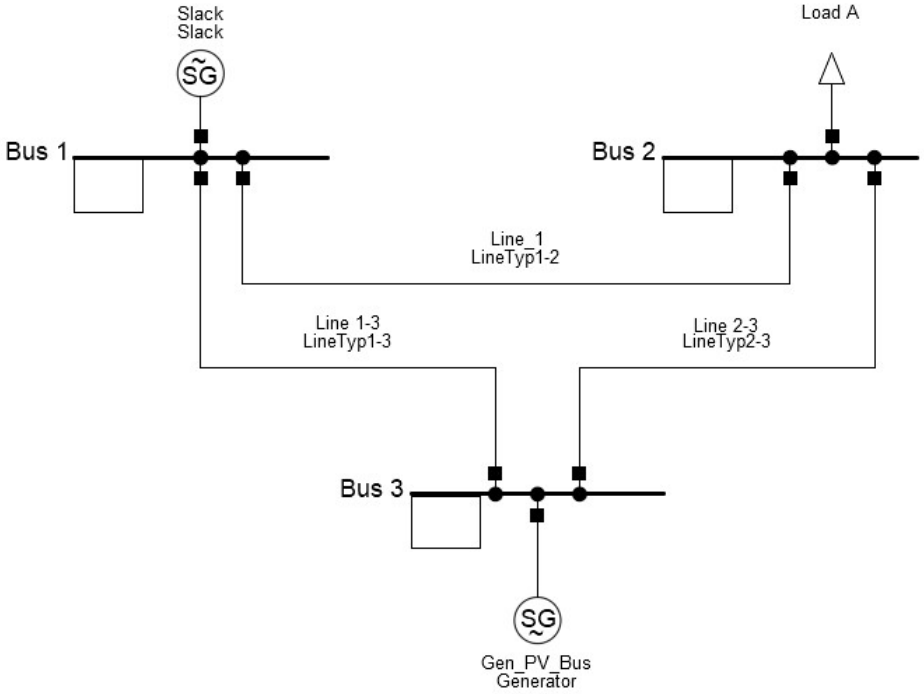


Fig. G.1: 3 Bus Grid

According to Gonzalez-Longatt [10] for his 3 Bus-Grid (see Fig. G.1) the following values, Tab. G.2 and Tab. G.3 are given:

$S_{\text{Base}} = 100MVA$
$V_{\text{Base}} = 115\text{kV}$
$Z_{\text{Base}} = 132.25\Omega$

Tab. G.2: Data of Lines in the 3 Bus Grid in PF

from	**to**	**r in [p.u.]**	**x in [p.u.]**
1	2	0.0400	0.0200
1	3	0.0300	0.0100
2	3	0.0250	0.0125

These values result in Tab. G.4. The model differs from Gonzalez-Longatt [10] in that the reactive power on the PV bus (3 bus) exceeds its limit. It was set in PowerFactory, but it exceeds this value, which is inexplicable at the time. It is also interesting to note that in his example the voltage on the PV bus changes, although it should remain the same, as indicated by the PV bus. However, as the aim is to give a rough overview of the calculation time, this should not impact on the result.

Tab. G.3: Buses in the 3 Bus Grid Based on 100 MVA [10]

bus	**typ**	**V in [p.u.]**	**V in [°]**	P_G **in [MW]**	Q_G **in [MW]**	P_L **in [MW]**	Q_L **in [MW]**
1	slack	1.05	0	0	0	0	0
2	PQ	1.00	0	0	0	400	200
3	PV	1.04	0	0	200	0	0

Tab. G.4: Data of lines in the IEEE 5 Bus Grid in PF

Line	**From**	**To**	**R in [Ω]**	**X in [Ω]**
Line 1-2	1	2	5.290	2.645
Line 1-3	1	3	3.968	1.323
Line 2-3	2	3	3.306	1.653

G.3 Testdata for Proper Implementation

Y bus matrix:

$$\boldsymbol{Y} = \begin{bmatrix} 0.01181474 - 0.03523623\text{j} & -0.00945180 + 0.02835539\text{j} & -0.00236295 + 0.00708885\text{j} \\ -0.00945180 + 0.02835539\text{j} & 0.02047889 - 0.06111532\text{j} & -0.00315060 + 0.00945180\text{j} \\ -0.00236295 + 0.00708885\text{j} & -0.00315060 + 0.00945180\text{j} & 0.02441714 - 0.07304342\text{j} \\ 0.00000000 + 0.00000000\text{j} & -0.00315060 + 0.00945180\text{j} & -0.01890359 + 0.05671078\text{j} \\ 0.00000000 + 0.00000000\text{j} & -0.00472590 + 0.01417769\text{j} & 0.00000000 + 0.00000000\text{j} \end{bmatrix}$$

$$\begin{bmatrix} 0.00000000 + 0.00000000\text{j} & 0.00000000 + 0.00000000\text{j} \\ -0.00315060 + 0.00945180\text{j} & -0.00472590 + 0.01417769\text{j} \\ -0.01890359 + 0.05671078\text{j} & 0.00000000 + 0.00000000\text{j} \\ 0.02441714 - 0.07304348\text{j} & -0.00236295 + 0.00708885\text{j} \\ -0.00236295 + 0.00708885\text{j} & 0.00708885 - 0.02111531\text{j} \end{bmatrix}$$

This implementation help use the following input. Compared to Tab. 4.2, it is without a Load at Bus 2:

Tab. G.5: Buses in the 5 Bus Grid, 100 MVA, Testimplementation

bus	**typ**	**V in [p.u.]**	**V in [°]**	$\boldsymbol{P_G}$ **in [MW]**	$\boldsymbol{Q_G}$ **in [MW]**	$\boldsymbol{P_L}$ **in [MW]**	$\boldsymbol{Q_L}$ **in [MW]**
1	slack	1.06	0	0	0	0	0
2	PV	1.00	0	40	0	0	0
3	PQ	1.00	0	0	0	45	15
4	PQ	1.00	0	0	0	40	5
5	PQ	1.00	0	0	0	60	10

Gauss-Seidel method:

$$\boldsymbol{U}_{\mathbf{Gauss}}^{(v=1)} = \begin{bmatrix} 243800.00000000 + 0.\text{j} \\ 229956.99789458 + 4447.37218034\text{j} \\ 233731.14173993 + 2521.80892707\text{j} \\ 234664.86047509 + 4389.17603675\text{j} \\ 239639.67027748 + 14455.81896748\text{j} \end{bmatrix}$$

Newton–Raphson method:

$$\boldsymbol{U}_{\mathbf{Newton}}^{(v=1)} = \begin{bmatrix} 243800.00000000 + 0.\text{j} \\ 229018.64290882 + 21224.07124475\text{j} \\ 236394.31269463 + 34386.18966541\text{j} \\ 231623.01640550 + 34206.48190411\text{j} \\ 241605.88528068 + 37358.80549657\text{j} \end{bmatrix}$$

Jacobian matrix based on $\boldsymbol{U}_{\mathbf{Newton}}^{(v=1)}$:

$$\boldsymbol{J}^{v=1} = \left[\begin{array}{c|c} \boldsymbol{J}_{\mathbf{1}}^{v=1} & \boldsymbol{J}_{\mathbf{2}}^{v=1} \\ \hline \boldsymbol{J}_{\mathbf{3}}^{v=1} & \boldsymbol{J}_{\mathbf{4}}^{v=1} \end{array}\right]$$

$$\boldsymbol{J}_{\mathbf{1}}^{v=1} = \begin{bmatrix} 1.99e+09 & -1.59e+09 & -3.98e+08 & 0.00e+00 & 0.00e+00 \\ -1.59e+09 & 3.34e+09 & -5.00e+08 & -5.00e+08 & -7.50e+08 \\ -3.98e+08 & -5.00e+08 & 3.90e+09 & -3.00e+09 & 0.00e+00 \\ 0.00e+00 & -5.00e+08 & -3.00e+09 & 3.88e+09 & -3.75e+08 \\ 0.00e+00 & -7.50e+08 & 0.00e+00 & -3.75e+08 & 1.12e+09 \end{bmatrix}$$

$$\boldsymbol{J}_{\mathbf{2}}^{v=1} = \begin{bmatrix} 3.04e+03 & -2.30e+03 & -5.76e+02 & 0.00e+00 & 0.00e+00 \\ -2.17e+03 & 4.58e+03 & -7.25e+02 & -7.25e+02 & -1.09e+03 \\ -5.43e+02 & -7.25e+02 & 5.58e+03 & -4.35e+03 & 0.00e+00 \\ 0.00e+00 & -7.25e+02 & -4.35e+03 & 5.62e+03 & -5.43e+02 \\ 0.00e+00 & -1.09e+03 & 0.00e+00 & -7.25e+02 & -1.09e+03 \end{bmatrix}$$

$$\boldsymbol{J}_{\mathbf{3}}^{v=1} = \begin{bmatrix} -6.63e+08 & 5.30e+08 & 1.33e+08 & 0.00e+00 & 0.00e+00 \\ 5.30e+08 & -1.11e+09 & 1.67e+08 & 1.67e+08 & 2.50e+08 \\ 1.33e+08 & 1.67e+08 & -1.30e+09 & 1.00e+09 & 0.00e+00 \\ 0.00e+00 & 1.67e+08 & 1.00e+09 & -1.29e+09 & 1.25e+08 \\ 0.00e+00 & 2.50e+08 & 0.00e+00 & 1.25e+08 & -3.75e+08 \end{bmatrix}$$

$$\boldsymbol{J}_{\mathbf{4}}^{v=1} = \begin{bmatrix} 9.03e+03 & -6.91e+03 & -1.73e+03 & 0.00e+00 & 0.00e+00 \\ -6.52e+03 & 1.36e+04 & -2.17e+03 & -2.17e+03 & -3.26e+03 \\ -1.63e+03 & -2.17e+03 & 1.67e+04 & -1.30e+04 & 0.00e+00 \\ 0.00e+00 & -2.17e+03 & -1.30e+04 & 1.68e+04 & -1.63e+03 \\ 0.00e+00 & -3.26e+03 & 0.00e+00 & -1.63e+03 & 4.82e+03 \end{bmatrix}$$

G.4 Contradiction in the Static Voltage Stability Criteria

This section discusses the stability criteria by the sensitivity analyse [25] and modal analysis [22].

Sensitivity analyse:

> "Static voltage stability refers to the ability of a power supply system to maintain the required supply voltage at all grid nodes in a steady state. This must also apply to planned outage situations. Static voltage stability is always given if the (dV/dQ) sensitivity at each grid node is positive for all states under consideration." [25]

By the values at each node/in each grid bus is meant the diagonal of the (dV/dQ) sensitivity [6].

Modal analysis:
Rehtanz explained in his lecture [22] that static voltage stability exists when the eigenvalues of J_R are positive. There is also instability when an eigenvalue is 0 and $\det(J_\mathrm{R}) = 0$.

It applies Eq. G.1.

$$(\frac{\Delta V}{\Delta Q})^{-1} = J_R \tag{G.1}$$

The following are examples that show that there are differences.

Example 1:
given term: $J_\mathrm{R} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$
calculations:
$\det(J_\mathrm{R}) = 1$

Eigenvalues of J_R are 1 and 1, therefore the conditions are fulfilled according to modal analysis and the system is stable.
$\frac{dV}{dQ} = J_\mathrm{R}^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$
$\mathrm{dia}(J_\mathrm{R}) = \begin{bmatrix} 1 & 1 \end{bmatrix}$

According to the sensitivity analyse, the system is stable.

Example 2:

given term:
$\frac{dV}{dQ} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$
According to sensitivity analyse, the system is stable.

calculations: $\left(\frac{dV}{dQ}\right)^{-1} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}^{-1} \rightarrow$ Error

The inversion is not possible! Modal analysis not possible

Example 3:

given term:

$$\frac{dV}{dQ} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}$$
$$\mathrm{dia}(J_\mathrm{R}) = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

According to sensitivity analyse, the system is stable.

calculations:
$$\left(\frac{dV}{dQ}\right)^{-1} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}^{-1} \rightarrow J_\mathrm{R} = \begin{bmatrix} -1 & 2 \\ 1 & -1 \end{bmatrix}$$
$$\det(J_\mathrm{R}) = -1$$

Eigenvalues of J_R are 0.4142 and -2.4142, therefore according to modal analysis the conditions are not fulfilled and the system is unstable.

In the result it can be stated that these criteria do not correspond with the values given as examples

For this reason, the sensitivity analyse criterion is used.

G.5 GitHub

The thesis as well as the programs used/created in this thesis are published on GitHub. For this purpose, the version control system Git was used and uploaded to the GitHub servers. The benefit of the version control system in connection with the server is that subsequent changes beyond this work can be tracked and improvements can be made.

The GNU GPLv3 license was selected for publication in order to allow for commercial and private utilization, distribution and modification. This choice permits the processes studied to be tracked and refined. Python was utilized in this thesis, thus endorsing the open-source initiative, however, a PowerFactory license is still required in its current form. In the future, the use of own implementations or open systems such as PyPSA (Python for Power System Analysis) should be considered.

```
git clone timon0conrad/static-voltage-stability-AI
```

The requirements for utilizing the model approach according to GNU GPLv3 dictate that any modifications made and distributed must be documented and made publicly available. Additionally, releases are exclusively permitted under the same license (=GNU GPLv3) and with copyright notices.

G.6 Dataset

Datasets were created based on the approaches described in section 5.1. The analysis of these datasets can be found below.

