A MASTER THESIS ON

Load Profile Forecasting of Manufacturing Processes with a Data-Driven Model

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

Diplom-Ingenieur
(Equivalent to Master of Science)

in

Embedded Systems (066 504)

by

Simon Howind

01607387

Supervisor(s):
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thilo Sauter
Projektass. Dipl.-Ing. Stefan Wilker, B.Eng.

Vienna, Austria
February 2024

Abstract

In order to mitigate climate change, countries all over the world invest in replacing fossil fuel power plants with electricity generation from renewable energy sources, whose availability is subject to changing weather conditions. Besides storing excess electricity generation in electrical storage systems, energy demand has to be adapted to the availability of renewable energy to maintain the balance between electricity generation and consumption in the electricity grid at all times. An incentive to do so are real-time energy tariffs. Manufacturing scheduling can take energy costs into account as an optimization criterion but depends on forecasts of the power profile of the individual manufacturing processes at the time of scheduling.

This thesis proposes a data-driven energy model architecture that generates a power profile from the process parameters. The architecture is aimed at discrete production, especially production types like batch production or job production, with processes of smaller quantities that run on adaptable machines with adaptable process parameters. Unlike model-driven approaches, the proposed data-driven energy model is easily adaptable to various use cases by training it with historical data, and unlike a simple lookup table, it can interpolate between parameter combinations from the historical data.

Different architectures were tested on synthetic energy data based on a real use case of battery pack assembly and on energy consumption data recorded in an experiment series conducted on an industrial robot. Of the tested model architectures, an Ensemble Long Short-Term Memory architecture and a Long Short-Term Memory-Sequence-to-Sequence architecture generally showed the best prediction accuracy while the Neural Network architecture proved to be unsuitable for the task. Altogether, an absolute prediction error of 5 % with the most suitable architectures in the respective cases can be expected.

Kurzfassung

Um den Klimawandel zu bekämpfen, investieren Länder auf der ganzen Welt in den Ersatz von fossilen Kraftwerken durch Stromerzeugung aus erneuerbaren Energiequellen, deren Verfügbarkeit von wechselnden Wetterbedingungen abhängt. Neben der Speicherung von überschüssiger Stromerzeugung in elektrischen Speichersystemen muss die Energienachfrage an die Verfügbarkeit erneuerbarer Energiequellen angepasst werden, um das Gleichgewicht zwischen Stromerzeugung und -verbrauch im Stromnetz jederzeit aufrechtzuerhalten. Ein Anreiz dafür sind Echtzeit-Energiepreise. Die Fertigungsplanung kann die Energiekosten als Optimierungskriterium berücksichtigen, ist jedoch auf Prognosen des Leistungsprofils der einzelnen Fertigungsprozesse zum Zeitpunkt der Planung angewiesen.

Diese Arbeit schlägt eine datengetriebene Energiemodellarchitektur vor, die ein Leistungsprofil aus den Prozessparametern generiert. Die Architektur zielt auf diskrete Produktion ab, insbesondere auf Produktionsarten wie Batch-Produktion oder Auftragsproduktion, bei denen Prozesse mit geringeren Mengen auf anpassbaren Maschinen mit anpassbaren Prozessparametern ablaufen. Im Gegensatz zu modellgetriebenen Ansätzen lässt sich das vorgeschlagene datengetriebene Energiemodell leicht an verschiedene Anwendungsfälle anpassen, indem es mit entsprechenden historischen Daten trainiert wird, und im Gegensatz zu einer einfachen Lookup-Tabelle kann es zwischen Parameterkombinationen aus den historischen Daten interpolieren.

Es wurden verschiedene Architekturen anhand synthetischer Energiemessdaten basierend auf einem realen Anwendungsfall der Batteriepackmontage und anhand von Energieverbrauchsdaten, die in einer Versuchsreihe an einem Industrieroboter aufgezeichnet wurden, getestet. Von den getesteten Modellarchitekturen zeigten eine Ensemble-Long-Short-Term-Memory-Architektur und eine Long-Short-Term-Memory-Sequence-to-Sequence-Architektur im Allgemeinen die beste Vorhersagegenauigkeit, während sich die Neural Network-Architektur als ungeeignet für die Aufgabe erwies. Insgesamt kann mit den jeweils geeignetsten Architekturen in den entsprechenden Fällen ein absoluter Vorhersagefehler von 5 % erwartet werden.

Preface

Parts of this work have been published in

- Simon Howind and Thilo Sauter. "Modeling Energy Consumption of Industrial Processes with Seq2Seq Machine Learning". In: *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*. IEEE, June 2023. DOI: `10.1109/isie51358.2023.10228118`.

Section II (State of the Art) from the paper is partly included in Chapter 2 of this thesis, and parts of Section III (Model Architecture) are included in Chapter 3. Simon Howind conducted all experiments and wrote the first version of the paper before Thilo Sauter wrote the abstract and made minor corrections and improvements.

Contents

List of Tables

List of Figures

Acronyms

ADAM	Adaptive Moment Estimation 52
AGV	Automated Guided Vehicle 43, 44
API	Application Programming Interface 25, 26
BIC	Battery Innovation Center 35, 36, 38, 42, 43, 44, 53
CEER	Council of European Energy Regulators 1
CNC	Computer Numerical Control 12, 13
CNN	Convolutional Neural Network 12
CO_2	Carbon Dioxide 5, 35
CSV	comma-separated values 36, 48, 49
DSM	Demand Side Management 2
ESS	Energy Storage System 2
F4R	Factories4Renewables 2, 3, 4
FFG	Österreichische Forschungsförderungsgesellschaft 4
FSM	Finite State Machine 8, 9, 14, 18
GPU	Graphics Processing Unit 52
GRU	Gated Recurrent Unit 22, 23, 24, 26, 27, 58, 62, 67, 68, 75, 77
HVAC	heating, ventilation, air conditioning 13
ICT	Institute of Computer Technology 2
IR	Industrial Robot 3, 46, 47, 50, 51, 53, 59, 61, 68, 70, 71, 75
k-nn	k-nearest-neighbors 12
LSTM	Long Short-Term Memory 11, 12, 13, 22, 23, 24, 26, 27, 28, 30, 31, 46, 51, 56, 57, 62, 63, 67, 68, 70, 71, 73, 75, 77

MAE mean absolute error 13

MSE mean square error 3, 34, 52, 55, 56, 57, 58, 62, 63, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 77

NaN Not a Number 43

NN Neural Network 12, 13, 20, 21, 26, 57, 58, 59, 67, 68, 73, 75, 77

PtP Point-to-Point 47, 48

RAD relative average distance 33, 70, 71, 72

ReLU rectified linear unit 26

RES Renewable Energy Sources 2, 5

RMSE root mean square error 12, 13

RNN Recurrent Neural Network 13, 20, 21, 22, 24, 26, 27, 30, 31, 56, 57, 58, 63, 67, 73, 75, 77

SAIDI System Average Interruption Duration Index 1

Seq2Seq sequence-to-sequence 20, 27, 28, 30, 56, 58, 62, 63, 67, 68, 70, 71, 73, 75, 77

SQL Structured Query Language 36

t-SNE t-distributed Stochastic Neighbor Embedding 54, 55

TU Technische Universität *(German for "University of Technology")* 2

USA United States of America 1

VSC Vienna Scientific Cluster 4, 52

Chapter 1

Introduction

This thesis explores the deduction of electrical load profiles from production schedules and process parameters in the manufacturing industry by means of a suitable model architecture. In this chapter, the motivation behind this work is presented. Subsequently, research questions are formulated, followed by the proposed approach to answer these questions. Finally, an outline of the structure of the thesis is given.

1.1 Motivation

Electricity in Europe nowadays is available almost always and everywhere. In international comparison, the reliability of electricity supply in Europe is very high. It is measured by the System Average Interruption Duration Index (SAIDI), an index to measure the average disruption time per customer (for details refer to [E-C22, p. 9]). Austria, for example, had a SAIDI of 44 minutes in 2018 [E-C19, p. 13]. This is approximately ten times less than the United States of America (USA)'s SAIDI of 470 minutes from 2017 [LH18]. European countries' [1] SAIDI values in 2016 ranged from approximately 20 minutes (Switzerland) to approximately 550 minutes (Romania), but only 2 out of 28 countries exceeded a SAIDI of 400 minutes [Eur18, pp. 20-22].

The high reliability and time-independent tariffs conceal the considerable efforts it takes to control the electricity grid. To keep the grid frequency stable, consumed and produced power must be balanced at all times. Traditionally, the flexibility to achieve this balance was located mainly on the generation side. Fossil fuels like coal and natural gas or uranium, with their high energy density, proved to be well-suited energy sources to react to long- and/or short-term fluctuations in electricity consumption independent of external influences. However, the decarbonization of the

[1] European countries in this context are the 28 countries included in the Council of European Energy Regulators (CEER) Benchmarking Report 6.1 [Eur18]

energy sector aiming to mitigate climate change and the nuclear phase-out of some countries with the goal of reducing the risk of a nuclear meltdown increase the importance of renewable energy sources like wind and solar energy to keep the grid frequency stable. Unlike fossil fuels, these energy sources are not immanent storages of energy, and due to their dependency on changing weather conditions, their generation is volatile. The lack of control in renewable electricity generation poses a challenge to keeping electricity generation and consumption aligned. One solution to address this challenge is buffering imbalances with Energy Storage Systems (ESSs). The search for suitable storage media and technologies for renewable energy is still a matter of ongoing research and prices for ESSs are expected to drop in the future while quality will increase [Far+22]. So far, however, this approach can be very costly, and limited existing storage capacity is a problem [ZS15].

To reduce the need for ESSs while still balancing electricity generation and consumption, measures on the demand side can be taken. Apart from reducing energy consumption overall, one measure is Demand Side Management (DSM), which “can be defined as the implementation of policies and measures to control, regulate, and reduce energy consumption” [Gud+12]. With DSM, some flexibility to keep the balance between power consumption and generation is shifted to the consumption side by consuming more energy at times of high Renewable Energy Sources (RES) electricity generation and consuming less energy at times of low RES electricity generation. With the increasing prevalence of volatile energy sources, the importance of DSM will likely grow over the next decades [Gud+12]. This also increases the likelihood of incentives to use DSM, for example in the form of time-dependent electricity tariffs with a smaller time granularity, as, for example, planned by the German Federal Ministry for Economic Affairs and Energy (German: Bundesministerium für Wirtschaft, BMWi) [BMWi2015, p. 69].

In this context, the project Factories4Renewables (F4R), conducted under the lead of the Institute of Computer Technology (ICT) at the TU Wien, investigates methods to adapt production schedules of factories to time-dependent electricity tariffs. Based on forecasts and statistical models, the most likely electricity price profile is selected, and the production schedule is optimized in such a way that the electricity cost is minimized, i.e., more energy is consumed at times of low electricity prices and vice versa.

A prerequisite for production schedule optimization under energy price and consumption criteria is the knowledge of the energy consumption down to the lowest level of production planning. Due to the huge variety of production machines, processes and the complex interdependencies between production processes with regard to their energy consumption, this proves to be a complex undertaking.

1.2 Research Questions

In this thesis, a model will be developed that predicts the energy consumption profile of industrial processes and production schedules based on process parameters and historical data. The following research question must be answered:

- What is the most suitable architecture with the smallest prediction error for an energy model for the context of manufacturing?

The following subquestions need to be answered as well:

- What is the deviation, i.e. the mean square error (MSE), between the prediction and the real power profile?
- What influence do the parameters of the model have on its MSE?
- What influence does the complexity of the training data have on the MSE of the model?

1.3 Research Strategy

Based on literature research, existing energy consumption prediction methods and suitable model architectures from other application areas will be collected. Their suitability will be evaluated with regard to the specific task of predicting individual processes' energy consumption based on the process parameters.

A use case will be developed to evaluate the performance of the different architectures. It is based on the battery assembly of the automotive supplier AVL Austria in Graz[2], a partner of the project F4R. The production process and historical power profiles of the production machines serve as a basis for generating training data for the energy model. Furthermore, real energy data from measurements on an Industrial Robot (IR) from an open-research dataset is used.

The suitable model architectures will be implemented and assessed with training data from the use case based on the prediction error, the required amount of data, and the required time for training.

1.4 Structure of the Work

The thesis is structured as follows: Chapter 2 gives an overview of the State of the Art concerning energy considerations in production planning as well as certain machine learning techniques used for this purpose. Chapter 3 then describes the development and implementation of an energy

[2] www.avl.com

model architecture and assessment metrics. The experimental setup to test the Energy Model is described in Chapter 4. Chapter 5 provides the results of the experiments with different model architectures, followed by a discussion. Finally, Chapter 6 concludes the thesis by summarizing findings and an outlook to possible future research.

1.5 Support

The presented work results mostly from the R&D project Factory4Renewables, funded in the KLIEN research program "Energieforschung", Österreichische Forschungsförderungsgesellschaft (FFG) grant no. 881136. The goal of F4R is to investigate methods to adapt the production schedules of factories to time-dependent electricity prices and time-dependent availability of renewable electricity.

The training of the energy model was conducted on the VSC-5 of the Vienna Scientific Cluster (VSC), at the time of writing Austria's most powerful supercomputer. VSC provides supercomputing capacities to the participating Austrian universities.

Chapter 2

State of the Art

So far, energy considerations in production planning have only played a minor role, e.g., in cutting peak loads [Bra+21]. This is also reflected in the existing amount of literature about energy measurements and modeling in manufacturing. The motivation behind previous research efforts regarding energy consumption in manufacturing industries can be categorized into several groups: One key factor is reducing energy consumption overall and increasing energy efficiency; this also includes the development of benchmarks to measure energy efficiency. In recent years, allocating Carbon Dioxide (CO_2) emissions down to the product level has become another driver for monitoring energy consumption in manufacturing. Finally, the third driver is the urge to increase the use of RES and the associated adaption of consumption profiles to the availability of electricity from RES.

Although there are numerous examples of incorporating energy-efficiency or energy cost into scheduling (be it in manufacturing [Gon+15; Del+16; Bän+21], computing [Bam+16; Che+22] or even robot kinetics [Her+21]), the methods to assign energy-consumption to single processes or operations are often not explained in detail. While physical models for the energy consumption of machine movements (e.g., drilling [Che+14]) or active logic gates [Cha+92] exist, it is not feasible in most cases to split manufacturing or computing processes into these kinds of atomic processes to model their energy consumption accurately. For applications with homogeneous processes and energy consumption patterns, it can be sufficient to use averaged energy profiles [Gon+15]. However, as the complexity of captured processes increases, more versatile methods and models are needed.

Literature Review reveals that, as of 2021, the scientific research on “Predicting and Forecasting the Electrical Energy Consumption in the Manufacturing Industry” has predominantly focused on energy efficiency, as opposed to energy flexibility [WW21, p. 9], there are even several literature reviews dedicated to energy efficiency in manufacturing in particular [May+17; Men+19]. In this context, it is essential to emphasize the difference between prediction and forecast according to [Box+16]: while prediction derives unknown values from known inputs, forecasts anticipate future values based on their present values and/or history. In the context of energy modeling, this corresponds to the difference between predicting energy or load values at time t from observations (present and past values) of other quantities available at time t as opposed to forecasting energy or load values at time $t + x$ from observations of the same quantity at time t [Box+16].

In this chapter, existing methods from the literature dealing with energy consumption modeling in manufacturing and load forecasting, in particular, are presented separately by model-driven and data-driven approaches.

2.1 Model-Driven Approaches

Model-driven approaches for forecasting energy consumption rely on explicit mathematical models based on physical principles and domain-specific knowledge. These models simulate the relationships within the system. The strengths of model-driven approaches include high accuracy and interpretability, while weaknesses involve complexity in development, data requirements (for calibration), reliance on assumptions, and potential computational intensity in application. Existing research can be categorized into physical models of the energy consumption of very specific processes or even parts of processes and methods that divide processes into subprocesses and classify the energy consumption or machines to build subclasses of averaged energy profiles.

2.1.1 Classification of Energy Consumption

The systematic classification of machine energy consumption can be helpful in gaining insight into a whole factory's, product's, or process' energy consumption and, furthermore, in assessing the potential for energy savings. It is also necessary for the detailed modeling of energy consumption.

One possible classification is based on the correlation between energy consumption and number of pieces produced. Supporting processes like cooling/heating, ventilation, or centrifuge rotation are independent of the number of pieces produced and can make up even more of the overall energy consumption than actual machining [Gut+06].

A similar yet slightly different classification is the one into direct and indirect energy. The distinction is made between the energy needed for the production itself (e.g., machining, painting) and energy needed to "maintain the 'environment' in which the production processes are carried

Table 2.1: Energy consumption classification based on [Gah+16].

Non-processing Energy Demand	**Processing Energy Demand**
Machine turn on	Job related
Machine idle	Varying job related
Machine setup	Machine related
Machine turn off	Flexible
Material storage	

out" [SR11, p. 260] (e.g., lighting, heating). The direct energy can further be divided into theoretical energy, i.e., the minimum energy required for a process, and auxiliary energy, i.e., energy consumed by supporting processes or "supporting activities and auxiliary equipment for the process" [SR11, p. 260]. While indirect energy is completely independent of throughput, direct energy is partly dependent (theoretical energy) and partly independent (auxiliary energy) of throughput. Likewise, [Gah+16] classifies energy consumption into non-processing and processing energy consumption but subdivides these categories in more detail (see Table 2.1).

Other methods to categorize manufacturing energy consumption, specifically in machining, include the energy consumption classifications by abstraction level, machine tool states, and breakdown by machine tool components, which are "three of the leading classification methods for energy consumption in machining" [Zha+17, p. 144]. The energy classification into abstraction levels has three categories: firstly, machine tool level; secondly, spindle level; and finally, process level [Liu+15, p. 477]. While machine tool level energy consumption is useful for an evaluation of the overall machine efficiency and more independent of the specific type of production, spindle level energy is very specific to mechanic processes, depends purely on the motor efficiency, and is "incomparable to other manufacturing processes" [Zha+17, p. 144]. The process level energy, as mentioned in [Zha+17], is specific to material removal processes and is the energy needed for the actual removal. It is independent of the machine tool used and useful for the selection of process parameters with regard to energy consumption and quality [Zha+17].

The energy classification based on machine tool states is based on two [Kel+12] or three [BM13] elementary machine states, respectively. [Kel+12] only uses a "BasicState" and a "CuttingState", while [BM13] also introduces a "ReadyState" between the "BasicState" and "CuttingState". Another possible state classification is based on an idle mode, a run-time mode, and a production mode [DG04]. Independent of the specific terms and methods to identify machine states, these classifications all have in common that they allow a temporal analysis of energy hotspots while implicitly highlighting the tools and parameters to modify in order to save energy.

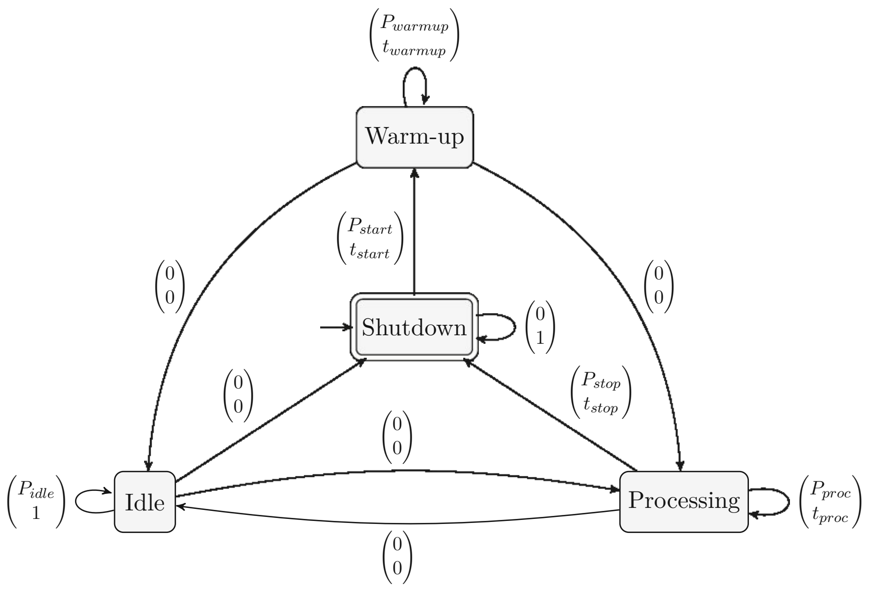

$\begin{pmatrix}P_{warmup}\\t_{warmup}\end{pmatrix}$
Warm-up
$\begin{pmatrix}P_{start}\\t_{start}\end{pmatrix}$
$\begin{pmatrix}0\\0\end{pmatrix}$
Shutdown
$\begin{pmatrix}0\\1\end{pmatrix}$
$\begin{pmatrix}P_{stop}\\t_{stop}\end{pmatrix}$
$\begin{pmatrix}P_{idle}\\1\end{pmatrix}$
Idle
Processing
$\begin{pmatrix}P_{proc}\\t_{proc}\end{pmatrix}$

Figure 2.1: FSM with assigned (power, time)-vectors, based on [Le+13],[Shr+14]

Like in [SR11] and [Gah+16], the breakdown by machine tool components distinguishes between fixed and variable energy with respect to the machine utilization [Zha+17, p. 145]. Furthermore, Li et al. associate the fixed energy with the specific tools that cause the consumption [Li+11]. This is helpful in identifying the tools most suitable for energy-saving measures.

2.1.2 Finite State Machine (FSM) Models

The state-based classification of energy consumption is closely related to the behavior of manufacturing equipment, which can be described with a Finite State Machine (FSM). The appeal of this model lies in its simplicity and clarity. Several publications have used FSMs to describe machine states and associate energy consumptions with these states and also state transitions [Le+13; Shr+14; DV08].

In [Le+13], the authors propose an "FSM energy consumption model" [Le+13, p. 585] consisting of states and state transitions to model machine behavior (see Figure 2.1). The three main objectives of their model are to extrapolate from real-time energy measurements to the current machine state, energy consumption evaluation on the machine level, and energy-efficient scheduling [Le+13, pp. 585-586]. However, their understanding of energy-efficient scheduling is to choose the most energy-efficient machine and not the temporal shift of operations on given machines to influence the energy consumption profile.

With the objective of minimizing energy costs, [Shr+14] propose an FSM-like scheme of states and state transitions to acquire time-discrete energy profiles of a production process. The FSM represents the three basic operational states, "processing", "idle", and "shutdown", and the possible transitions between these states (including staying in the same state). Each state and each state transition is assigned a tuple of duration and energy consumption, although how to determine these values is not described in detail. A mathematical model, consisting of constraints and an objective function, is presented to solve the scheduling problem while minimizing the costs of energy consumption [Shr+14].

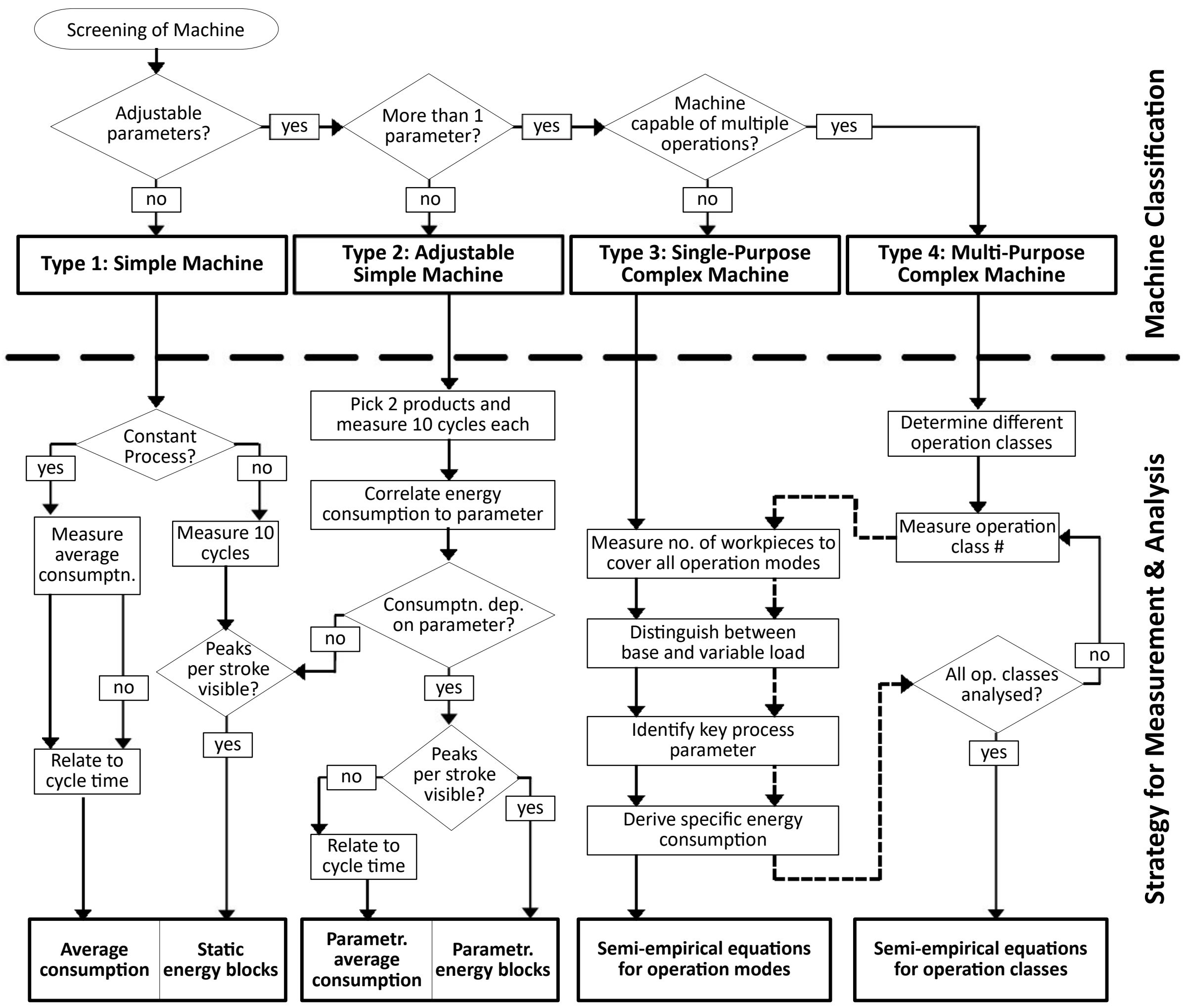
Dietmair et al. [DV08] described an approach to define machine states based on energy measurements during machine operation. Peaks and plateaus in the P(t)-diagram are linked to the corresponding machine states. Together with the state/transition graph of the machine, predictions about the energy consumption of a certain machine usage can be made. It is stressed that "a small number of basic measurements during elementary operations" is sufficient to parameterize the model [DV08, p. 577].

Even though FSMs are not mentioned explicitly in the paper, the EnergyBlocks methodology [Wei+11] can be used to assign energy consumptions to machine states as so-called "Energy-Blocks". These EnergyBlocks are stored in a database and can be strung together to model the energy consumption of any series of operational states. The methodology allows three different perspectives: the equipment perspective (i.e., from a machine point of view), the product perspective, and the production system perspective. Each perspective allows different forms of analysis, e.g., the equipment perspective enables the analysis of the energy costs of a machine and the share of the total consumption, while the product perspective allows the calculation of product-specific energy costs and also manufacturing times [Wei+11].

Complementary to purely state-based approaches, [LH23] presented a matrix of load categories with the dimensions process dependency (process-dependent/-independent) and load type (constant/controlled constant/variable) to disaggregate single machine loads into different components. Unlike the state-based categorization, this component-based approach also takes into account internal interdependencies and job characteristics, which enables it to extrapolate the energy consumption from one job to another. The model also applies a just-in-time method to determine a job's energy consumption to take into account not directly process-related energy consumption based on environmental conditions, which would be neglected when pre-fetching a job's energy consumption before scheduling. However, this method is less performant.

2.1.3 Machine Classification

Schmidt et al. [Sch+15] developed a model to determine the energy consumption of any product on a certain machine. The decision tree depicted in Figure 2.2 is their guideline for dealing with the huge variety of machines and choosing a suitable strategy for measurement and analysis. Machines



Screening of Machine
Adjustable parameters?
yes
More than 1 parameter?
yes
Machine capable of multiple operations?
yes
no
no
no
Type 1: Simple Machine
Type 2: Adjustable Simple Machine
Type 3: Single-Purpose Complex Machine
Type 4: Multi-Purpose Complex Machine
Machine Classification
Constant Process?
yes
no
Measure average consumptn.
Measure 10 cycles
Peaks per stroke visible?
no
yes
Relate to cycle time
Pick 2 products and measure 10 cycles each
Correlate energy consumption to parameter
Consumptn. dep. on parameter?
no
yes
Peaks per stroke visible?
no
yes
Relate to cycle time
Measure no. of workpieces to cover all operation modes
Distinguish between base and variable load
Identify key process parameter
Derive specific energy consumption
Determine different operation classes
Measure operation class #
All op. classes analysed?
no
yes
Strategy for Measurement & Analysis
Average consumption
Static energy blocks
Parametr. average consumption
Parametr. energy blocks
Semi-empirical equations for operation modes
Semi-empirical equations for operation classes

Figure 2.2: Machine classification according to [Sch+15].

are categorized into four groups according to their complexity, i.e., according to their number of adjustable parameters and their number of operations. Different strategies for measurement and analysis are proposed depending on the machine type and additional information about the consumption profile; these range from average consumption to parameterized energy blocks and semi-empirical equations. The proposed methodology was applied in a case study at a job-shop factory producing parts for the electricity grid with 33 machines to derive the specific energy consumption of 7 products. Also, a factory-wide energy hotspot analysis was carried out.

2.2 Data-Driven Approaches

Data-driven approaches use large data sets to identify patterns and correlations without relying on predefined models. They employ machine learning algorithms to learn from historical data. In contrast to model-driven approaches, data-driven methods allow for adaptability to complex relationships that may not be easily captured by predefined models. The advantages of data-driven approaches include flexibility in handling diverse data types and patterns. Still, they can be at a disadvantage with regard to interpretability, potential overfitting, and reliance on the quality and representativeness of the training data.

Industry 4.0 and smart factory paradigms have increased the interest in collecting data from the manufacturing process to leverage the extraction of useful information about operational efficiency, product quality, predictive maintenance, and overall system optimization. However, the evaluation of the collected data from sensors and production systems requires effective data analytics tools and machine learning algorithms to derive meaningful insights for the continuous improvement of the manufacturing processes. The availability of data about energy consumption linked with machining data enables research efforts in data-driven prediction models for machine tool energy consumption [Kum+20]. The data-driven models can either be used to derive present machine loads from aggregated load measurements or auxiliary machine data, i.e., for prediction, or to derive future machine loads from past and present load measurements, auxiliary machine data, and production schedules, i.e., for forecasting.

This section is divided into prediction and forecasting (as defined by [Box+16]) of energy consumption and power rather than by the different machine-learning algorithms since, in most existing literature, several machine-learning algorithms are tested at a time to find the most suitable one.

2.2.1 Prediction Applications

Applications to predict energy consumption in an industrial context can mostly be linked to non-intrusive load monitoring. Compared to intrusive load monitoring, which relies on sensors installed at each appliance of which the energy consumption should be measured, non-intrusive load monitoring relies on information by sensors installed only at the entrance of an electrical circuit [Liu20]. This implies load disaggregation (to assign the load to the individual appliances), often including additional parameters like control signals, process step information, or carrying out algorithm training [See+19].

In [Tan+21], an approach to predict individual machine loads based on the total load of four machines (laser welding, laser trimming, two ovens) was presented. Six different algorithms (four Long Short-Term Memory (LSTM)-related, two gradient-boost-related) were trained in a supervised manner with tuned hyperparameters and ranked according to the mean absolute error and root mean square error. Interestingly, the two rankings were quite different from one another, with a light gradient-boosting machine being the best in terms of mean absolute error and an ensemble bi-directional LSTM in terms of the root mean square error. Ultimately, the disaggregated loads were used to predict the activity state and to estimate the production of the single machines.

In [AB15], an ensemble model of Neural Networks (NNs) and the k-nearest-neighbors (k-nn) method were applied to predict the energy consumption of manufacturing certain parts on a machine and to optimize the energy consumption. The k-nn approach was used to identify the most similar training data to the data presented and to derive local information from that training data. The approach accounted for uncertainties in the prediction by estimating prediction intervals instead of points.

Both unsupervised and supervised deep learning were used in [He+20] to extract features from machinery data and predict the energy consumption of a Computer Numerical Control (CNC) system for milling and grinding. Unsupervised learning is used for feature extraction from the raw input data, and a 1-D Convolutional Neural Network (CNN) is used for the energy prediction. It produced predictions with a lower root mean square error (RMSE) than the three compared techniques (support vector regression, Gaussian process regression, extreme learning machine). It is worth noting that the method described in this paper only mentions energy prediction, even though the used input features (e.g., rotational speed, linear speed, feed rate) are probably already known at the time of production planning and could, therefore, also be used for energy forecasting.