Dataset of 100 rows based on PowerFactory with voltage contingencies:

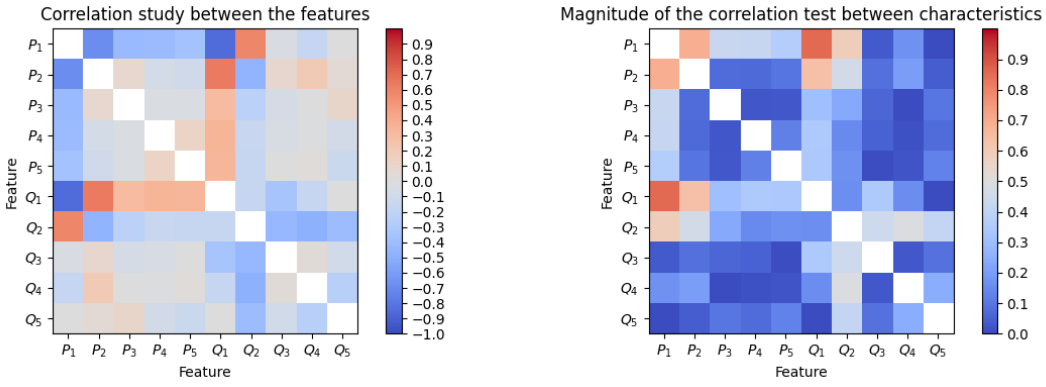


Fig. G.2: Correlation of the Feature, With Voltage Contingencies

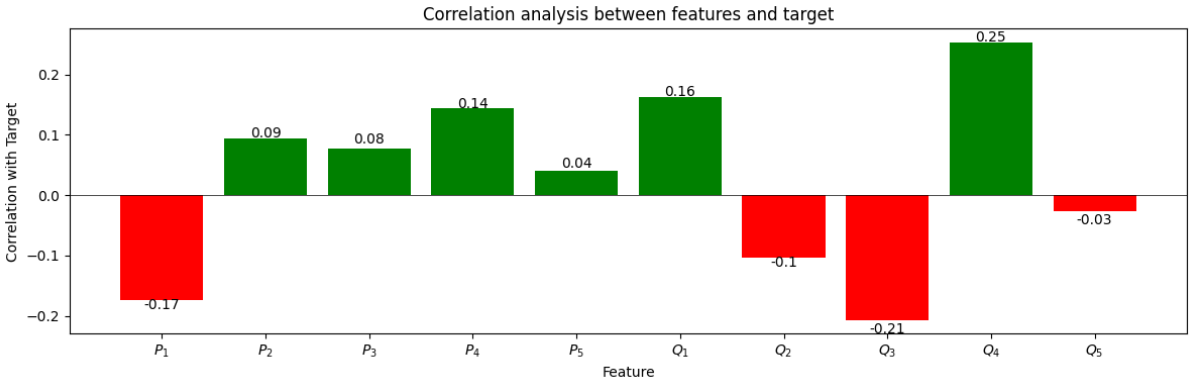


Fig. G.3: Correlation of the Target and the Feature, With Voltage Contingencies

Dataset of 100 rows based on PowerFactory, without voltage contingencies:

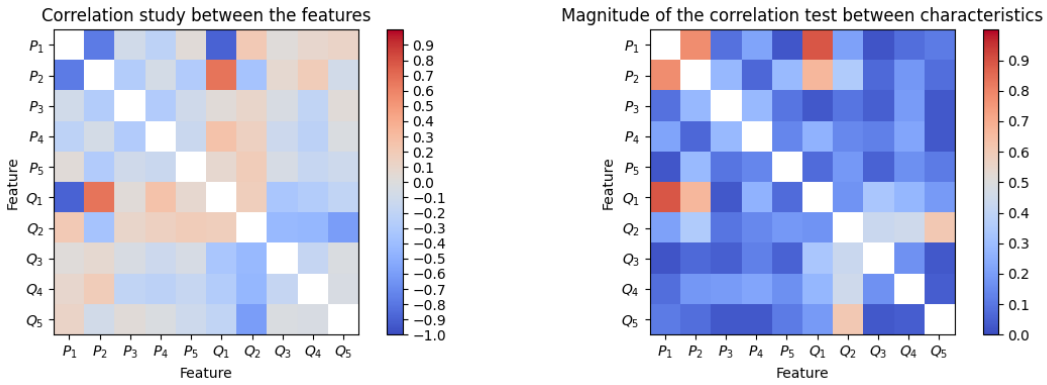


Fig. G.4: Correlation of the Feature, Without Voltage Contingencies

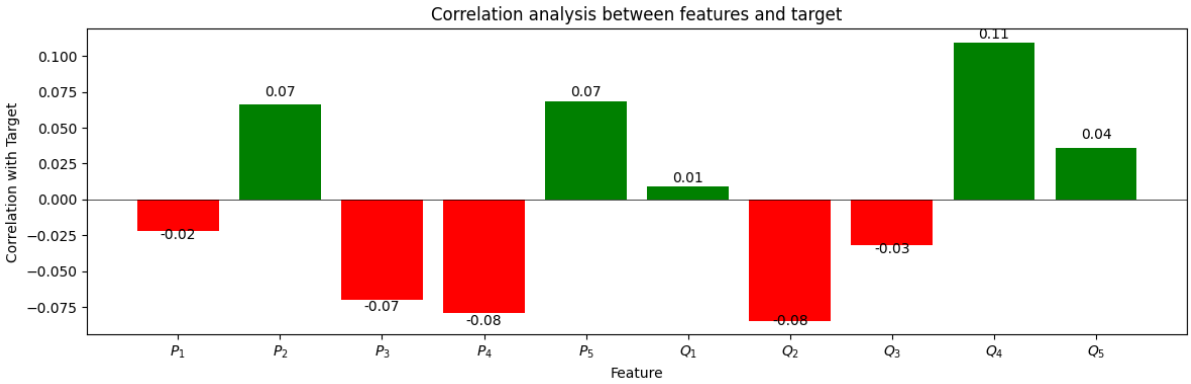


Fig. G.5: Correlation of the Target and the Feature, Without Voltage Contingencies

G.7 Settings for DataEngine

Since the settings are in German, the following Tab. G.6 and Tab. G.7 are in German.

Tab. G.6: Settings for the Pruning (MLPs)

setting	**value**
Lernverfahren	Backpropagation
Momentum	An
Lernstrategie	Einzellschritte
Präsentierreinfolge	zufällig
Pruning	An
Lernrate aller Schichten	0.1
Momentum aller Schichten	0.1
Gewichts-Decay aller Schichten	0.9999
Lernraten-Decay	Aus

Tab. G.7: Settings for the Kohonen Network

setting	**value**
Untere Grenze	0
Obere Grenze	1
Startwert	Aus
Lernrate	0.999
Lernratenfaktor	0.99
Lernradiusraktor	0.995
Präsentationsreinfolge	zufällig

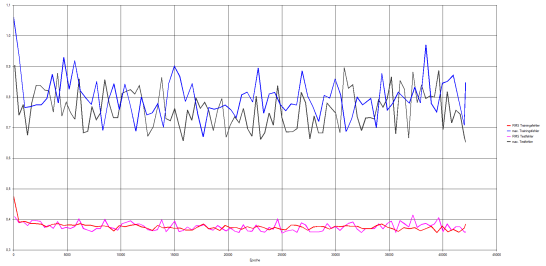
G.8 MLP Design with DataEngine

G.8.1 PowerVcon10F

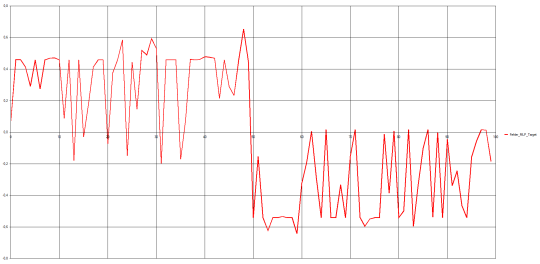
Dataset of 100 rows with voltage contingency

Pruning

One Hidden layer:

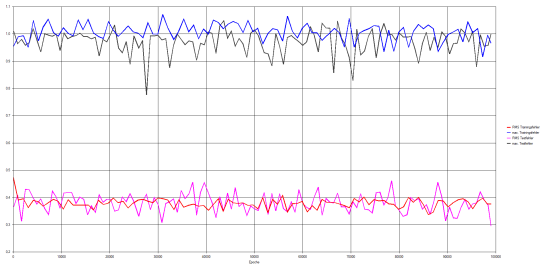


(a) Learning Curve

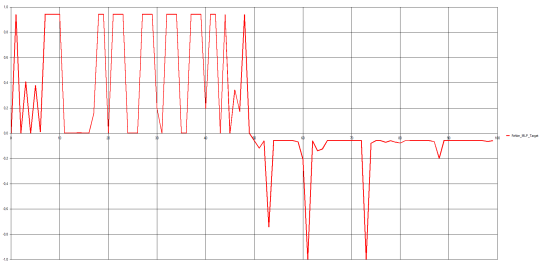


(b) Error of the Train Data

Fig. G.6: PowerVcon10F Variant 1(10 linear -10 sigmoid -1 linear)



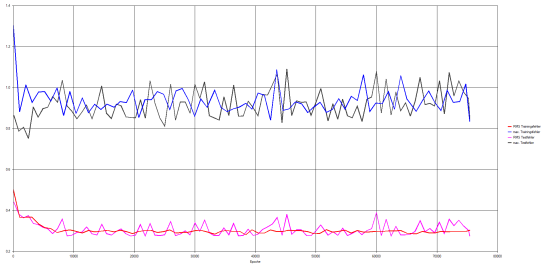
(a) Learning Curve



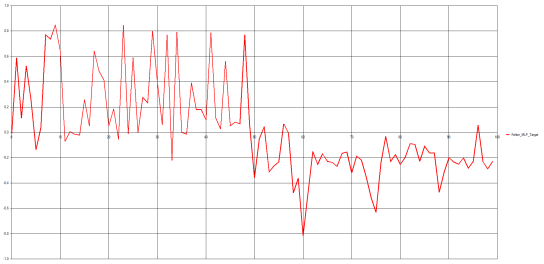
(b) Error of the Train Data

Fig. G.7: PowerVcon10F Variant 2 (10 linear -10 tanh -1 linear)

Two Hidden layer:



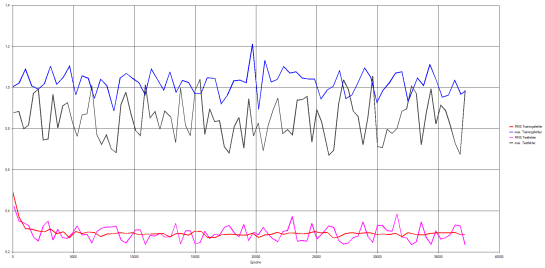
(a) Learning Curve



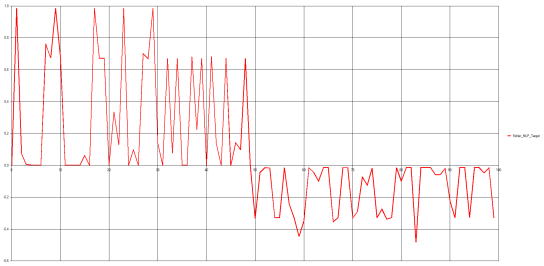
(b) Error of the Train Data

Fig. G.8: PowerVcon10F Variant 3 (10 linear -10 sig - 10 sig - 1 linear)

Power10F Variant 4 (10 linear -10 tanh - 10 sig - 1 linear) can be found as Fig. 5.1a and Fig. 5.1b in the main part in the section 5.2.

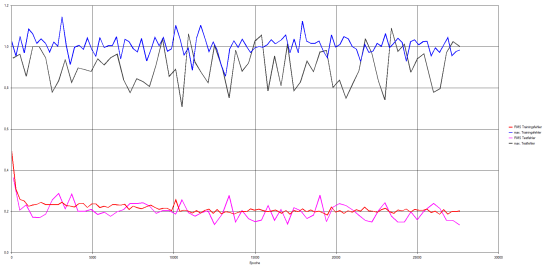


(a) Learning Curve

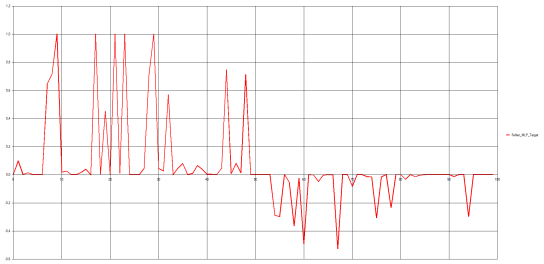


(b) Error of the Train Data

Fig. G.9: PowerVcon10F Variant 5 (10 linear -10 sig - 10 tanh - 1 linear)



(a) Learning Curve

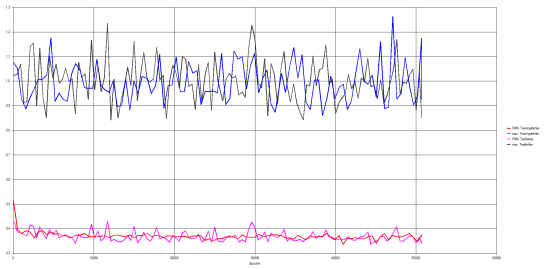


(b) Error of the Train Data

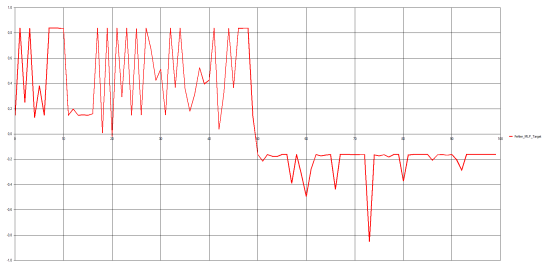
Fig. G.10: PowerVcon10F Variant 6 (10 linear -10 tanh - 10 tanh - 1 linear)

After Pruning

One Hidden layer:

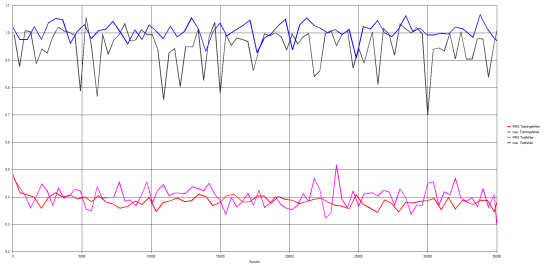


(a) Learning Curve

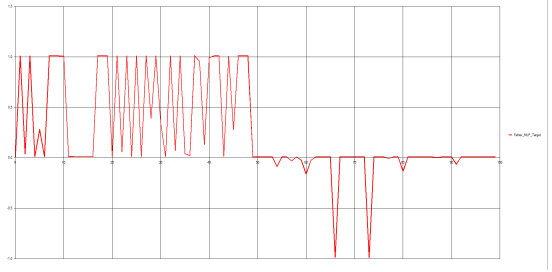


(b) Error of the Train Data

Fig. G.11: PowerVcon10F Variant a (10 linear - 2 sigmoid - 1 linear)



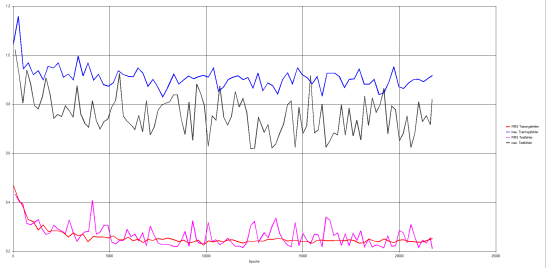
(a) Learning Curve



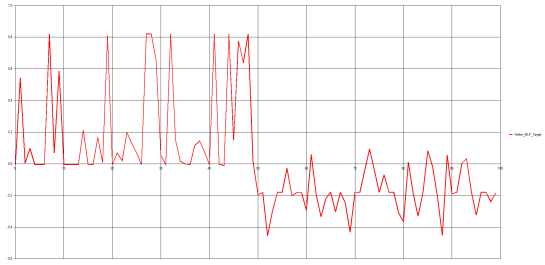
(b) Error of the Train Data

Fig. G.12: PowerVcon10F Variant b (10 linear - 1 tanh - 1 linear)

Two Hidden layer:



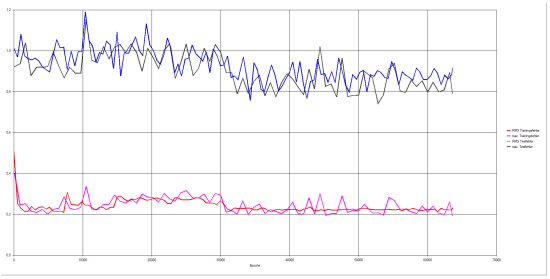
(a) Learning Curve



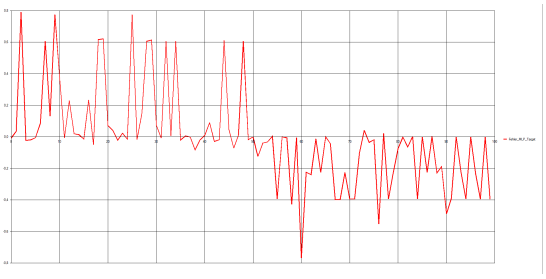
(b) Error of the Train Data

Fig. G.13: PowerVcon10F Variant c (10 linear - 3 sigmoid- 2 sigmoid - 1 linear)

PowerVcon10F Variant c-2 (10 linear - 3 sigmoid- 2 sigmoid - 1 linear) can be found as Fig. 5.2a and Fig. 5.2b in the main part in the section 5.2.

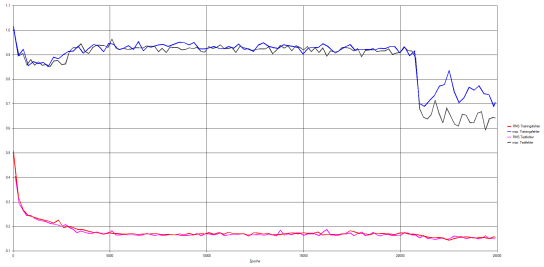


(a) Learning Curve

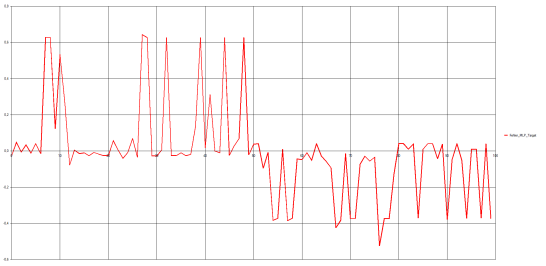


(b) Error of the Train Data

Fig. G.14: PowerVcon10F Variant d (10 linear - 4 tanh- 2 sigmoid - 1 linear)



(a) Learning Curve



(b) Error of the Train Data

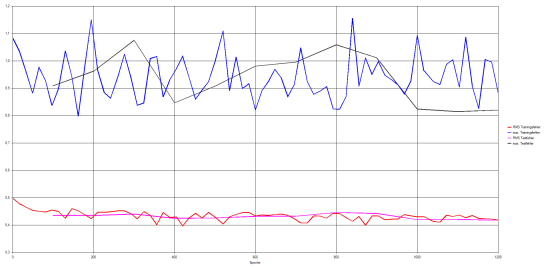
Fig. G.15: PowerVcon10F Variant d-2 (10 linear - 4 tanh - 2 sigmoid - 1 linear)

G.8.2 PowerVcon7F

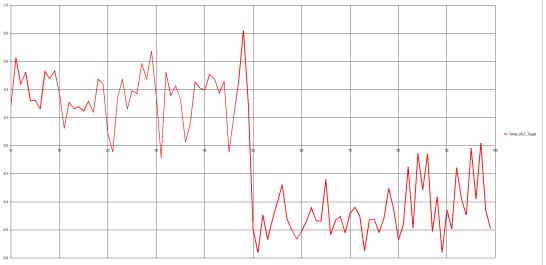
Dataset of 100 rows with voltage contingency

Pruning

One Hidden layer:

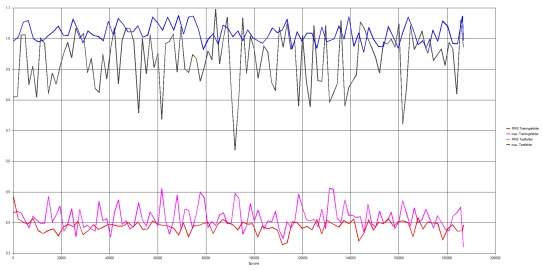


(a) Learning Curve

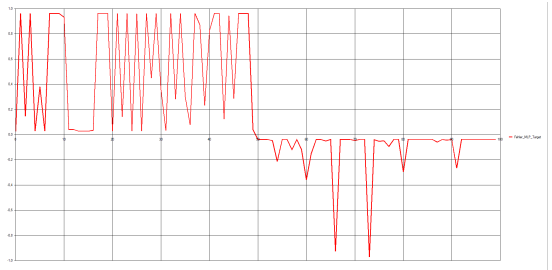


(b) Error of the Train Data

Fig. G.16: PowerVcon7F Variant 1(7 linear -15 sigmoid -1 linear)



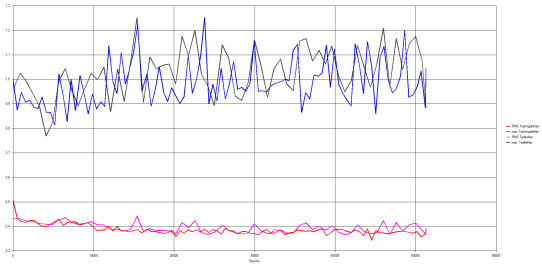
(a) Learning Curve



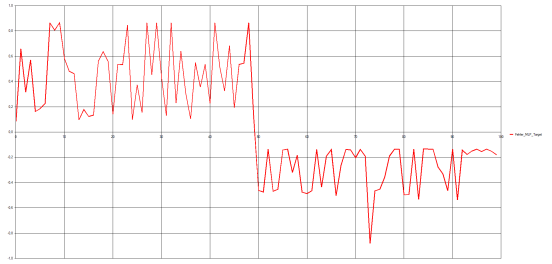
(b) Error of the Train Data

Fig. G.17: PowerVcon7F Variant 2 (7 linear -15 tanh -1 linear)

Two Hidden layer:

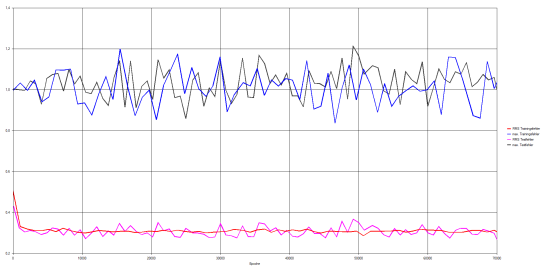


(a) Learning Curve

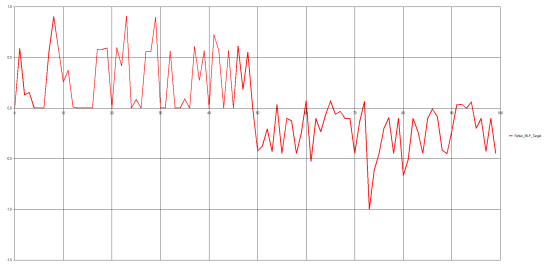


(b) Error of the Train Data

Fig. G.18: PowerVcon7F Variant 3 (7 linear -10 sig - 10 sig - 1 linear)

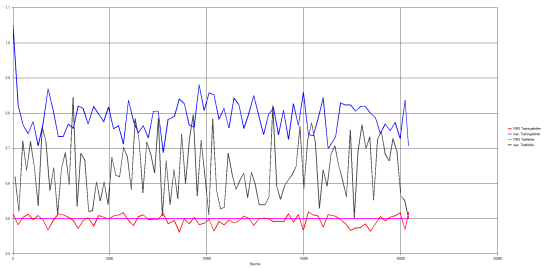


(a) Learning Curve

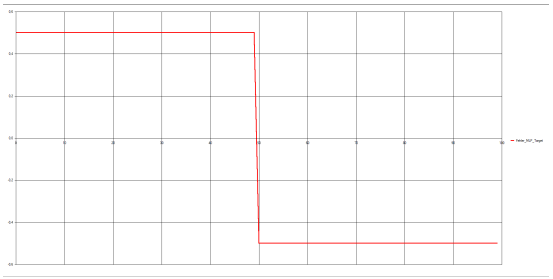


(b) Error of the Train Data

Fig. G.19: PowerVcon7F Variant 4 (7 linear -10 tanh - 10 sig - 1 linear)

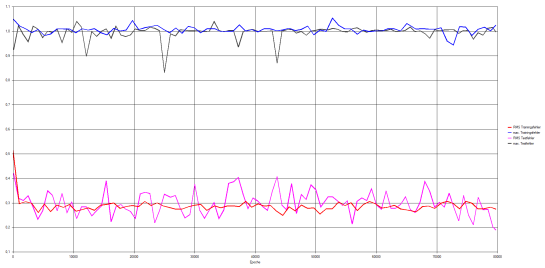


(a) Learning Curve



(b) Error of the Train Data

Fig. G.20: PowerVcon7F Variant 5 (7 linear -10 sig - 10 tanh - 1 linear)



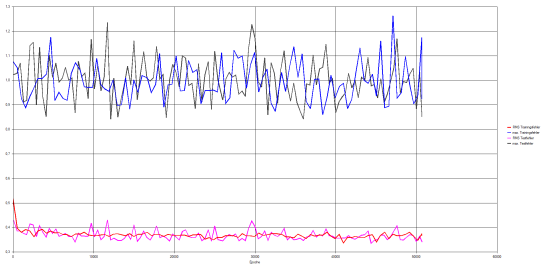
(a) Learning Curve

(b) Error of the Train Data

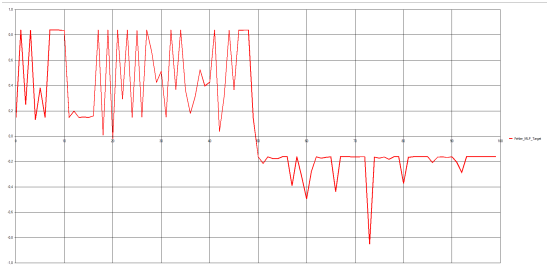
Fig. G.21: PowerVcon7F Variant 6 (7 linear -10 tanh - 10 tanh - 1 linear)

After Pruning

One Hidden layer:

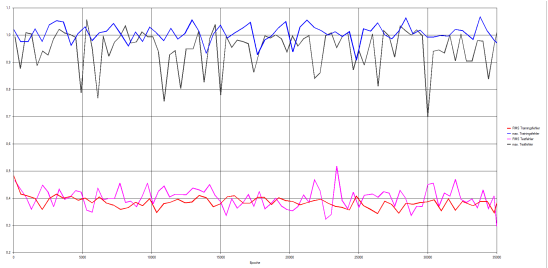


(a) Learning Curve

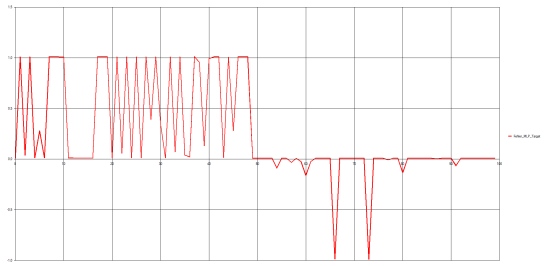


(b) Error of the Train Data

Fig. G.22: PowerVcon7F Variant a (7 linear - 2 sigmoid - 1 linear)



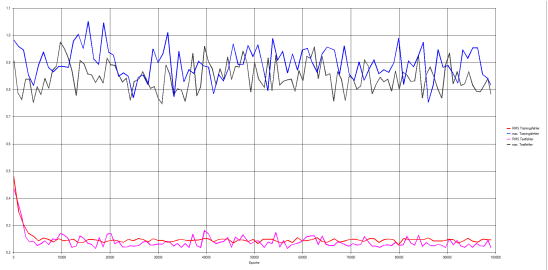
(a) Learning Curve



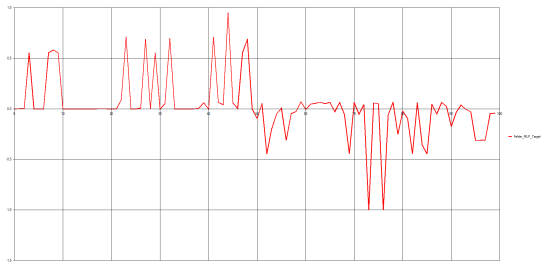
(b) Error of the Train Data

Fig. G.23: PowerVcon7F Variant b (7 linear - 1 tanh - 1 linear)

Two Hidden layer:

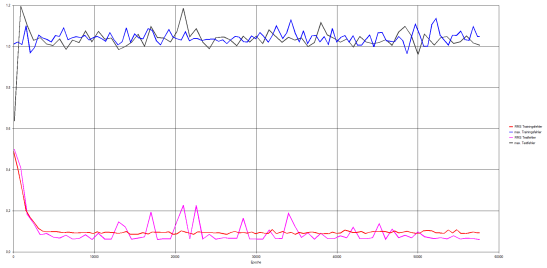


(a) Learning Curve

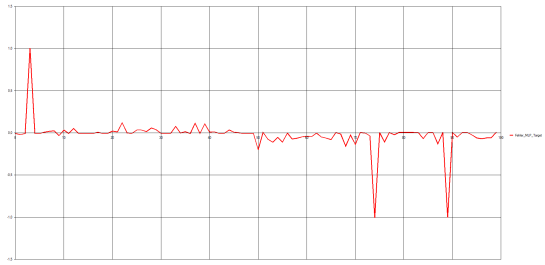


(b) Error of the Train Data

Fig. G.24: PowerVcon7F Variant c (7 linear - 3 sigmoid - 2 sigmoid - 1 linear)