Energy prediction can also be used for predictive maintenance based on energy consumption and to improve the energy efficiency of machinery. In [Ber+22], a state-space representation together with a Hammerstein modeling for non-linearities was used for online energy modeling, which also captures long-term drifts in the energy consumption caused by equipment degeneration. The model is data-driven in so far as it is automatically adapted when the error between the current consumption and the output of the reference model becomes too big. The approach was validated in an industrial testbed consisting of spindles, a servomotor for translation, a pneumatic system, and a cooling system (among others). As inputs, the spindle and servomotor speeds and the activation/deactivation signals of the cooling and pneumatic system were used. These devices were the most energy-consuming. The adaptive model doubled the fit rate of the non-adaptive model (68 % vs. 33 %) [Ber+22].

2.2.2 Forecast Applications

The distinction between energy forecasting and energy prediction in existing literature is often not made very clear. Most publications even seem to use it interchangeably. This is the reason why some sources in this section use the word "prediction", even though according to the temporal nature (present vs. future) of the definition by [Box+16], their concepts can be linked to forecasting.

Much literature exists on time series forecasting of energy consumption in the manufacturing industry (e.g., [Rib+20; MB18]). However, these examples mostly concern factory-wide energy consumption or only short-term forecasting. Another distinction has to be made between the forecasting of vectors (mostly load profiles) and scalar values (mostly the sum of the energy consumption). While forecasting the scalar energy consumption can be sufficient for ranking machinery or processes according to their energy intensity, the temporal resolution of these scalar values is limited to the duration of the processes. Also, the shape of the energy consumption (fluctuating or even) is omitted.

In [MH20], NNs (both feed-forward and recurrent) were used to forecast the energy consumption of a manufacturing environment as well as the temperature and air humidity of the manufacturing environment. Even though the models only forecast the entire building's energy consumption, manufacturing schedules played a major role in the forecast by determining the consumed energy for manufacturing itself as well as the released heat from machines, which in turn had an effect on the energy consumption of heating, ventilation, air conditioning (HVAC) facilities. Besides, weather conditions like dry bulb temperature, wind speed, and relative humidity were used. In their case study, the Recurrent Neural Networks (RNNs) exhibited better results than the feed-forward NNs for the building's energy consumption forecast (mean absolute error (MAE) 0.068 vs. 0.097, RMSE 0.040 vs. 0.060).

LSTM-based time series forecasting was used in [MB18] to optimize resource utilization and energy consumption. The decisions the LSTM could make comprised the next operation to be performed as well as the machine resource that this operation should be performed on. Decision criteria were energy, utilization, and time, weighted 60/20/20 %. Each of the N robots on a shop floor issued a bid on the costs of the next operation offered by the scheduler based on past operations. The scheduler then made the decision on which robot to schedule the operation on. However, the pilot implementation did not consider several possible operations and only assumed scalar values for the energy consumption.

A Random Forest was used in [Bri+21] to forecast the energy demand of CNC milling for lot size one manufacturing. First, a training part was produced to acquire training data to train a Machine Learning model. This model then forecasted the energy consumption for the production of a validation part based only on the numerical control instructions. Each instruction was broken down into features like the category of operation or go-to positions. Instructions whose

Table 2.2: Comparison between model-driven and data-driven approaches.

	Model-driven approaches	**Data-driven approaches**
Strengths	+ High accuracy + Interpretability	+ Adaptability to complex relationships + Less reliance on expert knowledge
Weaknesses	- Complexity in development - Reliance on assumptions	- Interpretability - Reliance on quality of data

energy consumption was not significantly higher than the idle consumption were ignored. The output was a scalar value of the energy consumption for a certain instruction. While the Random Forest performed better on predicting the energy consumption of the movements (0.2 % - 3.3 % vs. 4.0 % - 18.5 %), the Decision Tree performed better for the spindle and tool change system prediction (8.9 % - 11.8 % vs. 11.7 % - 46.2 %).

2.3 Concluding Evaluation

The focus of existing research conducted in the area of industrial energy consumption analysis is on energy efficiency and cutting peak loads rather than energy flexibility. Only very few publications have dealt with deriving energy consumption from production schedules. While some model-driven approaches already exist to predict the energy consumption of manufacturing processes, they have the strategic disadvantages of model-driven approaches: they are tailored to a certain process, require a deep understanding of the process itself, and are complex to develop. A summary of the differences between model-driven and data-driven approaches is given in Table 2.2.

Physical models of the energy consumption of very specific processes or even parts of processes are too unflexible and require too much development effort for wider application. Also, they often model the energy consumption on a much lower level than the process level. Averaged energy profiles per process can be sufficient for processes with non-alternating parameters and fixed energy profiles.

A more detailed variant of averaged energy profiles are methods that divide processes into subprocesses and classify the energy consumption or machines to build subclasses of averaged energy profiles, e.g., by means of an FSM. The development effort of this variant is limited to defining subprocesses or substates of machines (including their temporal sequence) and averaging the respective load profiles. However, such models only work with known parameter values and cannot interpolate between them.

Existing data-driven approaches often focus on load disaggregation or non-intrusive load monitoring, i.e., associating load shares from a central measurement point to single machines and processes using schedule data and process parameters. While certain machine learning algorithms used for load disaggregation can also be used for load forecasting from process parameters, they are fundamentally different concepts.

The distinction between prediction and forecasting by [Box+16] proves to be very useful in this context. However, this terminology is not very common in related work, instead, the predominant terminological distinction between prediction and forecasting seems to be linked to certainty and the time horizon (prediction associated with relative certainty and short-term horizon, forecasting associated with relative uncertainty and long-term horizon).

Existing load forecasting models either forecast the load profile based on the past load profile or forecast the load profiles on a higher level than the process level, e.g. factory-wide. Also, not all models forecast a whole load profile but instead just the scalar energy consumption. This way, if a process overlaps the border between two electricity price time slots, it cannot be determined what share of the energy consumption must be assigned to which time slot.

2.4 Scientific Value and Novelty of the Proposed Model Type

A research gap exists for data-driven approaches that forecast the electrical load profile of the single processes of a production schedule. A data-driven approach would have the benefit that the architecture would be generally applicable and easily adaptable by training the model with new data. This kind of energy forecast model would benefit efforts to adapt industrial energy consumption to the availability of renewable energy and fluctuating energy prices by assigning energy consumption to process steps already at the time of production planning. By having a pre-determined energy consumption assigned to each process step during production planning, manufacturers can strategically schedule energy-intensive tasks during periods of abundant renewable energy or lower energy costs. Moreover, the model facilitates the identification of energy-intensive stages, allowing for the exploration of alternative materials or processes that may reduce energy consumption without compromising product quality.

Chapter 3

Energy Model Architecture

This chapter presents the development of a generic Energy Model that is able to generate power profile predictions for manufacturing processes based on process parameters and which can be easily adapted to different use cases. First, requirements for the model architecture are defined, and the advantages of a data-driven model are outlined. After that, requirements are defined for the historical data that is used for training. Subsequently, the model architectures to be tested are laid out, followed by their implementations with the Python library Keras (details about Keras in Section 3.4). And finally, the metrics to assess and compare these model architectures are explained.

3.1 Requirements for the Model Architecture

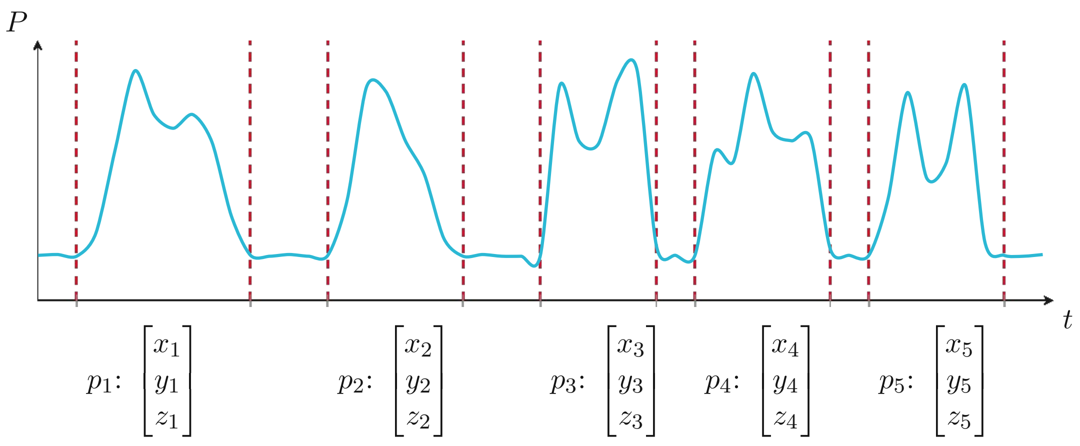
As evident from the abundance of existing literature on energy modeling in industrial manufacturing, numerous approaches exist to model the energy consumption of production processes with different focuses and strengths. Therefore, before selecting one approach or several approaches to compare against each other, it is necessary to clearly define the requirements and desired abilities for the Energy Model architecture with regard to forecasting the load profile of manufacturing processes and schedules.

The first requirement for the model architecture is that it be easily adaptable to various industrial use cases. This means that by simply replacing the historical data and performing automated procedures, the model should work in a different use case. The second requirement is that the model can interpolate between parameter values from the historical data. This ensures that the model also performs well in dynamic and flexible production environments, e.g., with customizable products or parameterizable machines. Lastly, the third requirement is that the model encompasses all process-dependent energy consumption, i.e., the energy consumption of all operations that are performed on the manufacturing machines. This includes idle consumption since it is not just some constant base load but depends on the timing of scheduled processes and on the duration of idle periods between processes, particularly for machines requiring standby mode during interventions. Scheduling processes consecutively followed by a longer idle period allows for complete machine shutdown during the break and, consequently, a lower idle energy consumption compared to multiple short idle periods scattered between the processes.

This combination of requirements cannot be fulfilled with existing energy modeling approaches from the literature. Finite State Machine (FSM) models [Shr+14; DV08; Wei+11], while being adaptable to various industrial use cases, cannot interpolate between parameter values from the historical data. Specific physical models can also handle unprecedented settings but require a lot of effort to develop, are tailored to a specific application, and are, therefore, hardly adaptable to various use cases. Data-driven approaches also only exist for specific use cases [Bri+21], to parameterize model-driven approaches [Ber+22], or for factory-wide energy consumption [MH20]. Therefore, by combining existing approaches from other application domains, a novel Energy Model has to be developed.

The major decision that has to be made is the one between a model-driven or a data-driven approach. As described in Chapter 2, the strengths of model-driven approaches include high accuracy and interpretability. Especially interpretability is an important advantage over data-driven approaches since it enhances trust and reliability in the model. Moreover, examining the model's inner workings enables insights for discovering hidden drivers of energy spikes beyond the obvious parameters and for the development of scheduling-specific optimization strategies. By understanding how process timing and process sequence impact energy consumption, production plans can be adapted to minimize, e.g., idle periods and optimize machine states. Model-driven approaches also fulfill the requirements of being able to interpolate between known data points as described above. A model based on principles of physics, thermodynamics, or engineering can make predictions for scenarios not explicitly present in the training data by using the underlying mathematical or physical principles that describe the system. Also, idle times can be modeled just like production processes.

However, with regard to the requirement of adaptability described above, data-driven approaches have an advantage. Their strengths include flexibility and adaptability; new data and information can be easily integrated. They require much less domain-specific knowledge and understanding of the underlying principles. At the same time, they are able to interpolate between known data points as long as they are not too far apart, which is also the main reason for not just using historical load profiles without further treatment as a forecast for the load



P
t
p_1: $\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$
p_2: $\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$
p_3: $\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix}$
p_4: $\begin{bmatrix} x_4 \\ y_4 \\ z_4 \end{bmatrix}$
p_5: $\begin{bmatrix} x_5 \\ y_5 \\ z_5 \end{bmatrix}$

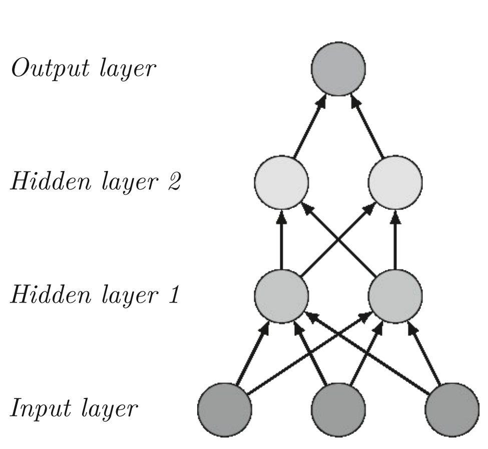
Figure 3.1: Schematic load measurement with overlayed process parameters.

profile of processes. Additionally, idle times, as long as they are present in the historical data, can be treated and modeled like production processes. This makes data-driven models strong candidates for the given use case, particularly in situations where domain-specific knowledge may be limited or where the system's behavior is complex and not fully understood.

3.2 Data Requirements

The training data is required to contain process parameters and the load profile of the process. Since the temporal sequence of processes and their parameters often influences the power consumption of a process, the full load profile of a machine, together with machine schedule data including process parameters, is required. This way, sections of the load profile can be combined with the corresponding process parameters including information about previous processes or idle times as schematically depicted in Figure 3.1. This information can be treated as another parameter of the succeeding process. As far as the Energy Model is concerned, the idle times and their load profiles can be treated just like normal processes. This way, also idle energy consumption is covered.

The model receives the process parameters as an input vector and should produce the power profile of the given process as an output vector. The power profile values are equidistant and the desired temporal resolution must be determined before training the model. If necessary, the training power profiles have to be resampled. The variable number of parameters per process, the variable duration of processes, and, hence, the variable length of power profiles require to deal with input and output vectors of varying lengths. This can be done either by choosing a model architecture that can handle variable-length input and output data or by padding the input and output vectors so that their lengths are equal. For the power profile, padding values



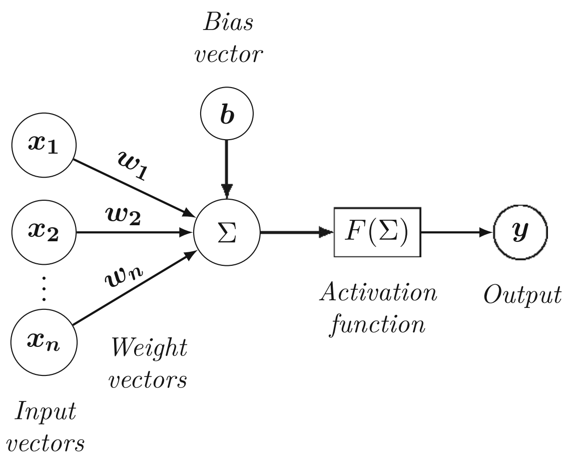


Figure 3.2: Schematic structure of a Neural Network (left) and a single neuron (right).

just need to be added at the end of a vector; for the process parameter vectors, it is better to use one column per parameter and add padding values in the columns of the parameters that are not used. This way, the position of a parameter remains the same and does not need to be learned by the model.

3.3 Data-driven Model Architectures

The selection of a suitable model architecture plays a crucial role in extracting complex patterns and dependencies embedded in energy consumption data. This section presents the chosen data-driven model categories to be tested, each tailored to address specific challenges in load profile prediction. NNs, known for their adaptability and ability to handle complex relationships, serve as a benchmark for comparison. RNNs explicitly consider temporal dependencies embedded in sequential energy data, allowing them to model dynamic patterns and capture long-term dependencies. Furthermore, Encoder-Decoder Networks, with their ability to process input sequences and generate output sequences, i.e., sequence-to-sequence (Seq2Seq) data, are well-suited for tasks like predicting future energy consumption based on historical data. Lastly, ensemble learning, collectively utilizing multiple models, is introduced as a strategy to improve predictive performance.