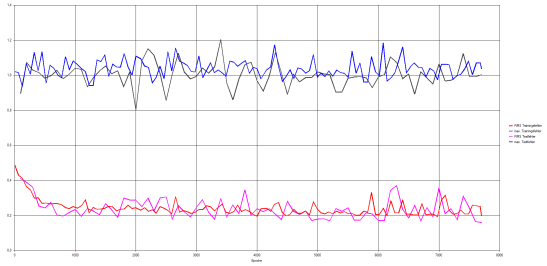


(a) Learning Curve

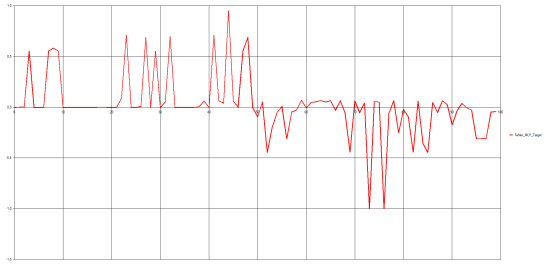


(b) Error of the Train Data

Fig. G.25: PowerVcon7F Variant d (7 linear - 5 sigmoid - 2 tanh - 1 linear)



(a) Learning Curve



(b) Error of the Train Data

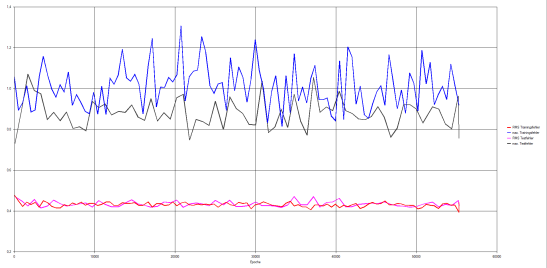
Fig. G.26: PowerVcon7F Variant e (7 linear - 3 tanh - 2 tanh - 1 linear)

G.8.3 Power10F

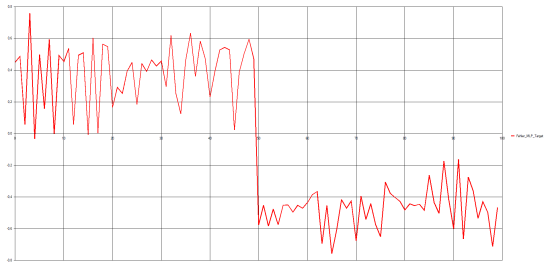
Dataset of 100 rows without voltage contingency

Pruning

One Hidden layer:

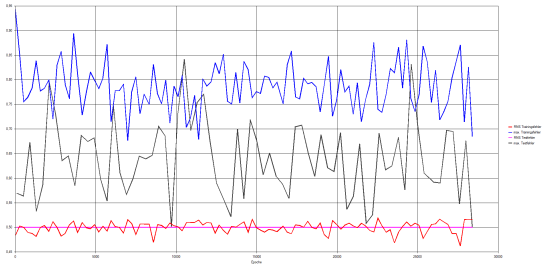


(a) Learning Curve

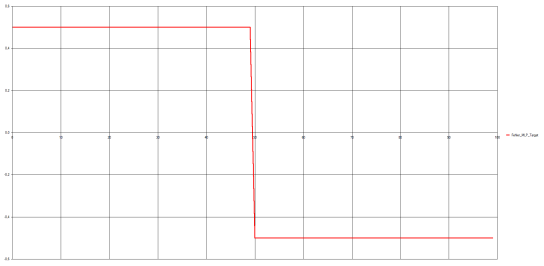


(b) Error of the Train Data

Fig. G.27: Power10F Variant 1 (10 linear -10 sigmoid -1 linear)



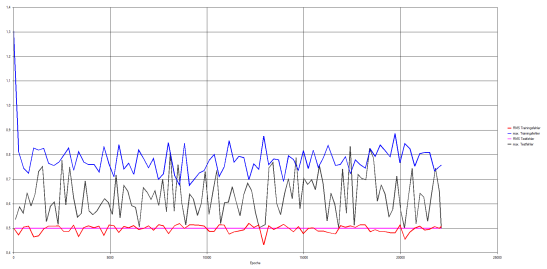
(a) Learning Curve



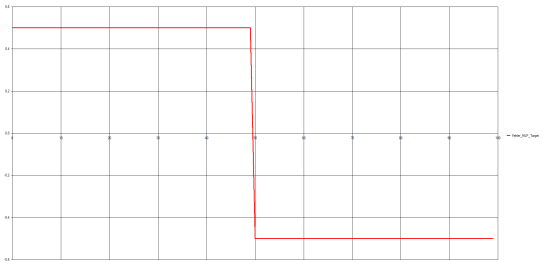
(b) Error of the Train Data

Fig. G.28: Power10F Variant 2 (10 linear -10 tanh -1 linear)

Two Hidden layer:

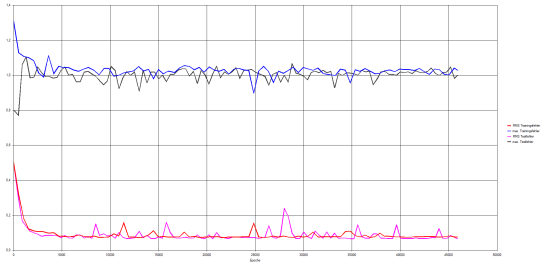


(a) Learning Curve

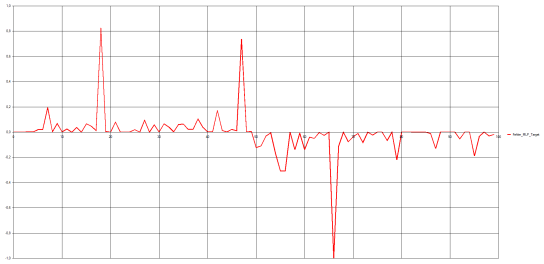


(b) Error of the Train Data

Fig. G.29: Power10F Variant 3 (10 linear -10 sig - 10 sig - 1 linear)

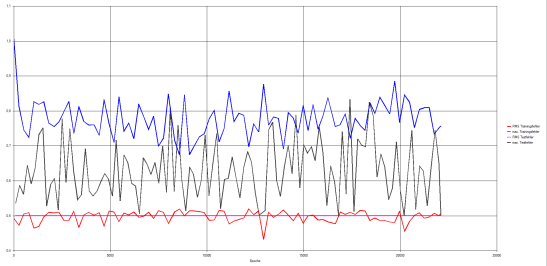


(a) Learning Curve

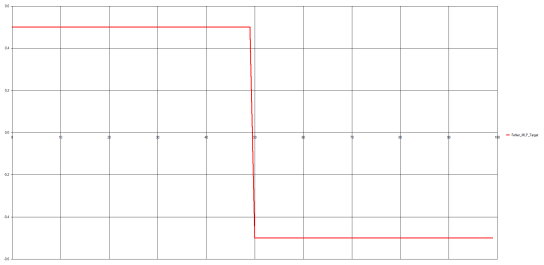


(b) Error of the Train Data

Fig. G.30: Power10F Variant 4 (10 linear -10 tanh - 10 sig - 1 linear)

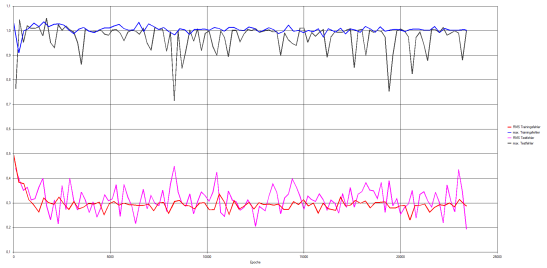


(a) Learning Curve



(b) Error of the Train Data

Fig. G.31: Power10F Variant 5 (10 linear -10 sig - 10 tanh - 1 linear)



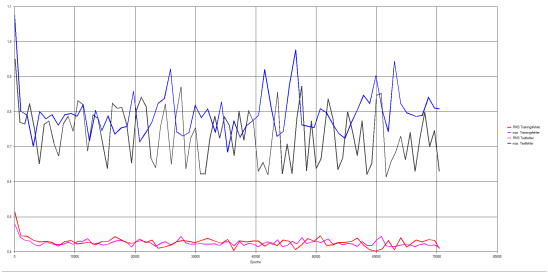
(a) Learning Curve

(b) Error of the Train Data

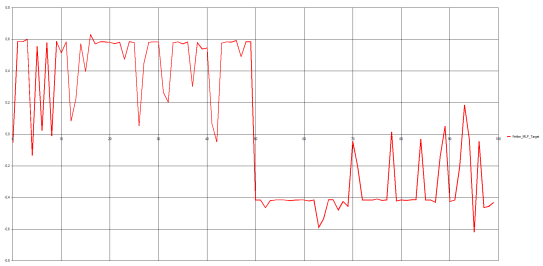
Fig. G.32: Power10F Variant 6 (10 linear -10 tanh - 10 tanh - 1 linear)

After Pruning

One Hidden layer:



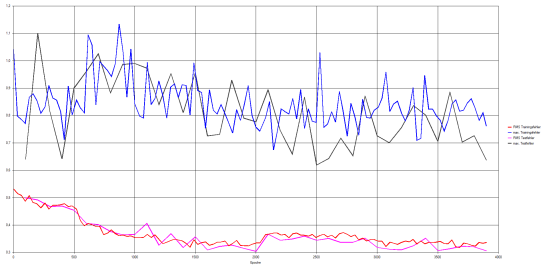
(a) Learning Curve



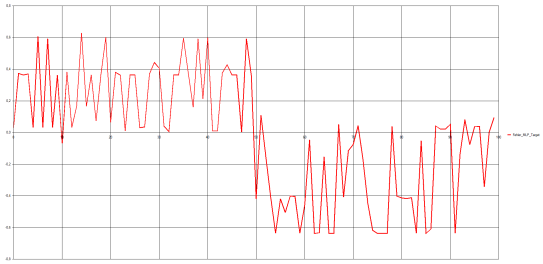
(b) Error of the Train Data

Fig. G.33: Power10F Variant a (10 linear - 2 sigmoid - 1 linear)

Two Hidden layer:

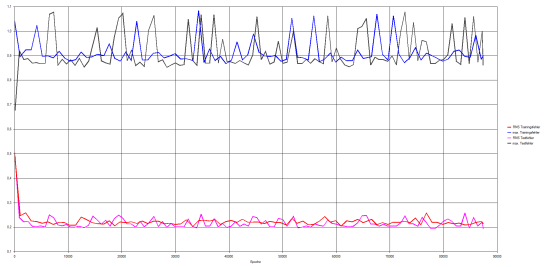


(a) Learning Curve

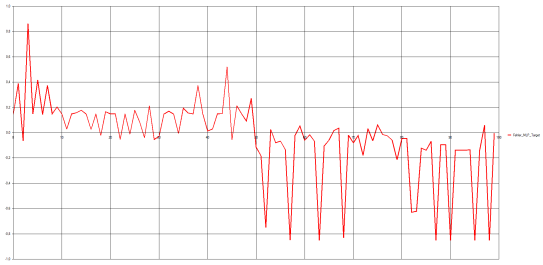


(b) Error of the Train Data

Fig. G.34: Power10F Variant b (10 linear - 3 tanh- 3 sigmoid - 1 linear)



(a) Learning Curve



(b) Error of the Train Data

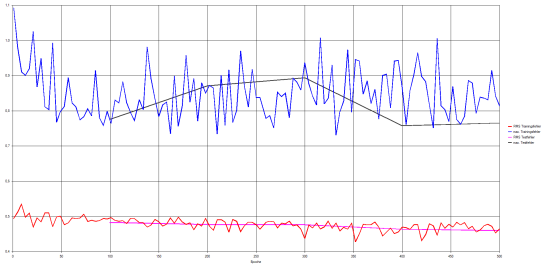
Fig. G.35: Power10F Variant b-2 (10 linear - 3 tanh- 3 sigmoid - 1 linear)

G.8.4 Power7F

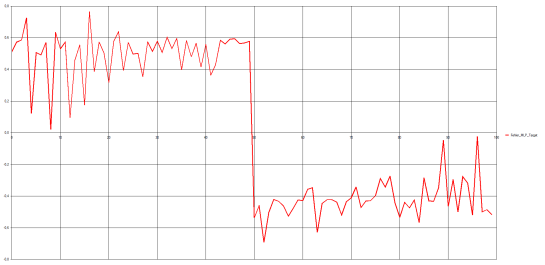
Dataset of 100 rows without voltage contingency

Pruning

One Hidden layer:

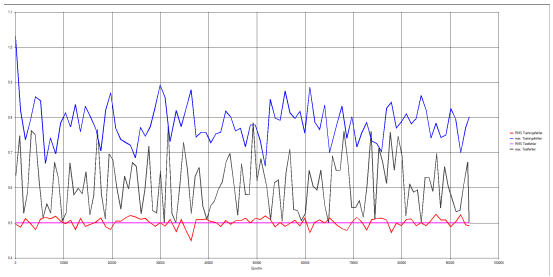


(a) Learning Curve

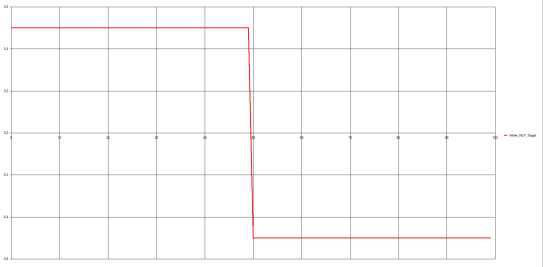


(b) Error of the Train Data

Fig. G.36: Power7F Variant 1(7 linear -10 sigmoid -1 linear)