3.3.1 Neural Networks

Neural Networks (NNs) have become very popular in recent years, thanks to their remarkable ability to learn complex patterns and relationships from data. Inspired by the structure and function of the human brain, NNs are versatile machine learning models capable of learning

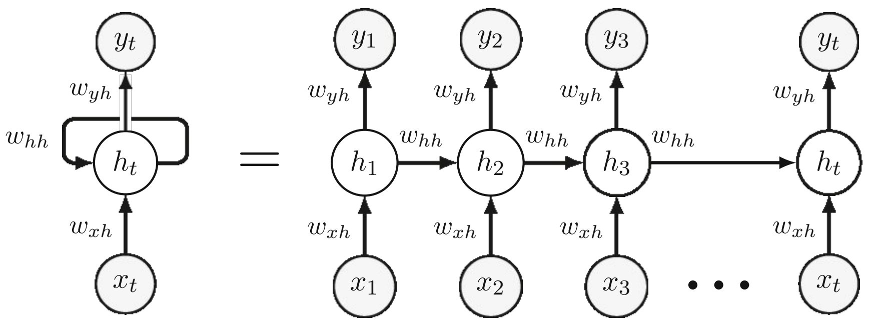


Figure 3.3: Schematic structure of a Recurrent Neural Network (based on [Agg18, p. 39]).

complex patterns and relationships from data. They are composed of layers of interconnected nodes, or neurons, that pass information between each other in a hierarchical manner (see Figure 3.2). Each neuron receives weighted inputs from other neurons, sums these weighted inputs together with a bias vector, applies a mathematical function (the activation function) to this sum, and produces an output. Activation functions introduce non-linearity into the network, which is crucial for learning complex patterns and relationships. The outputs of neurons in one layer are then passed to neurons in the next layer, and so on. This hierarchical structure allows NNs to learn complex relationships between data points.

During the training of the NN, the weight vectors and the bias vectors are optimized so as to minimize the network's loss function or error metric by iteratively adjusting the values of the weights and biases using optimization algorithms such as gradient descent. To calculate the gradient of the loss function with respect to the weights of the connections between neurons, backpropagation is used. This gradient information is then used by the learning algorithm to update the weights. NNs have a wide range of applications, including image recognition, natural language processing, and anomaly detection [Kub17].

3.3.2 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a type of artificial neural network that is capable of learning sequential data. This means that it can be used to model data that has a natural order, such as text, speech, and time series data. RNNs are able to do this by using feedback loops, which allow them to incorporate information from previous inputs into their current predictions.

The core of RNNs is the recurrent structure (see Figure 3.3), which allows them to process sequential data by maintaining a memory of previous inputs. This memory is typically represented by a set of hidden states h, which are vectors that capture the information from the previous inputs x. The hidden states are then used to generate the current output y. The weights w are

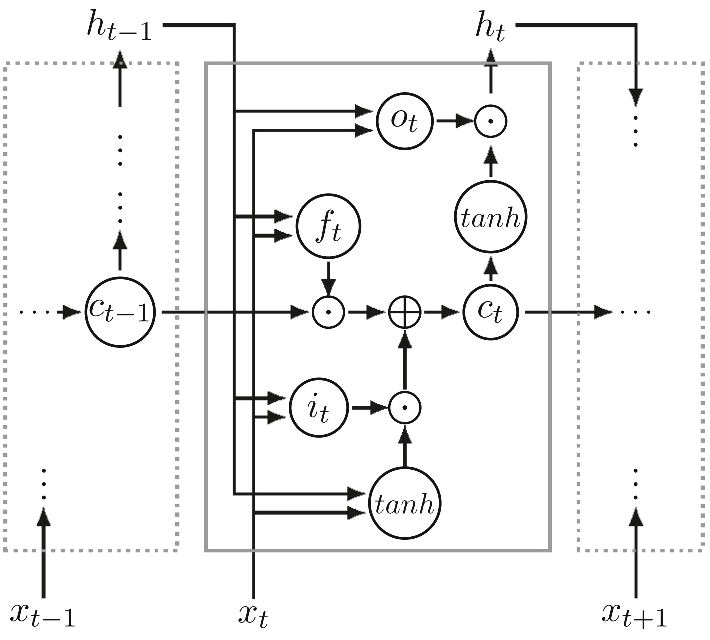


Figure 3.4: Structure of an LSTM cell (based on [SB18]). $\odot$ stands for element-wise multiplication and $\oplus$ for vector addition.

the parameters that determine the strength of the connections between the different parts of the RNN. These weights are learned through a process called backpropagation through time, which is an extension of the standard backpropagation algorithm that is used to train feedforward neural networks [Agg18, p. 38f].

An enhancement to the traditional RNN architecture are LSTMs and Gated Recurrent Units (GRUs). Both architectures address the vanishing gradient problem, which is a common issue that arises in traditional RNNs when training on long sequences. The vanishing gradient problem occurs because the error signal that is backpropagated through time becomes exponentially smaller as it goes back in time. This makes it difficult for the RNN to learn long-term dependencies in the data.

According to [SB18], LSTMs address the vanishing gradient problem through the implementation of three multiplicative gates (input gate, forget gate, output gate). These gates control the information flow within the LSTM cell. The input gate controls the proportion of the current input (x_i) incorporated into the memory cell (c_i), the forget gate controls the fraction of the previous memory cell information (c_{t-1}) retained, and the output gate controls the proportion of the current memory cell state exposed as the output (h_i).

Within the LSTM's recurrent structure, the forget gate uses the identity function (gradient = 1) as its activation function. This implies that when the forget gate is open, the gradient is fully transmitted to preceding time steps, enabling the model to learn long-term dependencies effectively. The internal structure of an LSTM cell is depicted in Figure 3.4. The LSTM processes each input (x_i) sequentially, updating its memory state (c_i) based on the gate values. Following each update, an output (h_i) is generated based on the modified memory state. The gate outputs, the cell state, and the hidden state are calculated as follows [SB18]:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{3.1}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{b}_f) \tag{3.2}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{3.3}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{3.4}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{3.5}$$

where $\mathbf{i}_t$ is the input gate, $\mathbf{f}_t$ the forget gate, $\mathbf{c}_t$ the cell state, $\mathbf{o}_t$ the output gate, and $\mathbf{h}_t$ the hidden state. The weight matrices $\mathbf{W}$ and biases $\mathbf{b}$ are time-independent and are used to parameterize the connections inside the LSTM cell. While weights represent the strength of connections between neurons in different layers and are specific to connections between neurons, biases act as offsets to the weighted sum of inputs reaching a neuron and are applied globally to all neurons in a layer. The functions $\sigma(x)$ and $\tanh(x)$ are defined as follows:

$$\begin{aligned} \sigma(x) &= \frac{1}{1+e^{-x}} \\ \tanh(x) &= \frac{e^{2x}-1}{e^{2x}+1} \end{aligned} \tag{3.6}$$

Solving the vanishing gradient problem through the described approach allows the LSTM to learn long-term dependencies in the data, which makes it well-suited for tasks such as machine translation, text summarization, and speech recognition [PG17].

GRUs are simpler than LSTMs; they do not use a dedicated cell state and control the information flow to the hidden state with a single so-called reset gate instead of a separate forget gate and output gate. Moreover, a so-called update gate combines the function of the input gate and forget gate from the LSTM cell. GRUs have the advantage of being faster to train than LSTMs and they have been shown to be just as effective in many tasks. With limited data, GRUs' parameter efficiency could lead to slightly better generalization. However, as data availability increases, LSTMs become increasingly powerful. The LSTM, being older and more widely studied, has benefited from more extensive testing compared to the GRU. Therefore it is often seen as a safer and more reliable choice, especially for longer sequences and larger datasets [PG17].

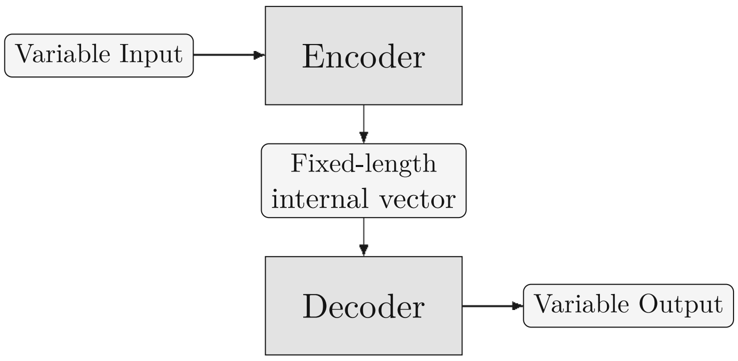


Figure 3.5: Schematic structure of an Encoder-Decoder network [HS23].

3.3.3 Encoder-Decoder Networks

Encoder-decoder architectures are a type of RNN that is commonly used for sequence-to-sequence tasks, such as machine translation, text summarization, and speech recognition. In an encoder-decoder architecture, the input sequence is encoded into a fixed-length vector, and then the decoder uses this vector to generate the output sequence (see Figure 3.5). This structure enables encoder-decoder architectures to predict data whose input and output vectors have different, varying lengths. Encoder and decoder can be any RNN architecture, the most prominent architectures being LSTMs and GRUs. Following the unfolded (i.e., the righthand) RNN structure in Figure 3.3, the unfolded structure of an encoder-decoder network is depicted in Figure 3.6 [Sut+14].

First, the inputs are passed one after the other to the input of the encoder, which discards the outputs. By the time the last input has been processed, the hidden state h (and, in the case of an LSTM, also the cell state c) is a representation of the input sequence with all its internal relations and is passed as a so-called context vector to the decoder. The decoder loads

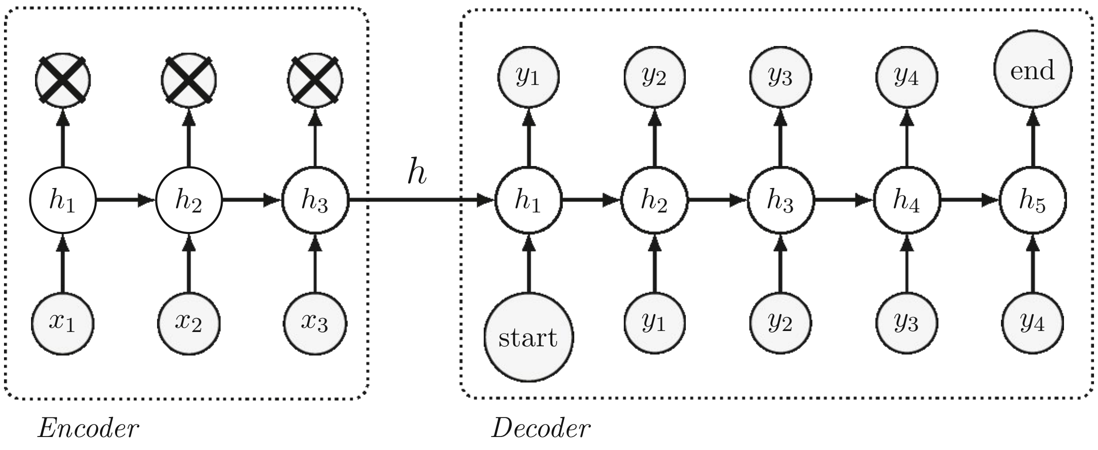


Figure 3.6: Unfolded structure of an Encoder-Decoder network (based on [Sut+14]).

the context vector to its hidden state and, after being passed a start token at the input, produces the first element of the output sequence. This output is then fed back in a loop to the input of the decoder, the next output is produced, and so on, until an end token is produced [Sut+14].

To accelerate the training, a method called teacher forcing is applied [Vas19]. This means that the output loopback to the input of the decoder is interrupted during training. Instead, the true output values are applied to the decoder input (shifted by one). This means that in Figure 3.6, the input of the decoder in the second time step would need to be changed from y_1 to $y_{1,true}$ and so on.

3.3.4 Ensemble Learning

Ensemble learning is a machine learning technique that combines multiple models to improve predictive performance. By combining the predictions of multiple models, ensemble learning can often outperform any single model. There are different techniques for combining multiple models [SN19]:

Bagging, which stands for Bootstrap Aggregation, trains multiple models on different subsets of the training data. Each model is then trained on its own subset of data and its predictions are combined to produce the final prediction. This method helps to reduce the variance of the model, which can improve accuracy.

Boosting, also known as sequential learning, trains models sequentially. Each model is trained to focus on the errors of the previous models. This helps to reduce the bias of the model, which can also improve accuracy. However, boosting cannot address the issue of overfitting.

Stacking, also referred to as blending, combines the predictions of multiple models by training a meta-learner on the predictions of the base models. Unlike bagging and boosting, which combine models at the prediction stage, stacking combines models at the learning stage. In stacking, multiple base models are trained on the original training data, and their predictions are used to train a meta-learner. The meta-learner learns to combine the predictions of the base models to produce the final prediction.

3.4 Implementation with Python and Keras

For the implementation of the model architectures mentioned in Section 3.3, the Python library Keras was used. Keras is a high-level neural network Application Programming Interface (API) written in Python that provides an interface for building and training deep learning models. Keras is built on top of TensorFlow, an open-source numerical computation library, and uses

TensorFlow's efficient backend for computations while providing a simpler and more intuitive API for model building.

In the following subsections, the Keras implementations of each model architecture are described and explained. The different model architecture implementations are all subclasses of the class "EnergyModel". They differ only by their constructor (which sets the architecture name) and the function "build_model", which defines the Keras model.

3.4.1 Neural Network

The NN architecture consists of a sequential model composed of several dense layers. Each dense layer employs the rectified linear unit (ReLU) activation function, which introduces non-linearity into the network, allowing the model to learn complex patterns from the data. The model architecture is as follows:

```
model = Sequential()

model.add(Dense(256, kernel_initializer='normal', input_shape=input_shape,
    ↪ activation='relu'))
model.add(Dense(256, activation='relu'))  # Layer 2
model.add(Dense(256, activation='relu'))  # Layer 3
model.add(Dense(256, activation='relu'))  # Layer 4
model.add(Dense(output_shape[0], kernel_initializer='normal', activation='linear'))

model.compile(loss='mse', optimizer='adam')
```

Listing 3.1: Code of Neural Network Model

The input layer (implicitly defined) receives the raw data, represented by a two-dimensional array (*input_shape*), where *input_shape* specifies the number of samples and the number of features, respectively. The NN comprises four dense layers, each with 256 neurons. Dense layers facilitate connections between all neurons in one layer and all neurons in the subsequent layer. The ReLU activation function is applied after each layer. The final dense layer has *output_shape*[0] neurons to achieve the desired dimensionality of the output. The linear activation function is applied to ensure that the output is a real number, as expected for the load profiles.

3.4.2 Recurrent Neural Network

The RNN architecture is implemented as a 2-layer model with dropout layers. Both LSTM and GRU versions were implemented. In this section, the LSTM version is described, but the LSTM layer and the GRU layer are interchangeable. The model architecture is as follows:

```python
model = Sequential()

model.add(LSTM(64, input_shape=input_shape, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(64, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(output_shape[0]))

model.compile(loss='mse', optimizer='adam')
```

Listing 3.2: Code of Recurrent Neural Network Model with LSTM

The first LSTM layer has 64 units (memory cells) and utilizes the *return_sequences* = `True` parameter to maintain the original sequence length. This allows the input sequence to be passed through subsequent LSTM layers without being truncated. The dropout layer is a regularization technique that randomly drops a certain percentage (20 % in this case) of neurons during training. This helps prevent overfitting and improves the generalization of the model. The second LSTM layer has the same number of units as the first layer and uses the *return_sequences* = `False` parameter to indicate that the output sequence length should be reduced to one value. Another dropout layer is applied to the output of the second LSTM layer, maintaining the regularization effect. The final layer is a dense layer with the same number of neurons as the desired output dimension and a linear activation function.

3.4.3 Sequence to Sequence Model

Just as for the RNN architecture, the Seq2Seq Model was also implemented as both an LSTM and a GRU version. The LSTM implementation is based on the description in the Keras blog [Cho23] and is as follows:

```python
# Models for training
encoder_inputs = Input(shape=(None, input_shape[1]))
encoder = LSTM(64, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
encoder_states = [state_h, state_c]

decoder_inputs = Input(shape=(None, output_shape[1]))
decoder_lstm = LSTM(64, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
decoder_dense = Dense(output_shape[1], kernel_initializer='normal', activation='
    ↪ linear')
decoder_outputs = decoder_dense(decoder_outputs)

```

```
model = Model(inputs=[encoder_inputs, decoder_inputs], outputs=decoder_outputs)
model.compile(loss='mse', optimizer='adam')
self.model = model

# Models for inference
encoder_model = Model(encoder_inputs, encoder_states)
decoder_state_input_h = Input(shape=(64,))
decoder_state_input_c = Input(shape=(64,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(
    decoder_inputs, initial_state=decoder_states_inputs)
decoder_states = [state_h, state_c]
decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model(
    [decoder_inputs] + decoder_states_inputs,
    [decoder_outputs] + decoder_states)
encoder_model.compile(loss='mse', optimizer='adam')
decoder_model.compile(loss='mse', optimizer='adam')
```

Listing 3.3: Code of Seq2Seq Model with LSTM

To apply teacher forcing during the training, the inputs and connections between encoder and decoder LSTM differ between training and inference. The upper half of the Python code describes the connections for training, and the lower half describes the connections for inference. The encoder input is a vector with a length equal to the number of features. This means that all features are passed to the encoder in a single time step. Theoretically, it would also be possible that the features would be passed one after the other in several time steps. However, since there is no temporal relation between the process parameters, this would be detrimental to the performance of the model.