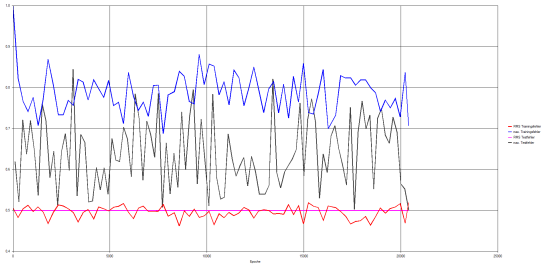
(a) Learning Curve



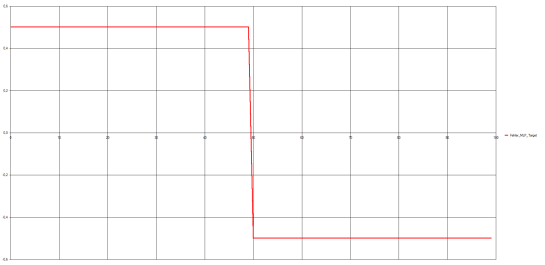
(b) Error of the Train Data

Fig. G.37: Power7F Variant 2 (7 linear -10 tanh -1 linear)

Two Hidden layer:

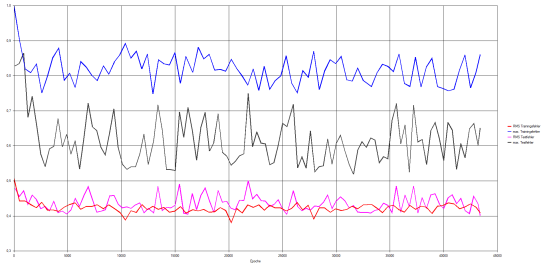


(a) Learning Curve

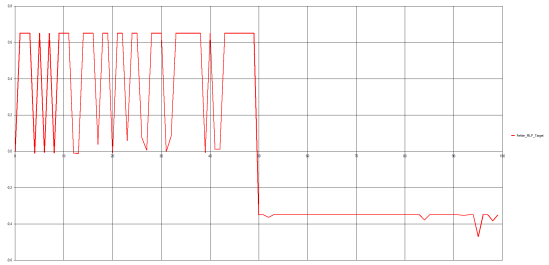


(b) Error of the Train Data

Fig. G.38: Power7F Variant 3 (7 linear -10 sig - 10 sig - 1 linear)

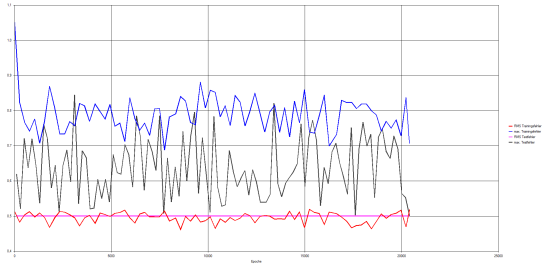


(a) Learning Curve

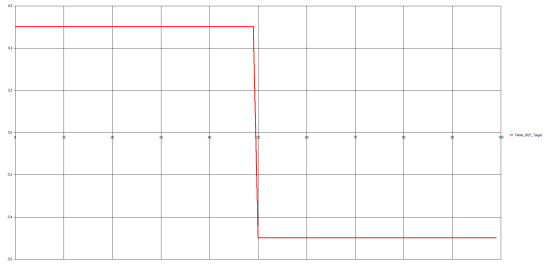


(b) Error of the Train Data

Fig. G.39: Power7F Variant 4 (7 linear -10 tanh - 10 sig - 1 linear)

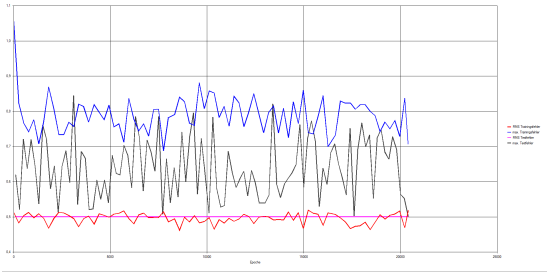


(a) Learning Curve

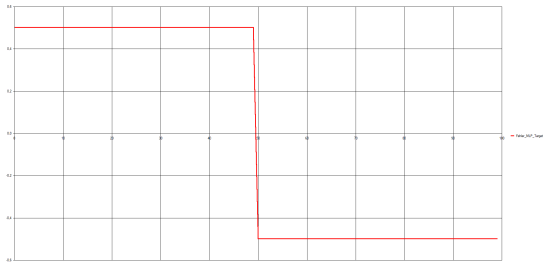


(b) Error of the Train Data

Fig. G.40: Power7F Variant 5 (7 linear -10 sig - 10 tanh - 1 linear)



(a) Learning Curve

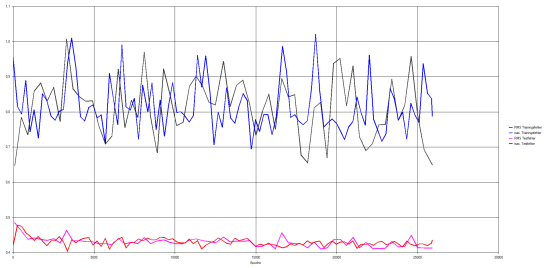


(b) Error of the Train Data

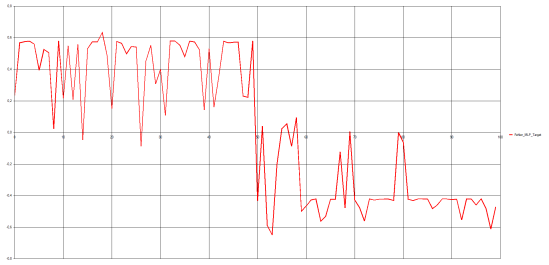
Fig. G.41: Power7F Variant 6 (7 linear -10 tanh - 10 tanh - 1 linear)

After Pruning

One Hidden layer:



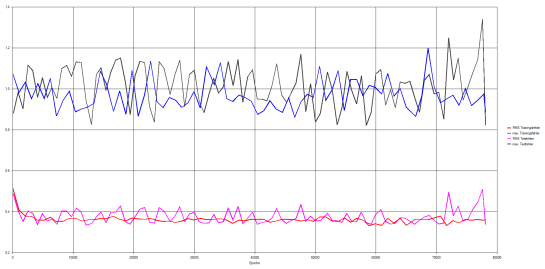
(a) Learning Curve



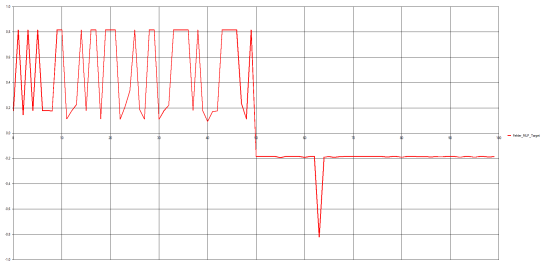
(b) Error of the Train Data

Fig. G.42: PowerVcon7F Variant a (7 linear - 2 sigmoid - 1 linear)

Two Hidden layer:



(a) Learning Curve



(b) Error of the Train Data

Fig. G.43: PowerVcon7F Variant b (7 linear - 2 tanh- 2 sigmoid - 1 linear)

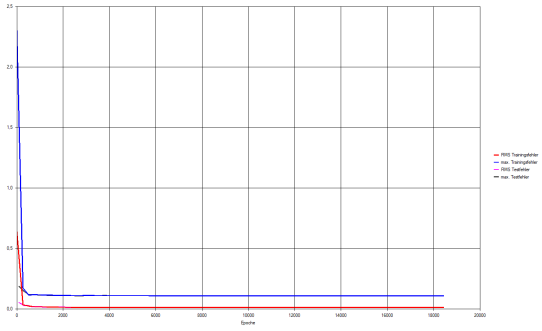
G.9 PQtoV

Tab. G.8: Examination of the Architecture for PQtoV (Pruning)

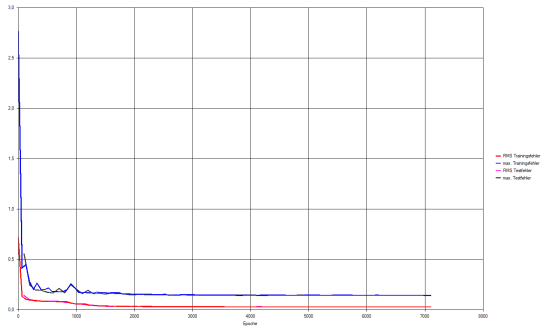
variant	M_1	$f(x)$ **from** M_1	M_2	$f(x)$ **from** M_2	n_c	n_c **after pruning**	E_{max}	**RMSE test**
1	20	sigmoid	20	sigmoid	800	350	0.1086	0.0116
2	20	tanh	20	sigmoid	800	201	0.1408	0.0275
3	20	sigmoid	20	tanh	800	258	0.0318	0.0087
4	20	tanh	20	tanh	800	741	3.1661	0.8934

Tab. G.9: Examination of the Architecture for PQtoV

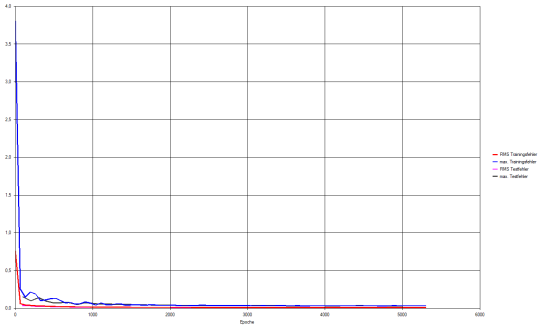
variant	M_1	$f(x)$ **from** M_1	M_2	$f(x)$ **from** M_2	n_c	ε_{max}	**RMSE test**
3	10	sigmoid	10	tanh	300	0.1012	0.0292



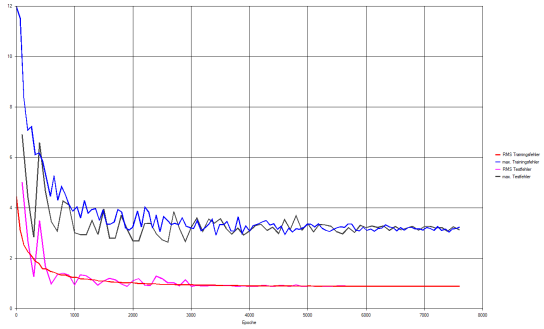
(a) PQtoV Variant 2(10 linear -20 tanh-20 tanh -10 linear)



(b) PQtoV Variant 2(10 linear -20 tanh-20 tanh -10 linear)

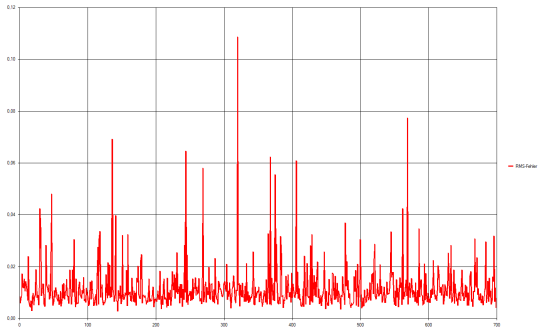


(c) PQtoV Variant 3(10 linear -20 tanh-20 tanh -10 linear)

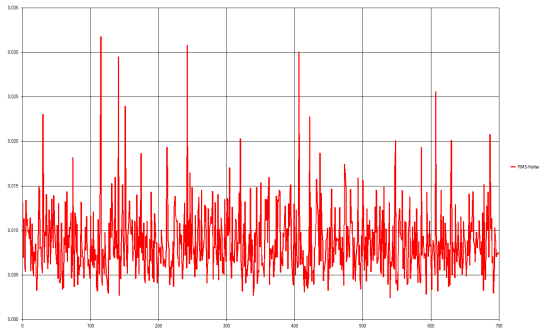


(d) PQtoV Variant 4(10 linear -20 tanh-20 tanh -10 linear)

Fig. G.44: Learning Curve for PQtoV Pruning

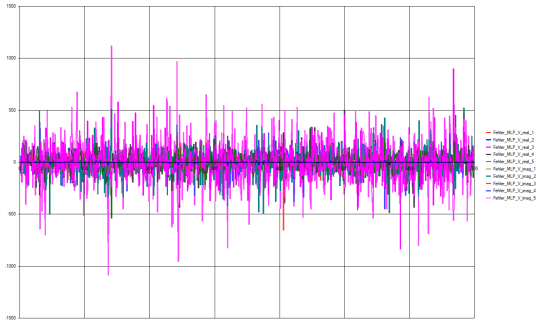


(a) PQtoV Variant 2(10 linear -20 tanh-20 tanh -10 linear)

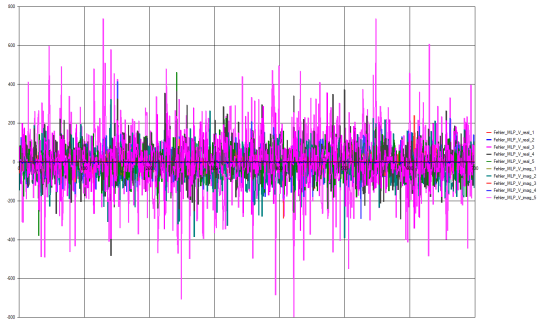


(b) PQtoV Variant 3(10 linear -20 tanh-20 tanh -10 linear)

Fig. G.45: RMSE for Train Data as Test



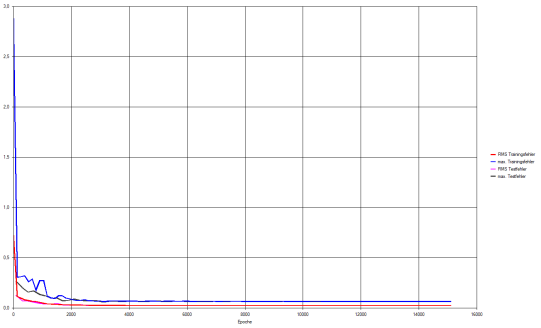
(a) PQtoV Variant 1(10 linear -20 sigmoid-20 sigmoid -10 linear)



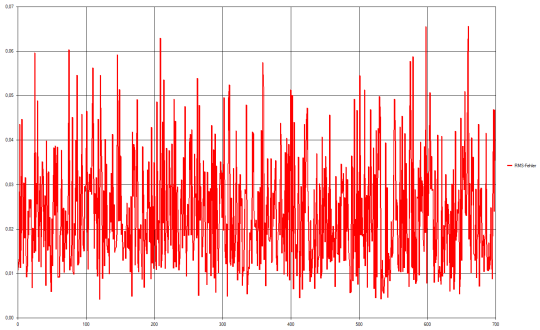
(b) PQtoV Variant 3(10 linear -20 tanh-20 tanh -10 linear)