The encoder itself is an LSTM with 64 units. While the internal state is passed to the output because it is forwarded to the decoder, the actual encoder output is discarded. During training, the decoder takes as input the desired output sequence shifted by one (teacher forcing), the start token is the value -2. The decoder, just like the encoder, is an LSTM with 64 units which passes the internal state to the output. However, "return_sequences" is set to True to return the full sequence of outputs (instead of just the last one). This is because the output sequence of the decoder is actually used and not discarded like the output sequence of the encoder. Also, the initial internal state is set to the internal state passed by the encoder. The output sequence of the decoder is then passed to a Dense layer to process the output of the decoder. The training model takes the parameter sequence (encoder input) and the shifted output sequence (decoder input) as inputs and is trained to produce the unshifted output sequence (power profile). Both input and output training sequences must be padded to have the same length.

For inference, the input sequence can be padded but does not necessarily need to be. The output sequence can either be inferred until the model returns the first padding value or for a specified number of times (e.g. until the maximum length is reached). In the second case, a well-trained model returns padding values after the actual output sequence has been returned. Here, the second case is implemented for compatibility reasons with the other model architectures. To remove the teacher forcing, the encoder model is configured to take a parameter sequence as input and to return its internal state. This internal state is passed to the decoder as its initial state. When receiving an input, the decoder returns an output and its internal state. The output is passed to a Dense layer, and the internal state is passed back into the decoder. Because the inference process actually involves two models, a manual inference function is created which produces a prediction sequence which is as long as the maximum profile duration:

```python
def predict(self, input_data):
    # Encode and normalize input data
    input_data = self._preprocess_input_data(input_data)
    input_data = input_data[self.parameter_sequence_encoded].values.astype(np.float32)
    input_data = np.reshape(input_data, (1, 1, len(self.parameter_sequence_encoded)))

    encoder_model = self.model['encoder_model']
    decoder_model = self.model['decoder_model']

    # Generate internal state from input
    state_for_decoder = encoder_model.predict(input_data, verbose=0)

    # Start token to initiate decoder output
    target_seq = np.zeros((1, 1, 1))
    target_seq[0, 0, 0] = -2.

    prediction = []
    # Produce a prediction sequence as long as the maximum profile duration
    for i in range(self.max_time_series_length):
        load_profile_element, h, c = decoder_model.predict([target_seq] + state_for_decoder, verbose=0)
        prediction.append(load_profile_element)

        # Update target sequence and states
        target_seq = np.zeros((1, 1, 1))
        target_seq[0, 0, 0] = load_profile_element
        state_for_decoder = [h, c]

    # Reshape the prediction to remove the extra dimension
    prediction = np.reshape(prediction, (1, -1))

    # Unnormalize each prediction value with the time series scaler
    scaler = self.scalers['time_series']
    prediction = scaler.inverse_transform(prediction)

    return prediction
```

Listing 3.4: Code of Seq2Seq inference method with LSTM

After receiving the parameter sequence and passing the internal state of the encoder to the decoder, the decoder output is initiated with the start token -2. In the inference loop, after every cycle, the internal state and the last output are fed back into the decoder until the desired output sequence length (maximum process duration) is reached.

3.4.4 Parameter-Based Ensemble Learning

The ensemble model is a parameter-based ensemble model of several instances of the RNN LSTM architecture. The model is chosen based on any categorical parameter, e.g., the recipe ID, so each model specializes in the load profile forecasting of one process type only. This way, the most decisive parameter of the load profile shape does not have to be learned by the model anymore and the model can rather focus on the detailed parameter dependencies. The choice of the architecture of the single model is arbitrary, it just needs to be compared against its "single model" counterpart. The following listing presents the implementation of the parameter-based ensemble model:

```
# Input layer for all features
input_layer = Input(shape=input_shape, name='input_features')

# Split the input into recipe ID features and other features
recipe_id_features = input_layer[:, :, :self.number_of_recipe_ids]
other_features = input_layer[:, :, self.number_of_recipe_ids:]

# Create lists to store model layers for each recipe ID
model_layer_lstm1 = []
model_layer_dropout1 = []
model_layer_lstm2 = []
model_layer_dropout2 = []
model_layer_output = []

for i in range(self.number_of_recipe_ids):
    model_layer_lstm1.append(LSTM(64, return_sequences=True, name=f'
    ↪ lstm_layer1_recipe_id_{i}')(other_features))
    model_layer_dropout1.append(Dropout(0.2, name=f'dropout1_recipe_id_{i}')(model_layer_lstm1[i]))
    model_layer_lstm2.append(LSTM(64, return_sequences=False, name=f'lstm_layer2_recipe_id_{i}')(model_layer_dropout1[i]))
    model_layer_dropout2.append(Dropout(0.2, name=f'dropout2_recipe_id_{i}')(model_layer_lstm2[i]))
    model_layer_output.append(Dense(output_shape[0], name=f'output_recipe_id_{i}')(model_layer_dropout2[i]))
    model_layer_output[i] = K.expand_dims(model_layer_output[i], axis=1)

# Concatenate all the model_layer_output tensors
concatenated_output = Concatenate(axis=1)(model_layer_output)

# Reshape recipe_id_features to match the shape of concatenated_output (for later multiplication)
recipe_id_features_reshaped = Reshape(target_shape=(self.number_of_recipe_ids, 1))(recipe_id_features)

# Use Multiply layer to perform element-wise multiplication
multiplied_output = Multiply()([concatenated_output, recipe_id_features_reshaped])

# Sum along the second axis to select the appropriate output
selected_output = K.sum(multiplied_output, axis=1)

model = Model(inputs=input_layer, outputs=selected_output)
model.compile(loss='mse', optimizer='adam')
```

Listing 3.5: Code of Parameter-Based Ensemble Model with LSTM

The input layer receives the input data as a tensor, including both recipe ID features and other features. Then, the input data is split into two parts: recipe ID features and other features. Recipe ID features are used to identify the corresponding model instance, while other features are used for forecasting the load profile. In a for loop, lists are filled to store the recurrent and dense layers for each recipe ID. Layer names are assigned to simplify debugging. The model architecture for each recipe ID follows the RNN LSTM architecture described in Section 3.4.2, but only the non-recipe-ID features are passed to the input of the single models. Also, an additional layer after the final dense layer of each model expands the output tensors from each recipe ID along the first axis. This way, the list *model_layer_output* contains individual output tensors for each recipe ID, which can be concatenated in the next layer to form one big tensor of the shape $(number\_of\_recipe\_ids, length\_of\_output\_profile)$ for the output of all models. The tensor with the recipe IDs is also reshaped, so both tensors can be multiplied. The recipe ID tensor is one-hot-encoded, i.e., the column of one categorical parameter is converted to one column for each value of this parameter, and the columns of the values are set to "0" or "1" depending on the categorical parameter value of the respective line. Due to the one-hot encoding, this multiplication isolates the output vector of the corresponding recipe ID, i.e., all other tensor elements are multiplied with zero. The summation along the first axis, therefore, produces the output vector of the corresponding recipe ID in the desired shape.

3.5 Evaluation Metrics

To assess the performance of the different model architectures, suitable metrics have to be defined before conducting the experiment. However, the model performance also depends on the training data. To test the model performance in different scenarios with different training data, metrics to assess the complexity of training data have to be determined as well. This section introduces and discusses various metrics for evaluating the performance of the different model architectures and the complexity of the data to be predicted.

3.5.1 Classification of Training Data

The most obvious metric for the training data is the number of samples. Since the data is augmented with Gaussian noise, the augmentation factor and the standard deviation of the noise are also relevant. Another relevant metric is the number of values to be predicted. It is defined by the process duration divided by the defined temporal resolution and can be directly deducted from the dataset.

Correlations between process parameters and the power profile are measured with the distance correlation coefficient for numerical parameters and the correlation ratio for categorical parameters. The distance correlation coefficient $dCor$ measures the dependence between two variables by analyzing the distances between data points in each variable. Unlike Pearson correlation, it does not assume a linear relationship and can effectively identify more complex patterns [Szé+07]. However, it is more computationally expensive. The calculation of $dCor$ involves calculating the distances between all pairs of data points for each variable, centering the data to remove the influence of overall shifts in the data, and comparing the variations in distances between data points of one variable with the variations in distances between data points of the other variable. For details about the calculation of $dCor$ refer to [Szé+07]. The values of $dCor$ range from 0 to 1 and indicate the strength of a linear or non-linear relationship, higher values suggest a stronger connection.

Since the categorical parameters do not have an order, their correlation with the power profiles must be measured with the correlation ratio η, which compares the variance of the numerical variable within each category of the categorical variable to the variance over the whole dataset [Ric11]:

$$\eta = \sqrt{\frac{\sum_x n_x(\overline{y}_x - \overline{y})^2}{\sum_{x,i}(y_{xi} - \overline{y})^2}} \tag{3.7}$$

In this equation, x is the specific category of the categorical parameter and i is the sample. A greater correlation ratio indicates that the variance within categories of the numerical variable is smaller than over the whole dataset, suggesting a stronger association between the categories and the numerical variable.

In order to quantify the distance between values in the test dataset and values in the training and validation dataset for a continuous numerical parameter, a relative average distance (RAD) is introduced, a metric which provides insights into the interpolation difficulty and gaps present in the parameter space. It is defined by the average absolute difference between each parameter value in the test dataset and the closest value of the same parameter in the training or validation dataset, i.e., the average "gap" between the test and the training/validation dataset, divided by the range of parameter values:

$$\text{RAD} = \frac{\frac{1}{N}\sum_{i=1}^{N} \min_{j \in M} |x_i - y_j|}{p_{max} - p_{min}} \tag{3.8}$$

where:

- N is the total number of data points in the test dataset
- M is the set of indices for training and validation data points
- x_i is the i-th parameter value in the test dataset
- y_j is the j-th parameter value in the training or validation dataset (whichever is closer to x_i)
- p_{min} and p_{max} are the minimum and maximum value of the respective parameter

By employing the average absolute difference instead of the median, the RAD metric gives more weight to potential outliers in the parameter space. This choice is intentional to highlight situations with sparse data coverage, even if a majority of test points have good coverage. Such cases can pose significant challenges for machine-learning models in interpolation tasks.

The pendant of the RAD for several dimensions (i.e., parameters) is the average minimal Euclidean distance $\overline{d_{\text{min}}}$ of each test data sample to any training or validation data sample, which is calculated as follows:

$$\overline{d_{\text{min}}} = \frac{1}{N}\sum_{i=1}^{N} \min_{j=1}^{M} \sqrt{\sum_{k=1}^{D}(x_i^k - y_j^k)^2} \tag{3.9}$$

In this formula:

- N is the number of test data samples.
- M is the number of training or validation data samples.
- D is the number of dimensions (parameters).
- x_i^k represents the k-th dimension of the i-th test data sample.
- y_j^k represents the k-th dimension of the j-th training or validation data sample.

To compare $\overline{d_{\min}}$ for scenarios with a different number of dimensions, it has to be normed by the number of dimensions:

$$\|\overline{d_{\min}}\| = \frac{1}{\sqrt{N}}\overline{d_{\min}} \tag{3.10}$$

This metric can be used to quantify the distance of test samples from training and validation samples and, hence, the interpolation difficulty of a particular dataset for the model.

The complexity of the dataset regarding the included parameter dependencies is difficult to quantify. In this study, the approach will be to list the number of dependencies of each kind as listed in the dependency configuration file of a dataset. However, there is no metric for which kind of dependency introduces the most complexity to the data, it can only be deduced from the resulting MSEs of models that were trained with datasets with different kinds of dependencies.

3.5.2 Model Performance Assessment

On the one hand, the model performance can be assessed with the duration of training (in seconds). On the other hand, given the numerical nature of the data to be predicted, the prediction quality can be measured with the MSE on the validation dataset:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.11}$$

where n is the number of samples in the validation dataset, y_i is the true value of the target variable for the i-th sample, and $\hat{y}_i$ is the predicted value of the target variable for the i-th sample.

Chapter 4

Experimental Setup

This chapter describes the setup to prove the concept of the Energy Model and to compare different architectures against each other. First, the use case of AVL's Battery Innovation Center (BIC) with its manufacturing processes is presented, followed by the training data acquisition through real data from AVL's BIC and through a synthetic data generator. Subsequently, the data preparation methods for effective and efficient training are outlined, and finally, the parameters for training the models are detailed.

4.1 Use Case 1: AVL Battery Innovation Center

AVL is the world's largest private and independent company specializing in developing propulsion systems with internal combustion engines, as well as measurement and testing technology. The Powertrain Engineering division focuses on the research, design, and development of various propulsion systems aimed at reducing fuel consumption, pollutant emissions, and noise while enhancing vehicle performance. An integral part of this division is the Battery Innovation Center (BIC) (overview shown in Figure 4.1), which offers customers to assemble battery cells to battery packs, from one-off prototypes to runs of several hundred pieces, with comprehensive monitoring of defined quality characteristics [AVL]. The BIC is used for research and development by advancing production engineering, gripper development, production logistics, quality assurance processes, and cell handling investigations. It adopts a Twin Factory approach, utilizing digital twins for both product and process and a function-oriented process control. This also includes energy metering of all machines and tracking single product parts with bar codes, making it possible to assign energy consumption and CO_2 emissions to products. The production line follows the matrix production approach and is capable of processing three different cell types (cylindrical, pouch, and prismatic). It comprises the stations stacking (assembling individual cells

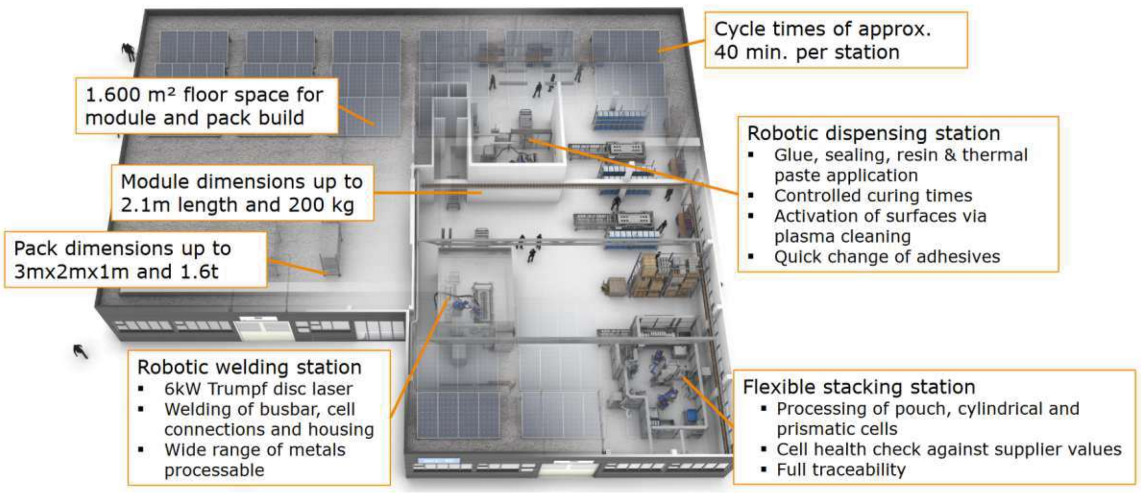

Cycle times of approx. 40 min. per station
1.600 m² floor space for module and pack build
Robotic dispensing station
▪ Glue, sealing, resin & thermal paste application
▪ Controlled curing times
▪ Activation of surfaces via plasma cleaning
▪ Quick change of adhesives
Module dimensions up to 2.1m length and 200 kg
Pack dimensions up to 3mx2mx1m and 1.6t
Robotic welding station
▪ 6kW Trumpf disc laser
▪ Welding of busbar, cell connections and housing
▪ Wide range of metals processable
Flexible stacking station
▪ Processing of pouch, cylindrical and prismatic cells
▪ Cell health check against supplier values
▪ Full traceability

Figure 4.1: Overview of AVL's BIC [Mei+23].

into a configured arrangement), welding (laser welding), gluing, packing (arranging individual battery cells within a casing), and quality control. Apart from quality control, all steps are carried out by robotic stations. While the BIC does not offer serial production itself, the speed of the individual production steps corresponds to large-scale production of 10,000 pieces/year [AVL], making it suitable for process optimization of serial production processes.

4.1.1 AVL Energy Data

Energy data from AVL's BIC are available for the gluing station (11th December 2021–19th November 2022), the stacking station (1st February 2022–11th November 2022) and the welding station (1st February 2022–8th February 2022 and 13th May 2022–20th May 2022). The times with missing energy data are caused by development works on the data processing and the machines themself, since the BIC was inaugurated only in September 2021.

The Energy Data is an extract from a Structured Query Language (SQL) database, exported to an Excel file by AVL and converted to comma-separated values (CSV) for compatibility reasons. It contains as columns the machine name, the active and reactive power of all three phases separately, the energy meter readings of active and reactive energy in- and outflow (all three phases combined), the longitudinal and transverse voltages, the currents of the single phases, the cos(ϕ) of the phases, and the timestamp. Data points are recorded at intervals of 20 seconds.

The process data are distributed across two files, "pse_segment" and "pse_pro". The file "pse_segment" contains the columns described in Table 4.1, and the columns of "pse_pro" are described in Table 4.2. There is a third table called "pse_rule" (the "id_pse_rule" columns in "pse_segment" and "pse_pro" reference to this table) which is not available and which defines

Table 4.1: Columns of the file “pse_segment”.

Column Name	Description
id	Identity
id_pse_rule	Identity of table pse_rule
name	Station name
group_name	Sub-Station Name
starttime	Processing start time
endtime	Processing end time
s_state	Status of the line (1= latest entry, 2= same record with newer date exists)

Table 4.2: Columns of the file “pse_pro”.

Column Name	Description
id	Identity
id_pse_rule	Identity of table pse_rule
id_pse_segment	Identity of table pse_segment
name	Variable name (also includes start and end)
value	Value
value_type	Data type of value

each component that has been processed in the system (see Table 4.3). In essence, “pse_rule” defines a whole component, “pse_segment” tracks it across the different stations and substations, and “pse_pro” records measured values for the respective sub-stations.

To get a process data overlay over the energy data like in Figure 3.1, the start and end times of a “pse_rule” (i.e., the first and last “pse_segment” with the same “pse_rule_id”) are relevant, since they mark the timestamps of a component being processed at a certain station. Processing timestamps at sub-stations (in “pse_segment”) would be an alternative, however, energy is only measured at a station level and therefore cannot be traced to the single sub-stations. To get the process parameters (mainly from “pse_pro”), the lines of “pse_pro” and “pse_segment” have to be linked along the column “id_pse_segment”. Relevant process parameters for the stacking station include duration (starttime - endtime in “pse_segment”), recipe, voltage, force_1_N, force_2_N, position and klt_position (variables from “pse_pro”). “Voltage” is the voltage measured when testing the battery pack after stacking, “position” is the position in the batch, and “klt_position” is the position of the small load carrier (“Kleinladungsträger” - KLT in German) containing the parts. Relevant process parameters for the gluing station include duration (starttime - endtime in “pse_segment”), contour_number, recipe_nr_glue, and glue_repetitions

Table 4.3: Columns of the file “pse_rule”.

Column Name	**Description**
id	Identity
rule_code	Serial number of the component (cell, cartridge, stack, etc...)
starttime	Processing start time
endtime	End time of processing (is only set after all sub-stations have been completed)
data_version	Change index of the data of the component (is increased by 1 with each writing action)

(variables from “pse_pro”). Depending on how interdependent the processes are with regard to their load profiles, possibly also the parameters of the preceding process can be included.

To assess the interdependency of processes, the similarity of the load profiles of processes with the same parameters can be examined. If the load profiles of processes with the same parameters differ more than a certain threshold, it indicates that the load profile of the process is influenced also by the preceding process. This method, of course, requires that all relevant parameters are captured and included in the parameter list. Otherwise, the difference in load profiles could also derive from such an unobserved parameter.

Since the “endtime” column of “pse_segment” is NULL in most cases, the end time of a “pse_rule” has to be derived from the “endtime” parameter of the last “pse_segment” belonging to the respective id_pse_rule to be able to calculate the duration of a process. Also, the start and end times of a process are needed to assign certain parts of the load measurement to the process. Since typically several components are processed simultaneously at the single sub-stations in the BIC, overlapping time spans occur. In this case, the power consumption of the respective overlapping period is distributed equally across the involved processes.

The data processing is shown in the example of the stacking station. After combining the process parameters from the two files “pse_segment” and “pse_pro” as described before, a table with one row per pse_rule_id and the columns “station”, “pse_rule_id”, “starttime”, “endtime”, “duration”, “rejected”, “order_number”, “recipe”, “position”, “force_1”, “force_2”, “voltage”, and “klt_pos”. The duration is calculated by subtracting the start time from the end time and the parameter “rejected” is a boolean value which is set based on the condition that a pse_segment with the name “REJECT” exists. The reason for a reject is most often too low a measured voltage at the test substation. The other values are taken directly from the file “pse_pro”.

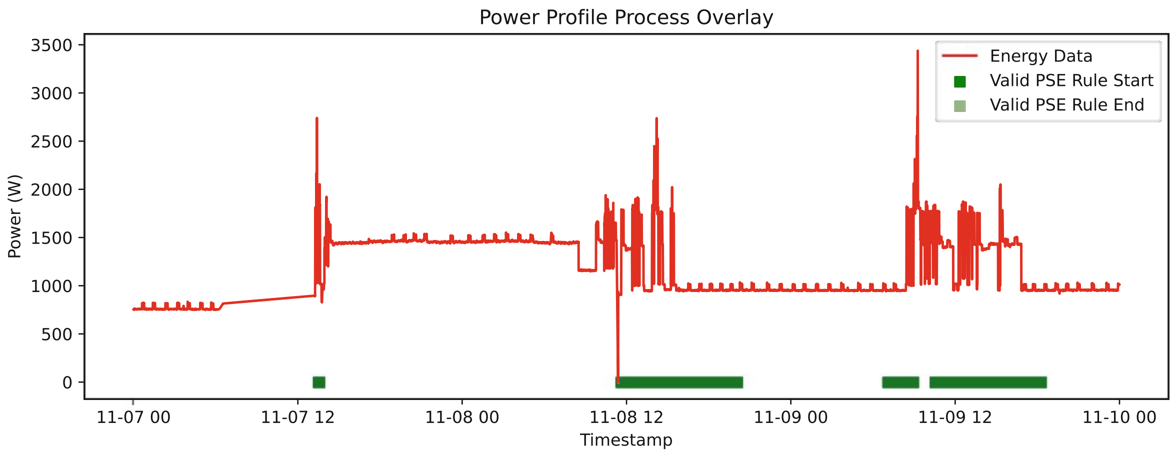


Figure 4.2: Load profile of stacking station with process data overlay.

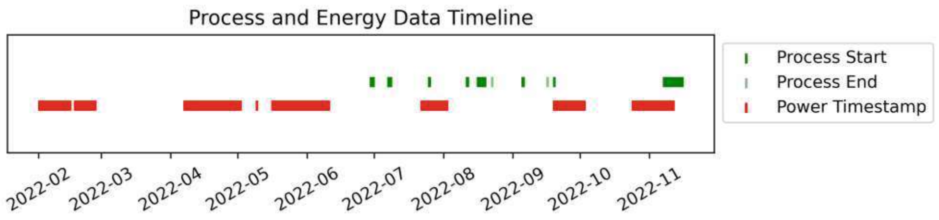
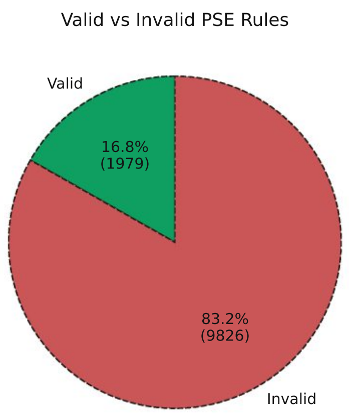


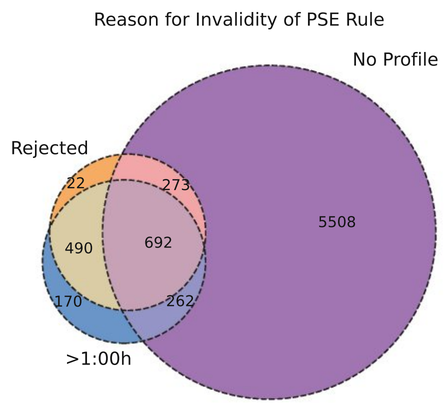
Figure 4.3: Timeline of process data and energy data.

Figure 4.2 illustrates how certain parts of the power profile are assigned to pse_rule_ids (hereafter called processes) with their start and end times. The power profiles are calculated by adding the phase power of all three phases from the energy data and also include equidistant time stamps. Figure 4.3 shows for which time spans energy data and process data are available. It becomes apparent that only in a limited time span process data as well as energy data exist. This is one reason for which the process data is required to be filtered to be useful (analyzed in Figure 4.4), the other two reasons being that the process output was rejected (this means that no process parameters are saved) and that the process was started on one day and finished on another day, indicated by a process duration greater than 1 hour (otherwise, also idle energy consumption during the night would be included in the process power profile). The share of valid and invalid process data is shown in Figure 4.4(a), and the reasons for the invalidity of process data are depicted in Figure 4.4(b). Missing energy data is the major reason for the high number of invalid process data in the data set, followed by too long processes and rejected processes. Interestingly, it is hardly the sole reason for the invalidity that the process was rejected or that it was too long; these two phenomena occur mostly in combination with other reasons.

Figure 4.5 shows the analysis of the distribution of process parameters across the valid process data in the form of boxplots. The energy-relevant process parameters include recipe, force_2,



(a) Share of valid and invalid process data.



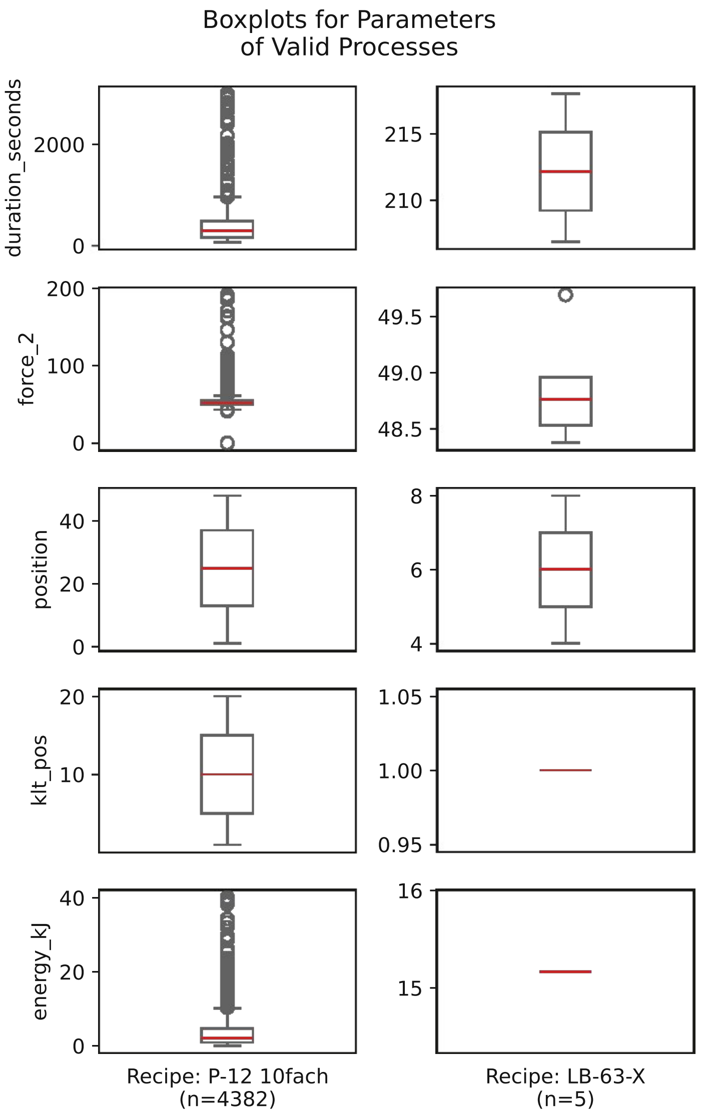
(b) Reasons for invalidity of process data.

Figure 4.4: Analysis of valid and invalid process data.

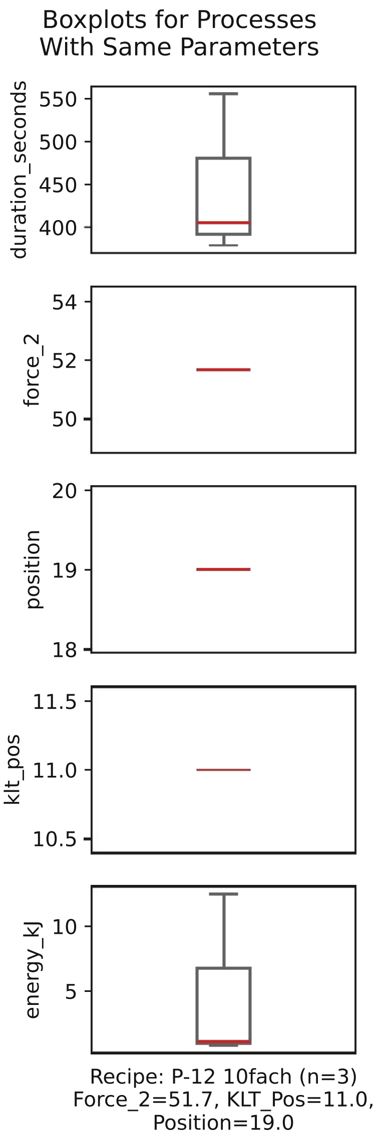
position, and klt_pos. Not included are force_1 (always 0) and voltage (measured voltage during testing, therefore not known a priori). The boxplots in Figure 4.5(a) show the distribution of all parameters across the valid data and are grouped by the two types of recipes that occurred in the process data, "P-12 10fach" and "LB-63-X". While recipe "LB-63-X" only has 5 valid samples and is therefore not representative, the wide range of the parameter values for recipe "P-12 10fach" becomes apparent. To analyze how repeatable, i.e. predictable, the duration and energy consumption of a process are, samples with the same parameter values are analyzed. Figure 4.5(b) shows the distribution of the process duration and energy consumption for recipe "P-12 10fach" with identical other parameter values ($\text{force\_2} = 2, \text{position} = 19, \text{klt\_position} = 11$). Despite these identical parameter values, there is still a considerable range in the duration (ca. 40 % of the minimum value) and the power consumption (ca. factor 3 of the minimum value). The low sample number of 3 samples in Figure 4.5(b) does not undermine the conclusion that if there is such a range in the energy consumption, there must be parameters affecting the energy consumption that are not captured. AVL's energy data is, therefore, not directly usable as training data for the Energy Model.

4.1.2 Synthetic Energy Data Generation

The ability of the Energy Model to accurately predict energy consumption patterns is directly linked to the complexity of the training data. Even with real data available, the limits of the Energy Model with regard to the complexity of the training data need to be tested. The complexity of the training data results from the number and nature (linear/non-linear) of relations



(a) Distribution of all valid process parameters.



(b) Range of duration and energy consumption despite same parameters.

Figure 4.5: Distribution of valid process parameters.

between process parameters and the resulting load profile, as well as from the number of data points to be predicted. The number of data points to be predicted, i.e., the length of the load profile, depends on the maximum process duration on the one hand and on the defined temporal resolution on the other hand. To assess the model's performance under varying data complexity scenarios, a data generator is developed to produce synthetic datasets.

The foundation for all generated data is a predefined factory setup, consisting of a certain machine and production step configuration, which is based on AVL's BIC. The machine configuration is represented by a file that details the available amount of each machine type, and the production step configuration is represented by a so-called recipes file that outlines the step ID, the step name, the product ID, the required machine type, and operator ID for this step, the duration, and the prerequisites, i.e., the production steps, which need to be completed before the respective step. Additionally, supplementary parameters with their types (numerical/categorical, discrete/continuous) and their ranges are given.

The data generator first generates a specified number of parameter sets that are plausible according to the machine file and recipe file. The distribution of the parameter values is random. A special case is the idle consumption of machines, which is represented by its own recipe ID but not mentioned in the recipe file. The duration can be anything up to the duration of the longest process; the previous recipe ID (possibly also idle) depends, just like for the other recipe IDs, on the machine type and which other processes are possibly executed on it. In the second step, load profiles for the given parameter sets are generated. The basis for the load profiles is a profile shape file that indicates the production step ID and name, the basic shape of the load profile as a list of equidistant points, the interpolation method (padding/nearest neighbor), and dependencies on parameters.