Fig. G.46: Absolute Error for Train Data as Test

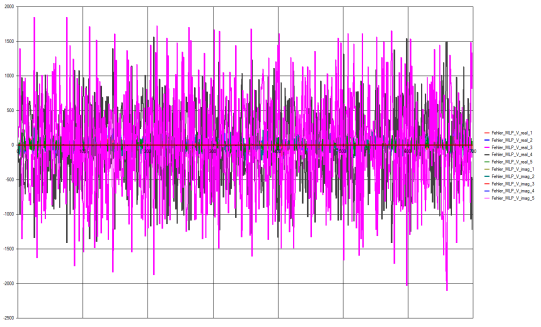
After Pruning



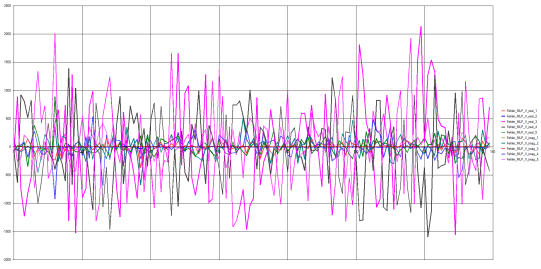
(a) Learning Curve



(b) RMSE for Train Data as Test



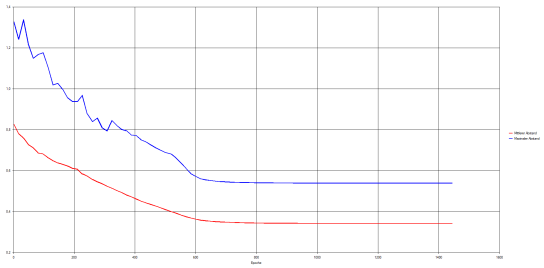
(c) Absolute Error for Train Data as Test



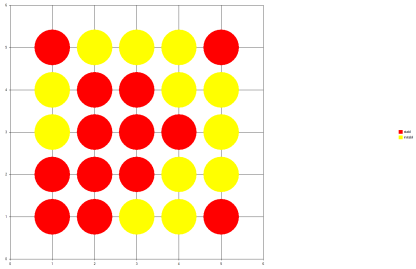
(d) Absolute error for the test

Fig. G.47: Variant 3(10 linear -10 sigmoid-10 tanh -10 linear)

G.10 Kohonen Network

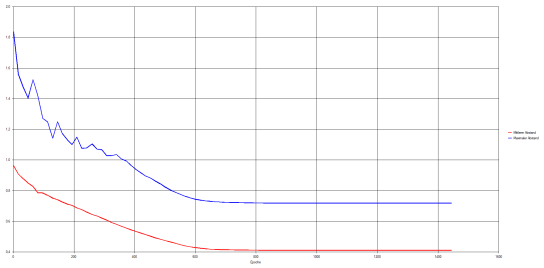


(a) Learning Curve

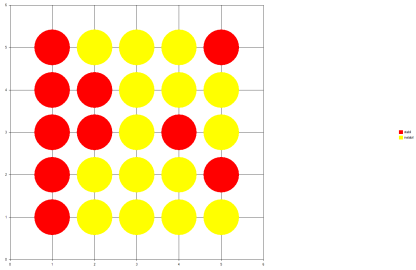


(b) Kohonenmap

Fig. G.48: Kohonen Network Variant 1

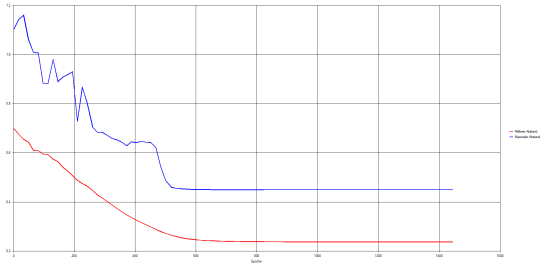


(a) Learning Curve

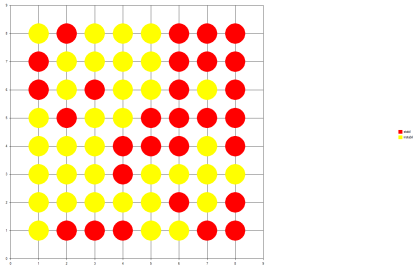


(b) Kohonenmap

Fig. G.49: Kohonen Network Variant 2

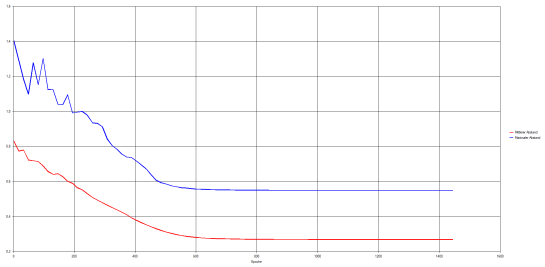


(a) Learning Curve

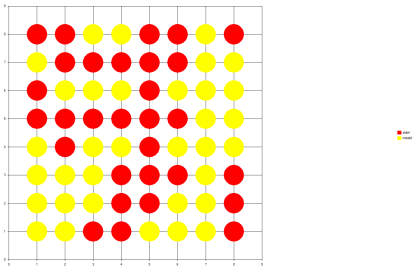


(b) Kohonenmap

Fig. G.50: Kohonen Network Variant 3

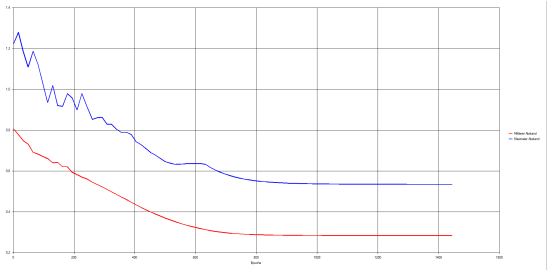


(a) Learning Curve

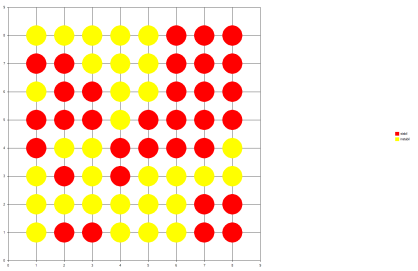


(b) Kohonenmap

Fig. G.51: Kohonen Network Variant 4

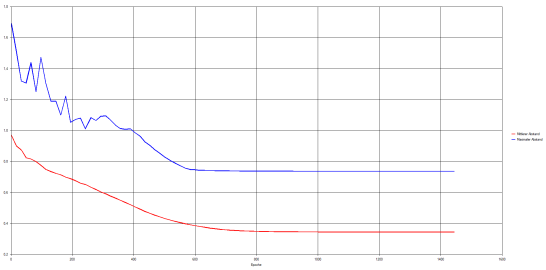


(a) Learning Curve

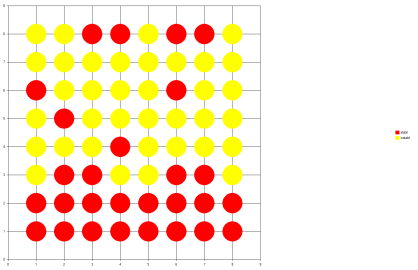


(b) Kohonenmap

Fig. G.52: Kohonen Network Variant 5

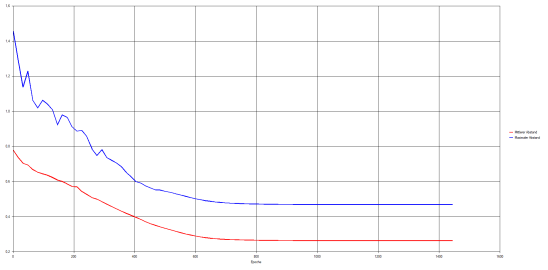


(a) Learning Curve

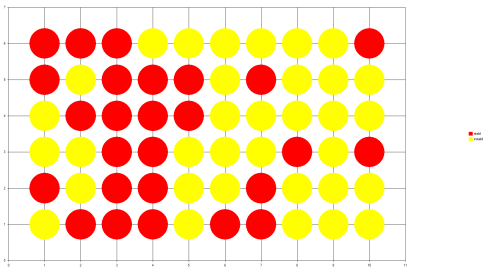


(b) Kohonenmap

Fig. G.53: Kohonen Network Variant 6



(a) Learning Curve



(b) Kohonenmap

Fig. G.54: Kohonen Network Variant 7
Kohonen Network Variant 8 can be found as Fig. 5.3a and Fig. 5.3b in the main part in the section 5.3.

G.11 Evaluation

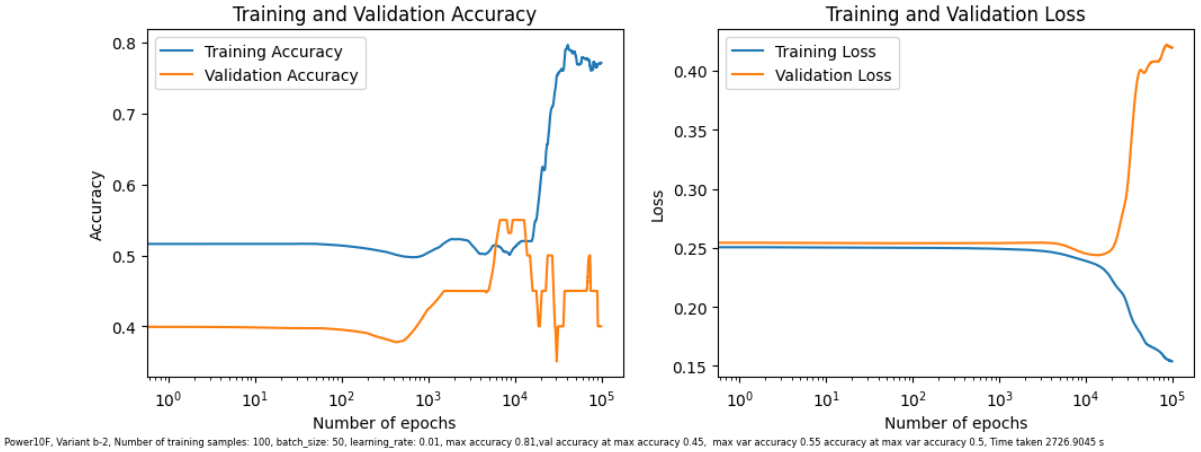


Fig. G.55: Accuracy and Loss Curve of Keras Training of Power10F Variant b-2

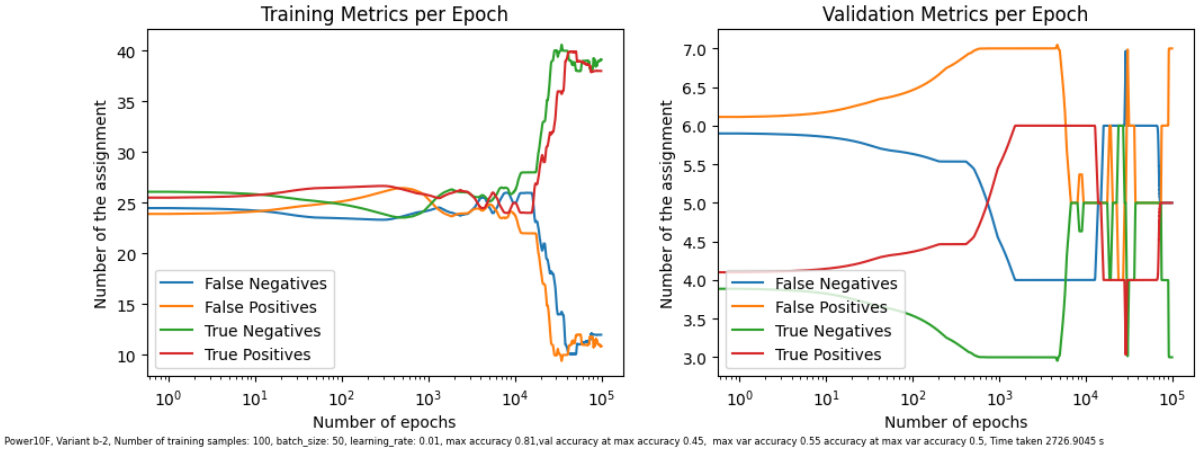


Fig. G.56: Confusion Matrix Progression During Keras Training of Power10F Variant b-2

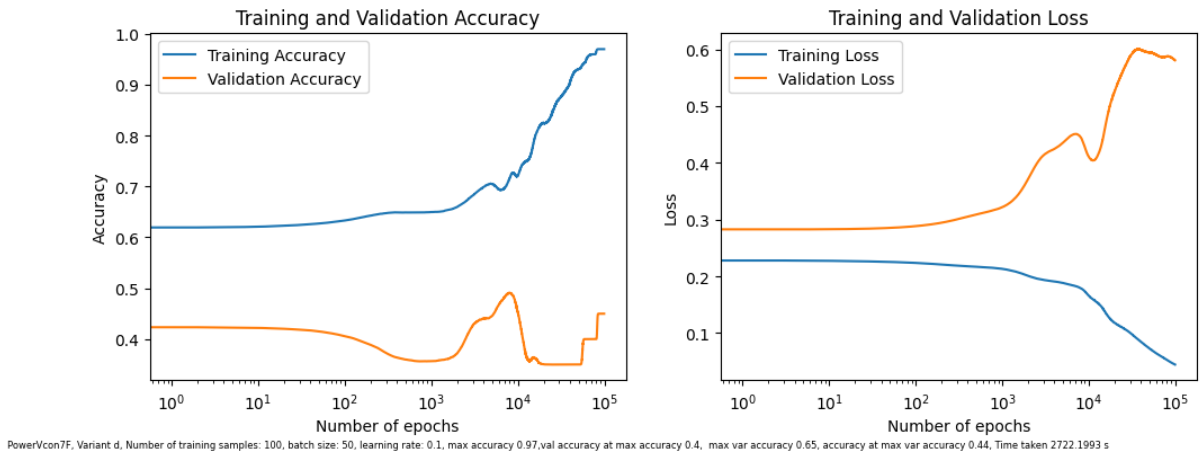


Fig. G.57: Accuracy and Loss Curve of Keras Training of PowerVcon7F Variant d

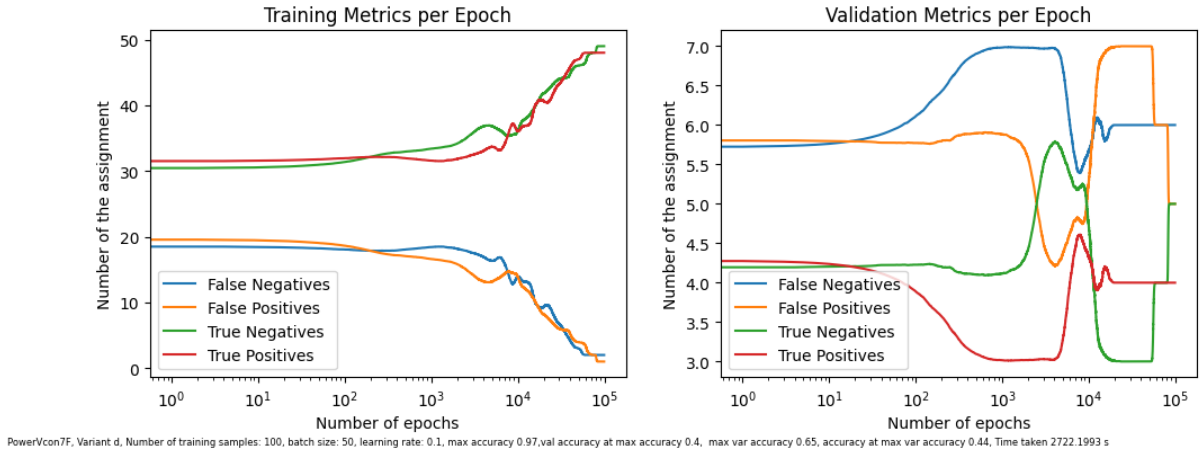


Fig. G.58: Confusion Matrix Progression During Keras Training of PowerVcon7F Variant d

PowerVcon10F Variant c with Keras can be found as Fig. 5.4 and Fig. 5.5 in the main part in the section 5.5.