There are four possible ways in which a parameter can influence the load profile:

- Height dependency: a factor by which the load profile is multiplied (stretch along the y-axis).
- Offset dependency: a constant offset that is added to the profile (shift along the y-axis).
- Repetition dependency: the profile repeats itself for a number of cycles.
- Profile dependency: the shape of the profile itself depends on a parameter.

Theoretically, there is also a fifth possibility, which is that a parameter could influence the duration of a production step. However, manufacturing planning requires the duration to be known a priori; and in the recipe representation, a different duration would result in a separate recipe ID. Therefore, this kind of dependency can be ignored.

Dependencies are represented as a list of Python dictionaries that have two or three keys: the parameter name and either a so-called "factor_dict", i.e., another dictionary with parameter values and the corresponding value of the dependency (mainly for categorical parameters), or a "min_factor" and "max_factor" which are linearly assigned to the whole observed range of a numerical parameter in the parameter set. The effect of the factor depends on the column of the dependency, e.g., a factor of an offset dependency would indicate the offset that is added to the basic profile shape, whereas the factor of a repetition dependency would indicate the number of repetitions of the basic profile shape. Profile dependencies can only contain a "factor_dict", which defines the basic profile shape depending on a certain parameter.

Based on the parameters of each parameter set, the basic profile shape is selected and the relevant dependencies are applied. After the dependency application, the synthetic load profile is stretched or compressed to the desired duration as indicated by the parameter set. This duration is given as a multiple of the defined temporal resolution, i.e., the number of data points of the load profile. Stretching is conducted by evenly inserting Not a Number (NaN) values in the profile and applying the indicated interpolation method while compressing is conducted by evenly deleting data points of the load profile. It is possible that, due to compressing the load profile to the desired length, certain characteristics of the load profile are not as clearly visible anymore, e.g., if you compress a profile with 10 repetitions to a length of 7, the whole number of repetitions is not visible anymore. This effect is not unique to the synthetic data but can also occur in real data (caused by a low sampling rate of energy measurements), it is an effect of the chosen temporal resolution. After the creation of the final load profile, the profile is padded with a value of -1 (which cannot be mistaken for a regular load profile value) to ensure that the load profile has a consistent length and can be processed seamlessly by the model.

Despite the synthetic nature of the generated data, it should have some relation to the real data from AVL's BIC. Therefore, the sequence of process steps, their duration, and the profile shape are based on real energy data. The configuration of the sequence of process steps and their duration is listed in Table 4.4. The names, sequence, and duration of the process steps are taken directly from the real use case, as well as the additional parameter "batch size". An exception to this is the transports between the stations, which in reality are done manually while they are simulated as Automated Guided Vehicles (AGVs) in the energy data which need to be charged after each transport. The parameter "welding temperature" for laser welding is introduced to get a continuous numerical parameter that affects the power profile. The parameter value range is chosen on the basis of melting points of materials that can be processed with laser welding [Dun21]. Batch size and welding temperature are applicable only to a subset of recipe IDs. The respective parameter column for instances of other recipe IDs is therefore filled with the invalid value “-1”. Also, a column “has_parameter_[. . .]” is created for these parameters to simplify training. This column is either “0” or “1”, depending on whether the parameter exists for the recipe ID of the respective instance.

The duration of the process steps varies between 15 and 45 minutes. Given a defined temporal resolution of 1 minute, this means that the load profiles have a length of 15 to 45. While the durations of normal process steps are multiples of 15 minutes, the duration of the idle time of a machine in between two process steps is arbitrary. However, to adhere to the maximum process duration, idle times that are longer than the maximum process duration are split into several process steps in the training data and also later when inferring load profiles from a schedule.

The machine list as compared to the real BIC, which only has one machine of each type, is extended by several instances of most machine types (see Table 4.5) to be able to simulate load profile dependencies on the machine instance, e.g., older machines which consume more energy than modern ones.

The basic profiles and parameter dependencies are detailed in Appendix A, Tables A.1–A.4. They comprise the basic profiles and interpolation methods as well as height, repetition, and profile dependencies. Offset dependencies are not defined. The basic profiles (listed in Table A.1) are created based on the shapes and values visible in AVL’s real energy data. These shapes can be described as stair steps of different levels with occasional spikes in them. In any case, no complex forms, such as sawtooth functions or polynomial functions, are visible in the real energy data.

The parameter dependencies are defined without regard to the real energy data. The only purpose is to test the capabilities and limits of the Energy Model to capture such possible dependencies. Nevertheless, every dependency is defined with a plausible cause and effect in mind. For all transport processes, height dependencies are defined with the preceding recipe ID as the parameter, simulating that the AGVs do not charge after every transport and that the charging profile also depends on the preceding transport in this case. Height dependencies are also defined for welding (depending on the welding temperature), gluing (depending on the preceding duration, simulating possible necessary warm-up of the glue), and idle time (depending on the machine). Repetition dependencies are defined for stacking, welding, gluing, packing, and the quality check with the batch size as the parameter. Profile dependencies are only defined for idle time with the machine as the parameter, simulating different times until standby.

To test the Energy Model on training data of varying complexity, data sets with different combinations of the mentioned parameter dependencies are generated. The other decisive factor

Table 4.4: Recipe configuration for the synthetic data generation.

id	**product_ type_id**	**step_name**	**machine_ type_id**	**operator_ type_id**	**duration**	**prerequisites**	**param_ numerical_ batch_ size**	**param_ numerical_ welding_ temperature**
0	0	stacking	0	0	00:45:00	[]	{'type': 'discrete', 'min': 1, 'max': 5}	
1	0	stacking_ to_welding	5	2	00:15:00	[0]	1	
2	0	welding	1	0	00:30:00	[1]	{'type': 'discrete', 'min': 1, 'max': 5}	{'type': 'continuous', 'min': 500, 'max': 1500}
3	0	welding_ to_gluing	5	2	00:15:00	[2]	1	
4	0	gluing	2	0	00:30:00	[3]	{'type': 'discrete', 'min': 1, 'max': 5}	
5	0	gluing_ to_packing	5	2	00:15:00	[4]	1	
6	0	packing	3	0	00:30:00	[5]	{'type': 'discrete', 'min': 1, 'max': 5}	
7	0	packing_ to_quality	5	2	00:15:00	[6]	1	
8	0	quality	4	1	00:15:00	[7]	{'type': 'discrete', 'min': 1, 'max': 5}	

Table 4.5: Machine Configurations

machine_id	**machine_type**	**amount**
0	stacking_robot	6
1	welding_robot	4
2	gluing_robot	4
3	packing_robot	4
4	testing_station	2
5	trolley	8

for the complexity of the training data is the number of predicted data points, i.e., the duration of the process steps (assuming a fixed temporal resolution). Even though the durations vary across the recipe IDs anyway, the Energy Model is not inherently tested on training data with diverse load profile lengths since load profiles of different lengths are padded to have the same length. This means that datasets with varying load profile temporal resolutions have to be created to assess the influence of the load profile length. In the Ensemble Learning LSTM architecture, the number of inner models for the experiments based on the synthetic data is determined by the number of recipe IDs, so there is one model for each recipe ID.

4.2 Use Case 2: Industrial Robot

The described Energy Model works best for fully automated processes due to their regular patterns and clearly defined parameters. IRs play a significant role in automating production lines and are, therefore, a good example to test the Energy Model on. In [Gad+21], the authors present a series of experiments that record the energy consumption of an IR when performing different test procedures. The results are published as an open research dataset [Gad+20]. The following use case description and data analyses are based on [Gad+21].

4.2.1 Measurement Setup

The IR model used in the series of experiments is the “KUKA KR210 R2700 Prime”, a six-axis industrial robot with a weight of 111 kg and a high payload capacity of up to 210 kg (see Figure 4.6). The exact experimental setup is detailed in [Gad+21]; in the following, just the most important specifications are described. The supplied voltage and current to the robot are measured on all three lines, and the electrical power is calculated by multiplying the voltage with the current on each line and adding the resulting power flow of each line. The raw measurement has a high sample rate of 40 kHz, which is necessary to capture the pulsating power flow through the diode rectifier. To reduce the amount of data and required computational effort, the raw

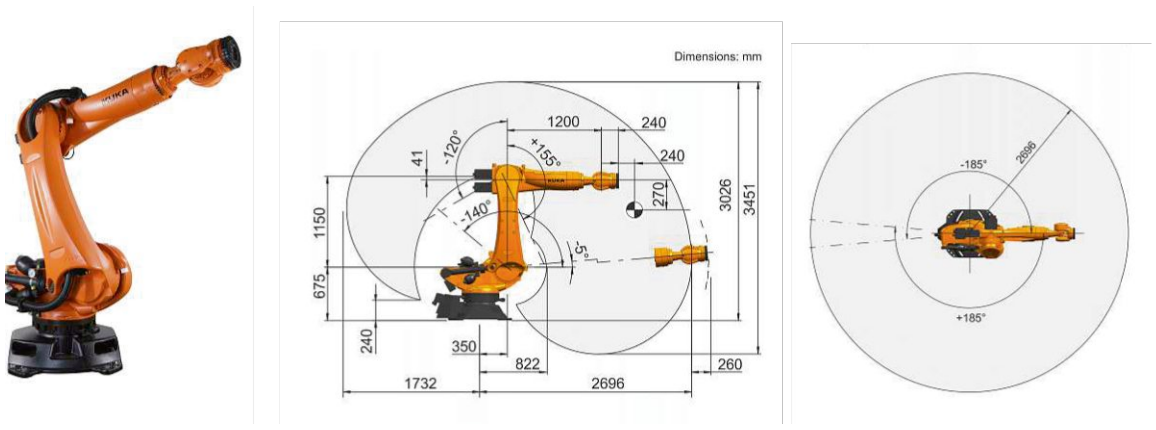


Figure 4.6: KUKA KR210 R2700 Prime (from data sheet [KUKA20]).

power data is post-processed to filter the impulsive effect (without changing the amount of energy) and to reduce the sample rate to 300 Hz. In addition to the power measurements, mechanical, electrical, and thermal quantities of the robot are recorded with monitoring software provided by KUKA. However, this data can be discarded for testing the Energy Model since it is not a parameter of the operation (and therefore known in advance) but only recorded during runtime. Since the energy consumption of the IR is affected by the temperature of the oil in the reducers, which is linked to the viscosity of the oil, the oil temperature is recorded with a thermocouple.

4.2.2 Experiment Series Execution

Each experiment is defined by 7 input parameters: number of axes involved, type of motion – Point-to-Point (PtP) or linear, velocity parameter, acceleration parameter, brake closing delay, payload applied to the end-effector, and system temperature. There are three types of experiment series conducted, each varying different parameter values during the series:

A) Concurrent motion of 6 axes at quasi-constant temperature
B) Motion of axis 1 at quasi-constant temperature
C) Motion of axis 1 at increasing temperature

The parameter value sets are shown in Table 4.6. Within an experiment series, two experiments were conducted for each possible parameter value combination, one from the defined starting point to the endpoint and one in the reverse direction. While linear motions are defined in the Cartesian space, PtP motions are defined in the joint space, i.e., the movement of each joint is defined. The brake closing delay, also known as brake reaction time, refers to the time it takes for the brakes to fully engage after receiving a signal to do so. This delay can have a significant impact on the energy consumption of a system since the IR continues to consume energy for

Table 4.6: Experiment parameter value sets.

Experiment Series	Axes Involved	Motion Type	Brake Closing Delay	Payload Type
A	6	[linear, PtP]	$[1, 20]$ s	Load $[1, 2, 3]$
B	1	PtP	10 s	Load $[1, 2, 3]$
C	1	PtP	10 s	Load $[1, 2, 3]$

Experiment Series	Acceleration Parameter	Temperature
A	$[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$ m/s^2 (linear) $[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$ % (joint)	~ 45 °C
B	$[1, 2, 5, 10, 25, 50, 75, 100]$ %	~ 45 °C
C	50 %	$[22.0, \ldots, 49.7]$ °C

Experiment Series	Velocity Parameter
A	$[0.17, 0.34, 0.51, 0.68, 0.85, 1.02, 1.19, 1.36, 1.53, 1.70]$ m/s (linear) $[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$ % (joint)
B	$[1, 2, 5, 10, 25, 50, 75, 100]$ %
C	50 %

movement even though it is decelerating. The acceleration and velocity parameter units also differ between these motion types: the parameters for linear motions are given in m/s and m/s^2, respectively, and the parameters for PtP motions are given in % of the maximum allowable values. All three experiment series were conducted with three types of payload: no external load (load 1), a beam weighing 66 kg (load 2), and a beam weighing 133 kg (load 3). For the details of the center of mass, the orientation of the principal inertia axis in the center of mass, and the principal mass moment of inertia, please refer to [Gad+21].

To keep a quasi-constant temperature in experiment series A and B, a cyclic motion was performed in a warm-up phase (taking several hours) until the system temperature (both oil and actuators) stabilized. Then, one experiment with a specific set of parameter values was conducted. After each experiment, the system temperature was brought back to the original value with a short warm-up (taking approximately 1 minute) to mitigate the influence on the temperature of the experiments themself.

The experiment data are published as a Matlab file. For further use, the relevant data is extracted and converted to a CSV file with Python. The columns of the CSV file for every experiment series are shown in Table 4.7. The interval between the energy measurements is reduced by calculating the average power in the respective time frame. Different versions with different sampling rates are created for each experiment to evaluate the influence of the output vector length on the forecast accuracy.

Table 4.7: Columns of the CSV files with the extracted data for each experiment series.

Experiment Series A	Experiment Series B	Experiment Series C
motion_direction	motion_direction	motion_direction
load_type	load_type	load_type
velocity	velocity	velocity
acceleration	acceleration	acceleration
duration	duration	duration
motion_type	oil_temperature	oil_temperature
brake_delay		
time_series_value_0 . . . x	time_series_value_0 . . . x	time_series_value_0 . . . x

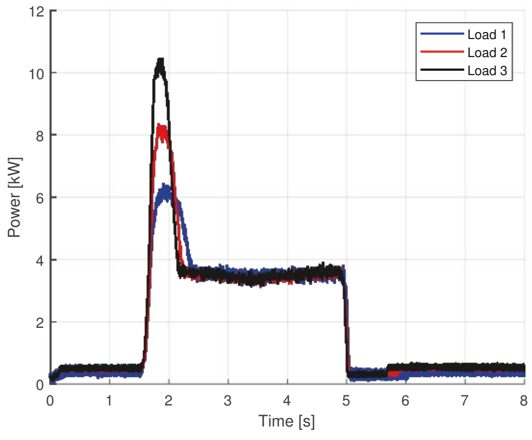
4.2.3 Parameter Influence on Energy Consumption

The authors of [Gad+21] also provide several analyses of parameter influences on the energy consumption and power profile of the motion execution. The graphics are replicated and shown in Figure 4.7. In Figures 4.7(a) and 4.7(b), three different power profiles of motions with varying payloads for a 1-axis motion and a 6-axes motion, respectively, are shown. It is visible that the power peaks, caused by the acceleration, are far more dependent on the load type than the more constant parts of the power profile, associated with constant movement.

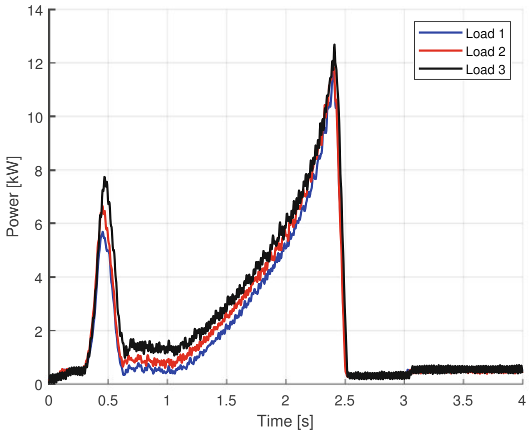
In Figures 4.7(c) and 4.7(d), the influences of varying velocity and acceleration parameter values, respectively, on the power profile are shown. A greater velocity parameter value increases the power peak as well as the more constant part but shortens the duration of the motion. A greater acceleration parameter value, on the other hand, also increases the power peak but shortens only the duration of the power peak, while the duration of the complete motion remains almost unchanged.

In Figure 4.7(e), power profiles of a 1-axis motion with varying oil temperatures are depicted. With increasing oil temperature, the whole power profile is compressed by an almost constant factor. However, the higher the oil temperature, the smaller the marginal decrease in power consumption. The reason for this behavior can be seen in Figure 4.7(f), which shows the simultaneous development of the oil temperature and the measured friction torque over several experiments of experiment series C. With increasing oil temperature, the friction torque decreases significantly due to the decreasing oil viscosity.