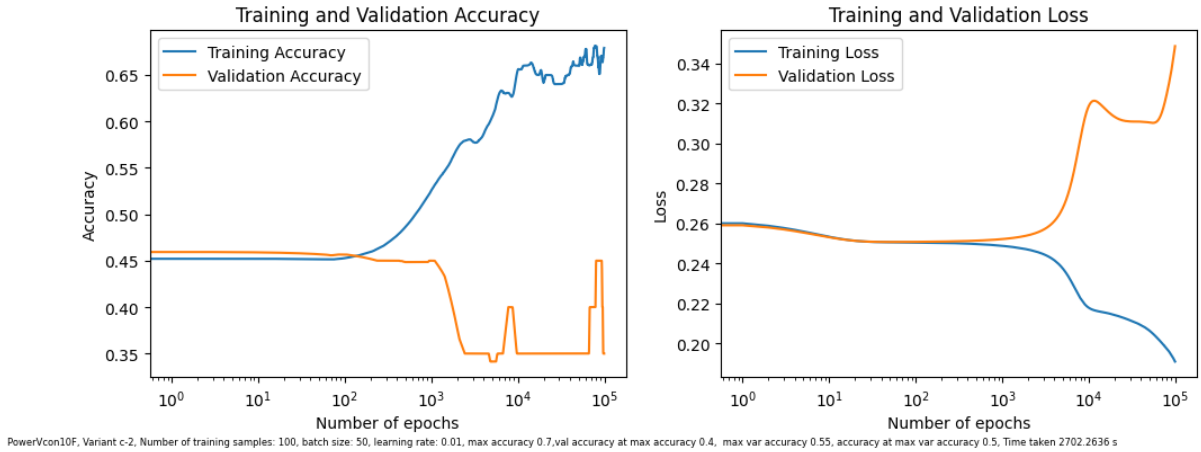


Fig. G.59: Accuracy and Loss Curve of Keras Training of PowerVcon10F Variant c-2

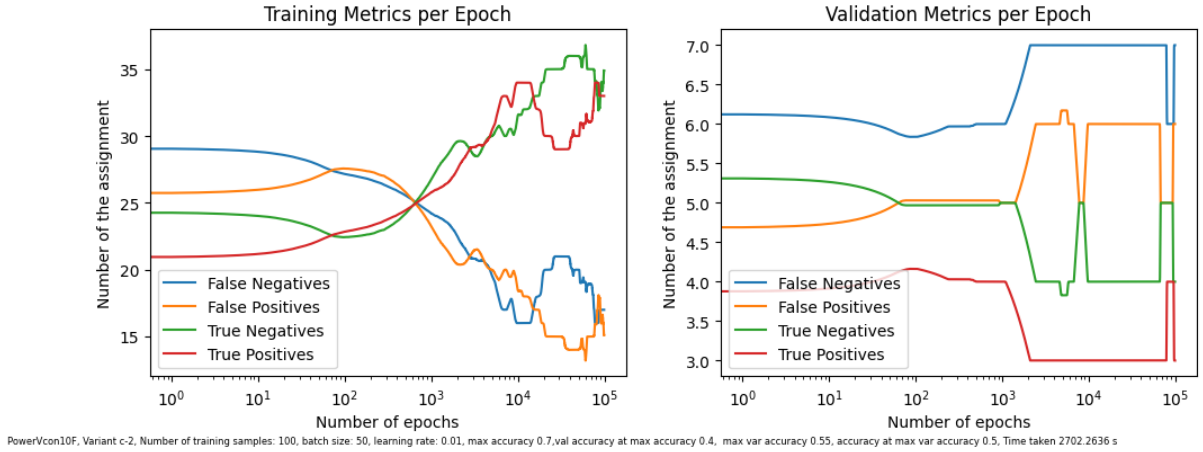


Fig. G.60: Confusion Matrix Progression During Keras Training of PowerVcon10F Variant c-2

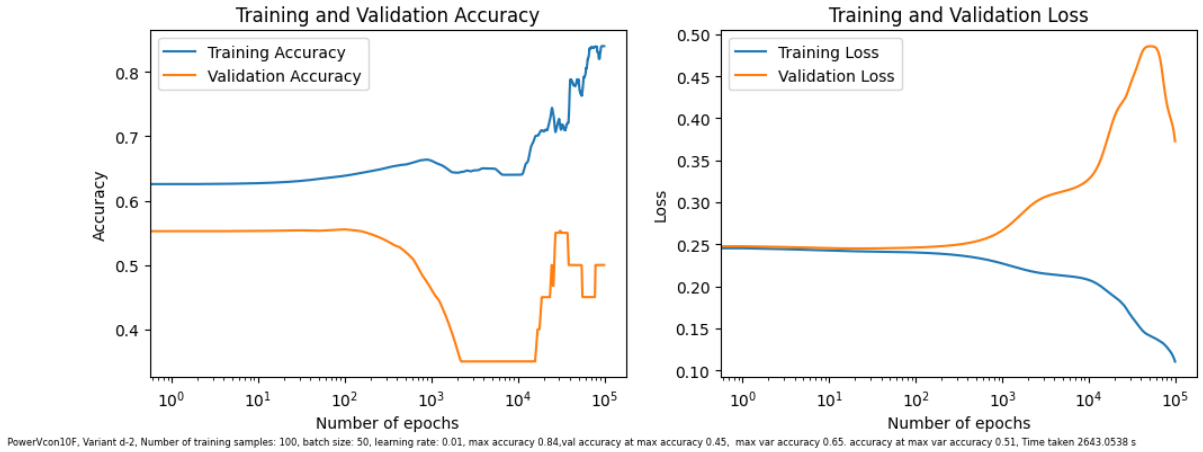


Fig. G.61: Accuracy and Loss Curve of Keras Training of PowerVcon10F Variant d-2

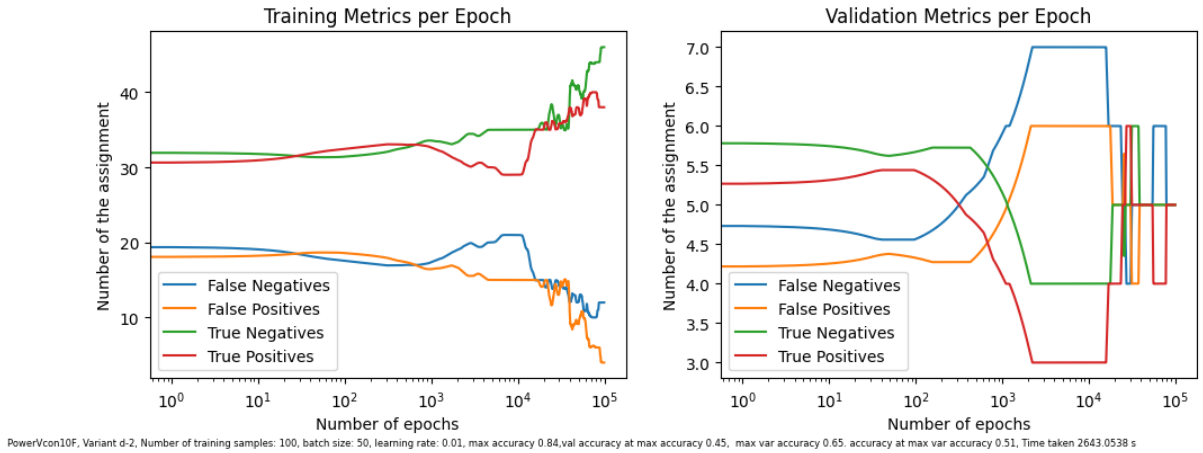


Fig. G.62: Confusion Matrix Progression During Keras Training of PowerVcon10F Variant d-2

G.12 Data Scale

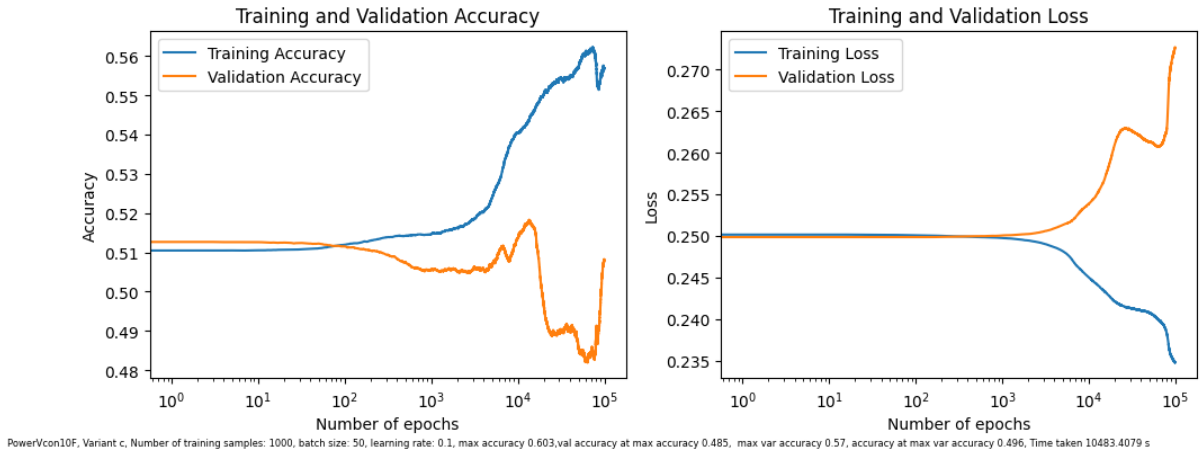


Fig. G.63: Accuracy and Loss Curve of Keras Training of Power10F Variant c with Batch Size 50, Dataset 1000

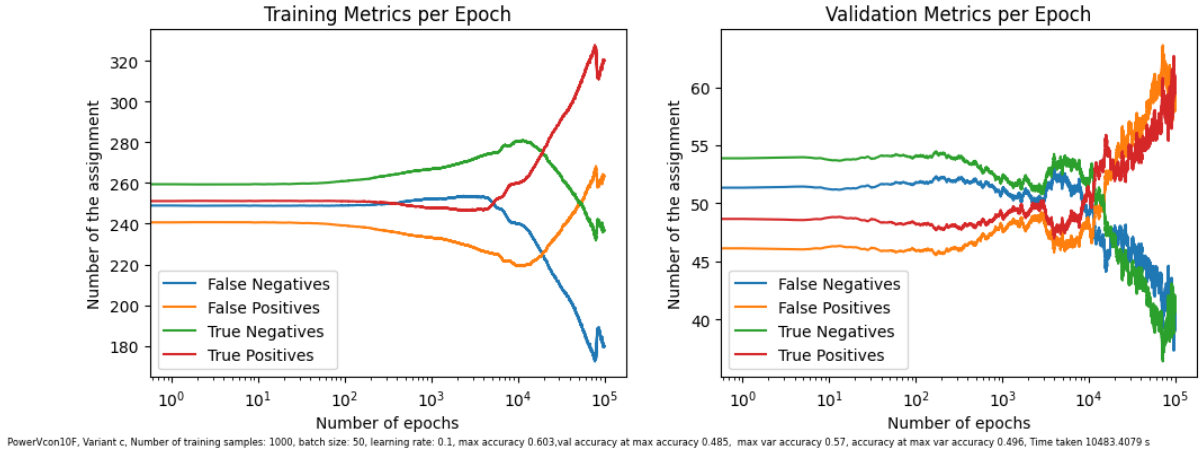


Fig. G.64: Confusion Matrix Progression During Keras Training of Power10F Variant c with Batch Size 100, Dataset 1000

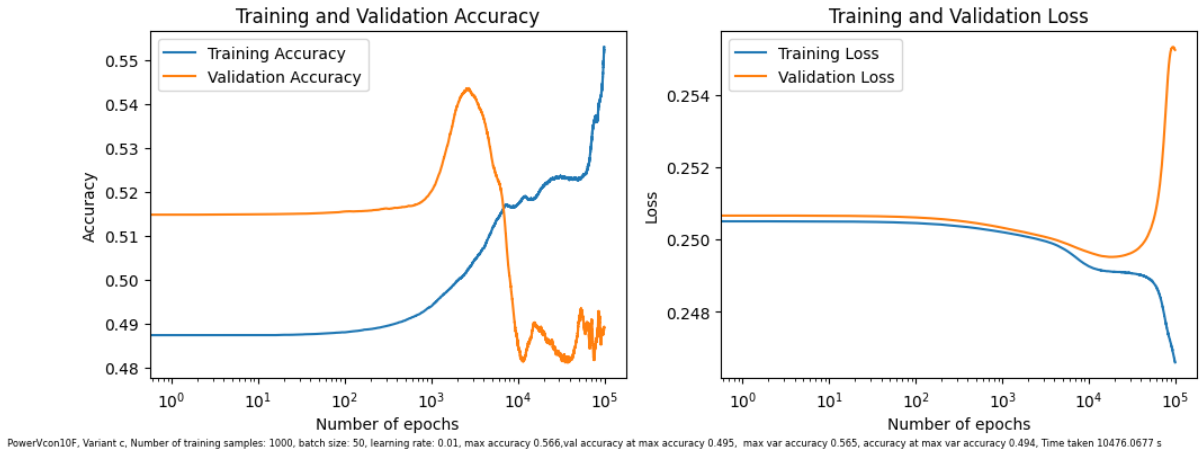


Fig. G.65: Accuracy and Loss Curve of Keras Training of Power10F Variant c with learningrate 0.01, Batch Size 50, Dataset 1000

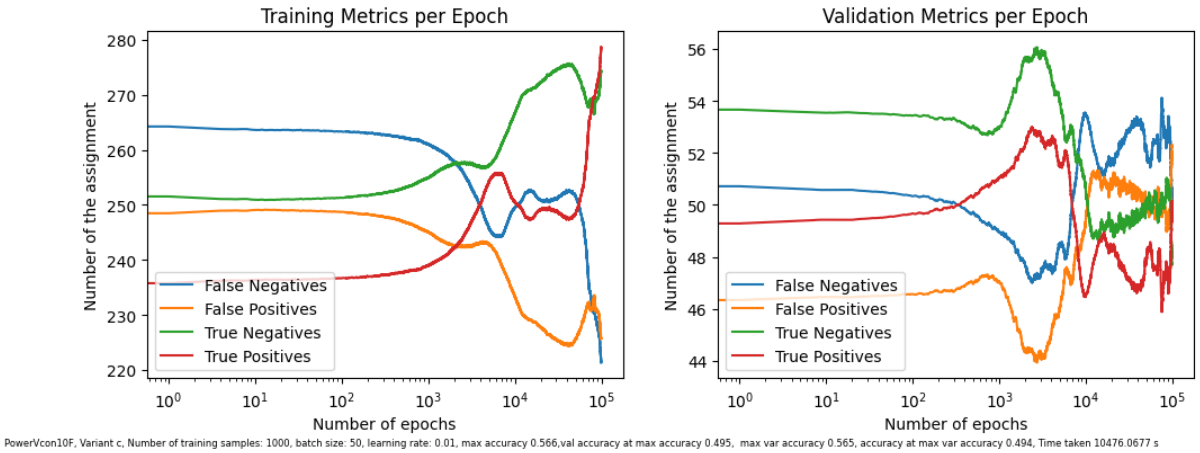


Fig. G.66: Confusion Matrix Progression During Keras Training of Power10F Variant c with learningrate 0.01, Batch Size 100, Dataset 1000

Power10F Variant c (with Batch Size 100, Dataset 1000) can be found as Fig. 6.1 and Fig. 6.2 in the main part in the section 6.1.

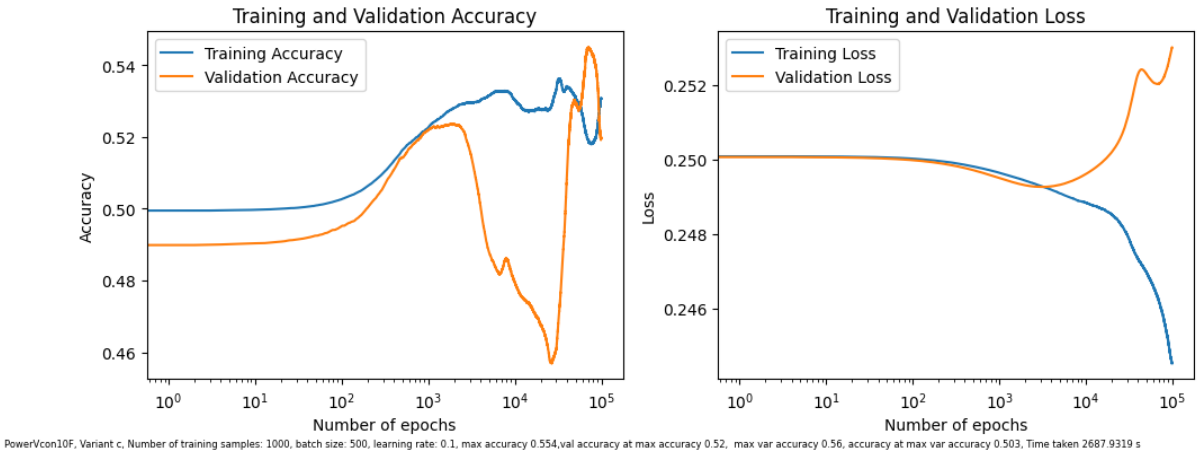


Fig. G.67: Accuracy and Loss Curve of Keras Training of Power10F Variant c with Batch Size 500, Dataset 1000

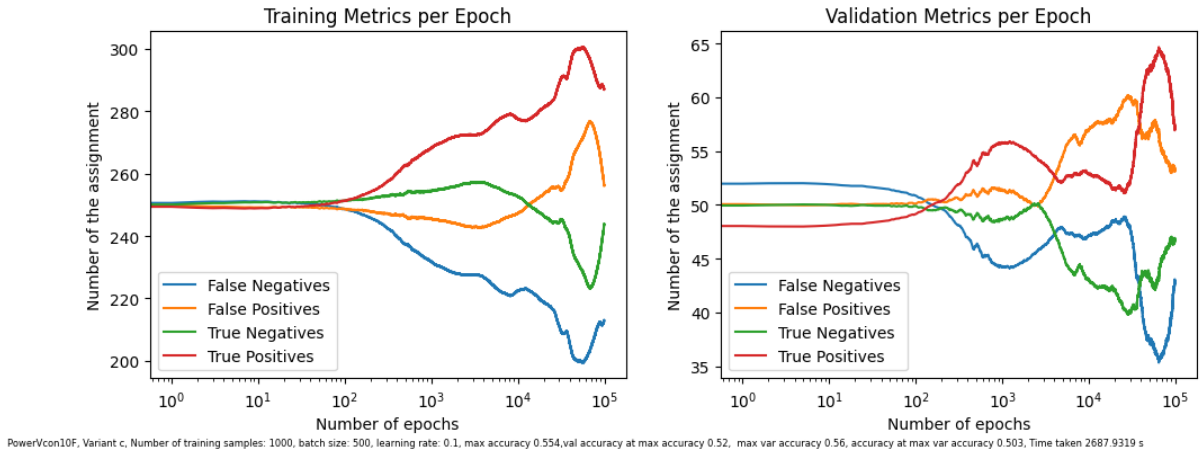


Fig. G.68: Confusion Matrix Progression During Keras Training of Power10F Variant c with Batch Size 500, Dataset 1000

The consideration of the scaling of the data 10000 and 100000. For computation time reasons not to 10^5 epochs.

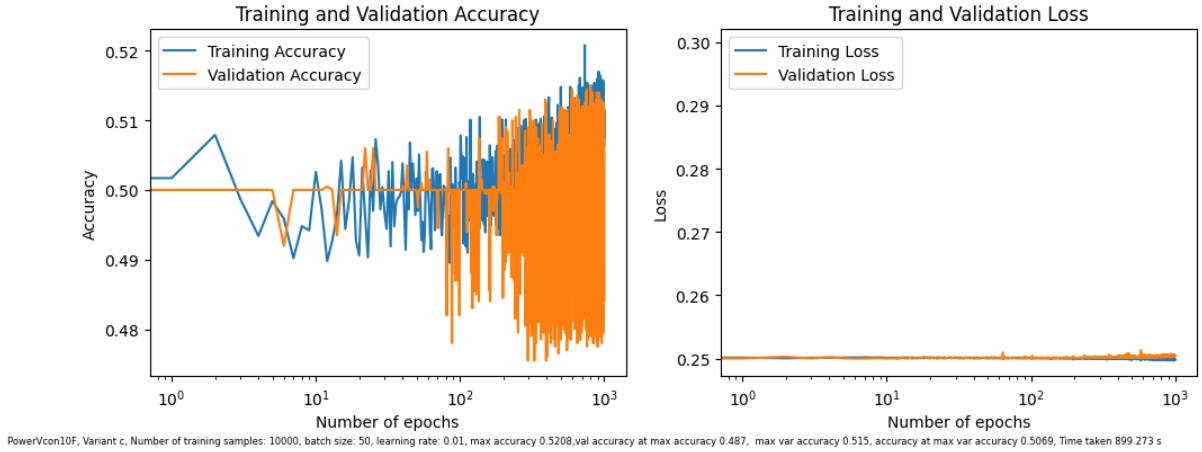


Fig. G.69: Accuracy and Loss Curve of Keras Training of Power10F Variant c with Batch Size 50, Leaning Rate 0.01, Dataset 10000

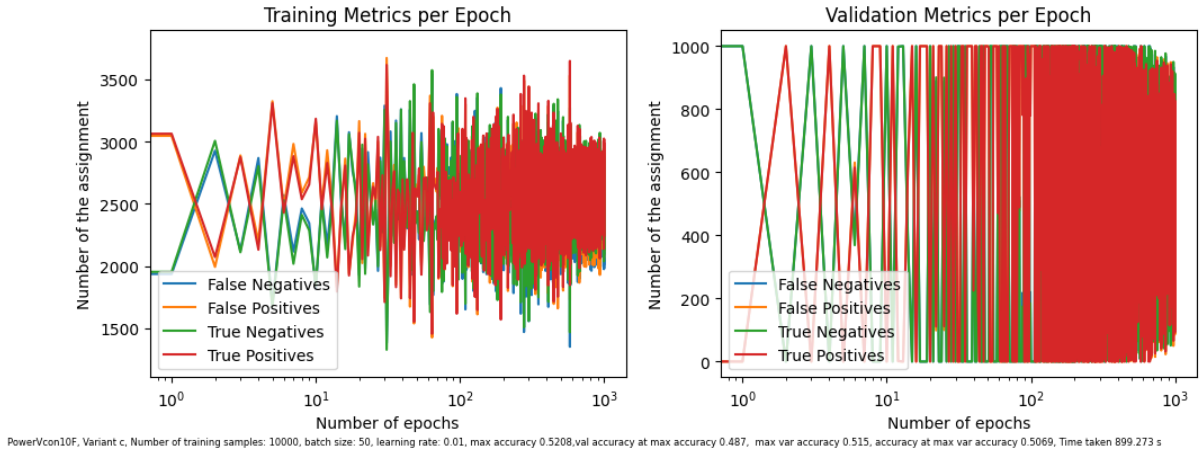


Fig. G.70: Confusion Matrix Progression During Keras Training of Power10F Variant c with Batch Size 50, Leaning Rate 0.01, Dataset 10000

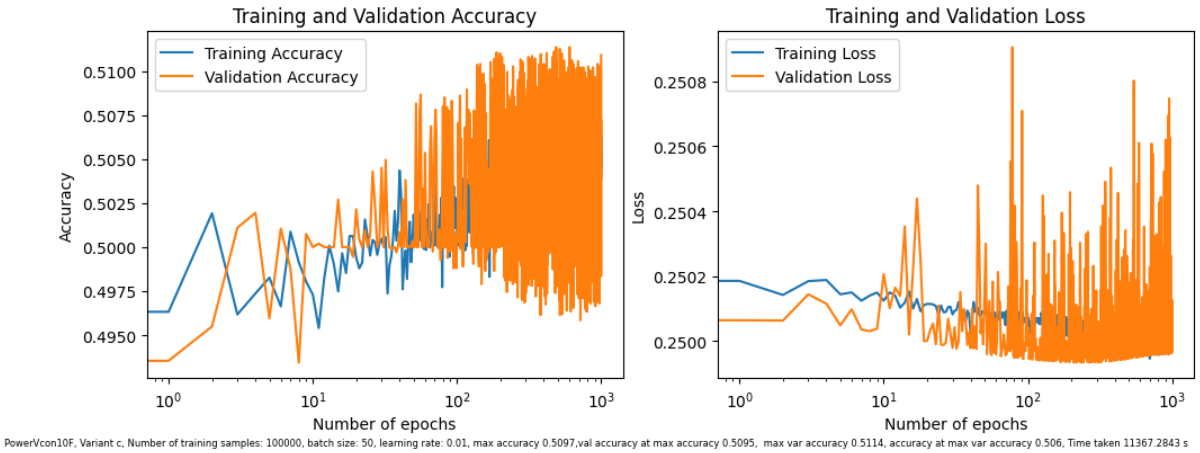


Fig. G.71: Accuracy and Loss Curve of Keras Training of Power10F Variant c with Batch Size 50, Leaning Rate 0.01, Dataset 100000

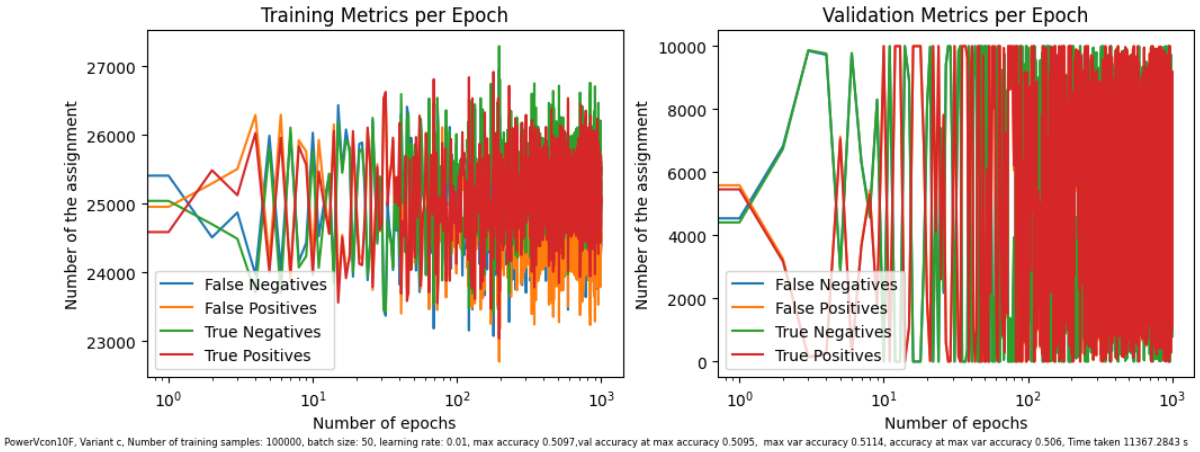


Fig. G.72: Confusion Matrix Progression During Keras Training of Power10F Variant c with Batch Size 50, Leaning Rate 0.01, Dataset 100000