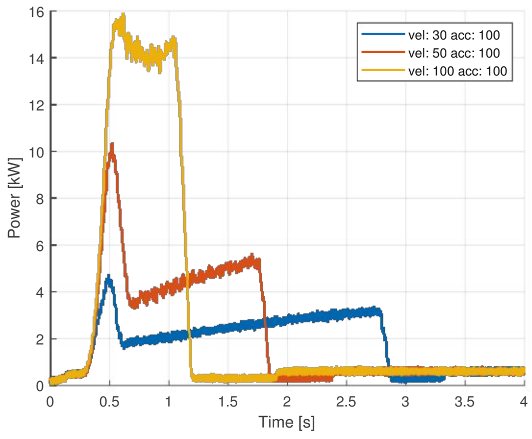
In [Gad+21], the authors also show that a brake delay of 1 s compared to 20 s can lead to energy savings of up to 25 %, depending on the other motion parameters. In summary, it can be observed that each of the captured parameters from Table 4.7 (except for the motion direction) has a characteristic influence on the performance profile of the robot motion which the Energy



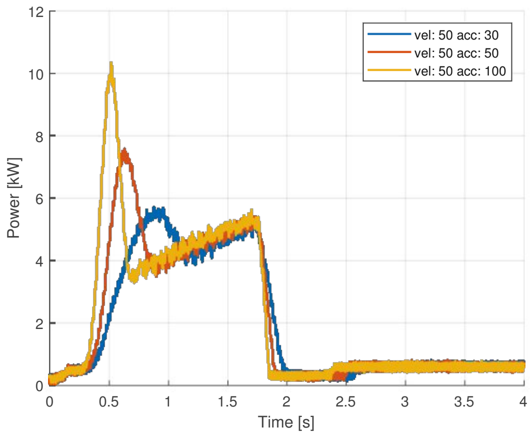
(a) Influence of payload on power profile (1 axis motion).



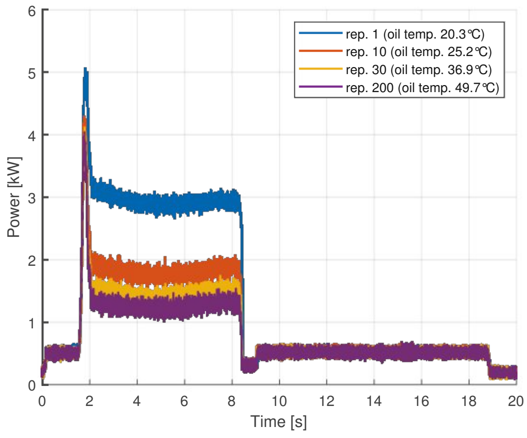
(b) Influence of payload on power profile (6 axes motion).



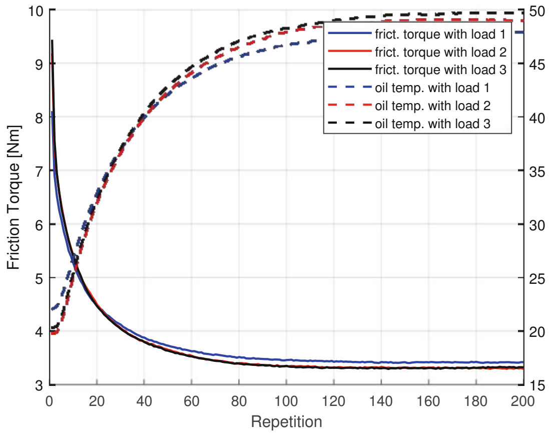
(c) Influence of velocity on power profile.



(d) Influence of acceleration on power profile.



(e) Influence of system temperature on power profile.



(f) Influence of oil temperature on friction torque.

Figure 4.7: Influence of various parameters on the power profile of the IR motion.

Model must be able to model. The number of inner models in the Ensemble Learning LSTM architecture for the experiments based on the IR dataset is determined by the number of load types, i.e., there is one model for each load type.

4.3 Data Preprocessing

To improve the performance of the Energy Model architectures, a range of preprocessing methods are applied to the training data before training the architectures. To distinguish between input and output values, the data columns have a prefix of either "parameter_" or "time_series_value_". To further distinguish between numerical and categorical input parameters (the output/load profile values are only numerical), the parameter columns have another prefix of either "numerical_" or "categorical_".

First, the datasets are divided into a training, a test, and a validation dataset with a share of 70/20/10 %. This way, overfitting is prevented by evaluating the models already during training on a different dataset (the test dataset) and evaluating the final model on yet another dataset (the validation dataset). When splitting the dataset, special attention is put on not having two identical parameter vectors (input) in two different datasets. Also, some numerical parameter value ranges of the welding temperature are excluded from the training and test data and only included in the validation dataset, to be able to assess the capability of the different architectures to cope with unknown parameter values by interpolating between known parameter values.

Then, the training dataset is augmented to some extent by adding Gaussian noise with a standard deviation of 5 W to the output values, i.e., the load profile values. The input values, i.e. the process parameter values, remain unchanged. This way, scarce training data can be multiplied without risking too much overfitting.

The categorical process parameters (recognizable by their column name prefix) are encoded with one-hot-encoding to make them processable for the Energy Model architectures. Furthermore, since many machine learning algorithms, especially those involving optimization techniques like gradient descent, converge faster when the features are on a similar scale, the numerical values (both the numerical process parameters and the load profile values) are scaled with a MinMax scaler to the interval $[-1, 1]$.

Finally, to ensure that the data is presented in a format that the model can effectively process, the input vectors are reshaped to the suitable dimensions. The shape follows the format [batch size, number of time steps, input features]. Since there is no temporal relation between the input parameters, the final shape is [number of samples, 1, number of process parameters].

4.4 Training Parameters

The Energy Models are trained on nodes of the VSC with two AMD EPYC 7713 (Milan) processors (64 cores operating at 2GHz and 512 GB memory) and two NVIDIA A100 Graphics Processing Units (GPUs) (40GB memory) [VSC5]. To prevent overfitting, the MSE is used as the loss function; the stopping condition is a minimal improvement of the validation MSE of 10^{-6} (on the scaled data), with the patience set to 30 epochs. The maximum number of epochs is set to 500. To update the weights during training, the Adaptive Moment Estimation (ADAM) optimizer is used. To reduce the risk of getting stuck in a local minimum while also fine-tuning the model weights, a cyclical learning rate scheduler is set up. The initial learning rate is 10^{-4}, the maximum learning rate is 10^{-1}, the step size is 2 epochs, the mode is “triangular2”, i.e., the learning rate is scaled according to

$$f(x) = \frac{1}{2^{x-1}} \tag{4.1}$$

where x is the number of cycles (batches). With such a learning rate scheduler, the learning rate does not decrease monotonically but instead oscillates between the initial and maximum learning rates, allowing the model to explore different regions of the training area and potentially escape local minima. In the end, the model with the best validation MSE is saved and used.

Chapter 5

Experimental Results

This chapter presents the evaluation results of six machine learning architectures for manufacturing load forecasting. The architectures are evaluated on both synthetic and real-world datasets from the use cases described in Chapter 4. First, the architectures are compared with the default datasets of the AVL BIC use case (synthetic dataset) and the Industrial Robot (IR) use case (real-world dataset). Thereafter, the key factors driving forecast accuracy, training efficiency, and inference speed are explored by analyzing the influence of the amount of training data, augmentation, and feature characteristics like profile length, feature sparsity, and embodied parameter dependencies in the load profile. The chapter concludes with a discussion of the results with regard to the real-world applicability of the model architectures.

5.1 Architecture Comparison on Complete Synthetic Dataset

The first experiment is dedicated to comparing the different model architectures on a synthetic dataset based on the AVL BIC use case. The dataset is created based on the dependencies listed in Tables A.1–A.4. The number of total samples is set to 1000 (more on this choice in Section 5.3). The dataset is split into training, validation, and test datasets with a ratio of 70/20/10 %. The normalized average minimal Euclidean distance $\|\overline{d_{\text{min}}}\|$ between the test dataset and the training/validation datasets of the normalized parameter samples according to Equation 3.10 is 0.063. Training and validation datasets are augmented by a factor of 5 by adding Gaussian noise ($\mu = 0\,\text{W}$, $\sigma = 5.0\,\text{W}$) to the power profiles. This accounts for potential measurement noise in real-world data and acts as a form of regularization, helping to prevent overfitting by introducing additional data variability.

The distribution of recipe IDs in the respective datasets is shown in Figure 5.1. As can be seen from the figure, the share of recipe IDs is not equal across the datasets for all recipe IDs

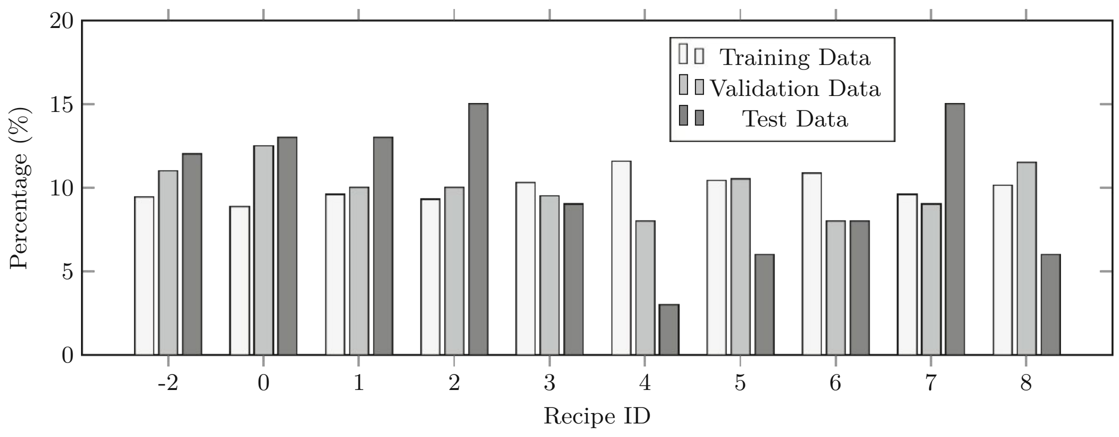


Figure 5.1: Percentage of recipe IDs in the training, validation, and test datasets.

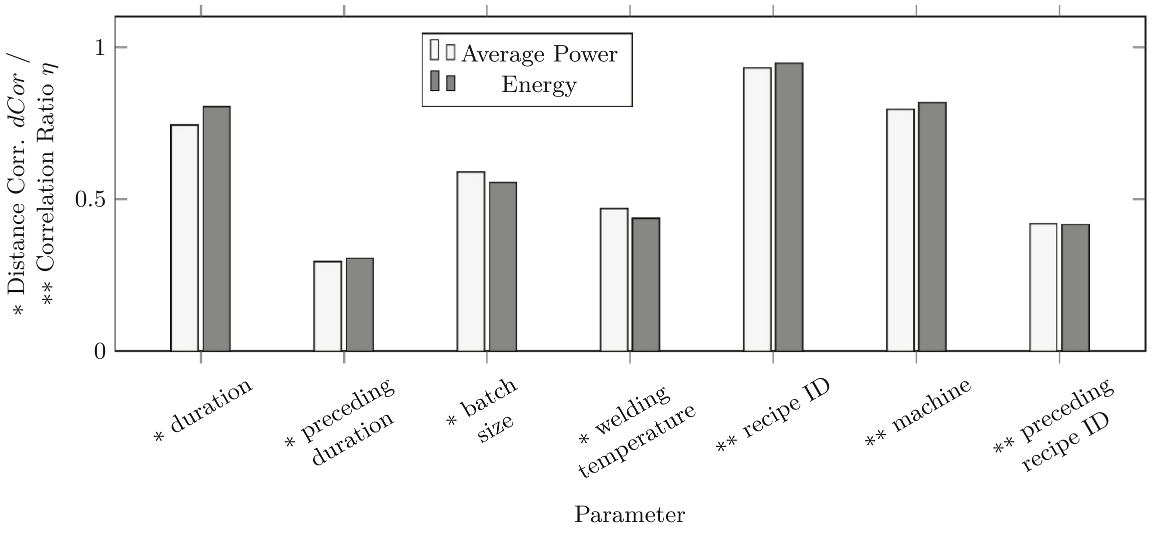
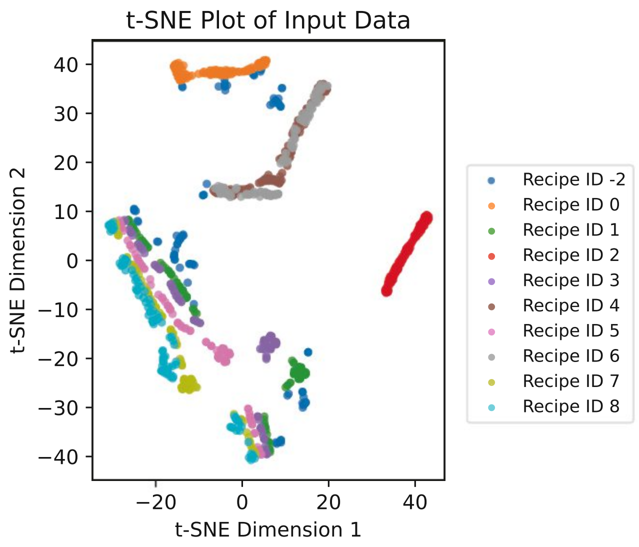


Figure 5.2: Correlation between parameters and power profile of the synthetic dataset.

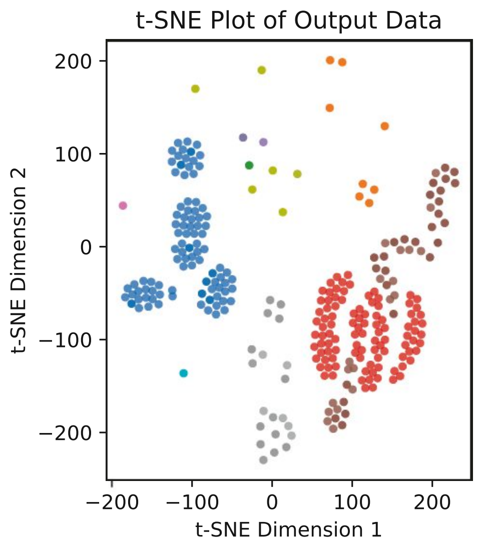
(although for most recipe IDs it roughly is). However, this does not impair the validity of the split since also, in reality, the share of executed recipe IDs might vary over the lifetime of a factory, which is the same as a different distribution within the training/validation dataset and the test dataset.

In Figure 5.2, the correlation between parameters and power profile of the synthetic dataset is shown. The correlation (explanation of $dCor$ and η in Section 3.5) is calculated between all process parameters and both the average power and the energy of the power profile. The strongest correlation can be observed for the recipe ID and the machine, which is also linked to the recipe ID. The weakest correlation is observed between the power profile and the preceding duration. The correlations for the average power and the energy are very similar.

In Figure 5.3, t-SNE plots for the input (process parameter) and output (power) values of the synthetic dataset are shown (Figure 5.3(a) and 5.3(b), respectively). It strikes the eye that



(a) t-SNE plot of the input (process parameter) values.

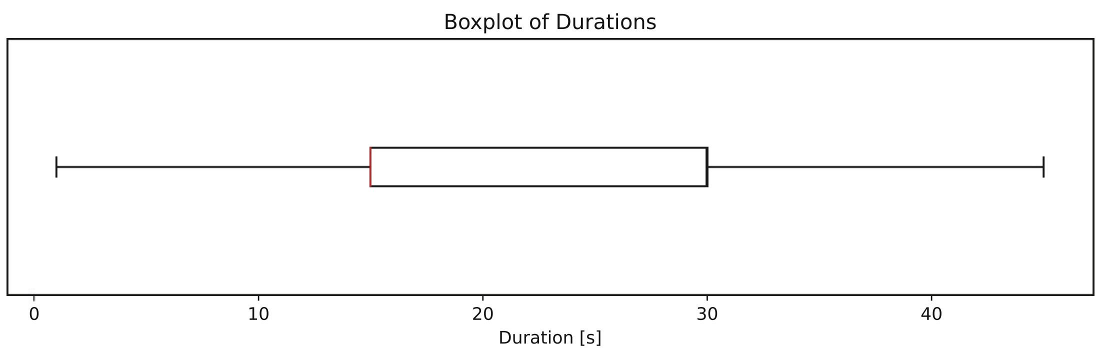


(b) t-SNE plot of the output (power) values.

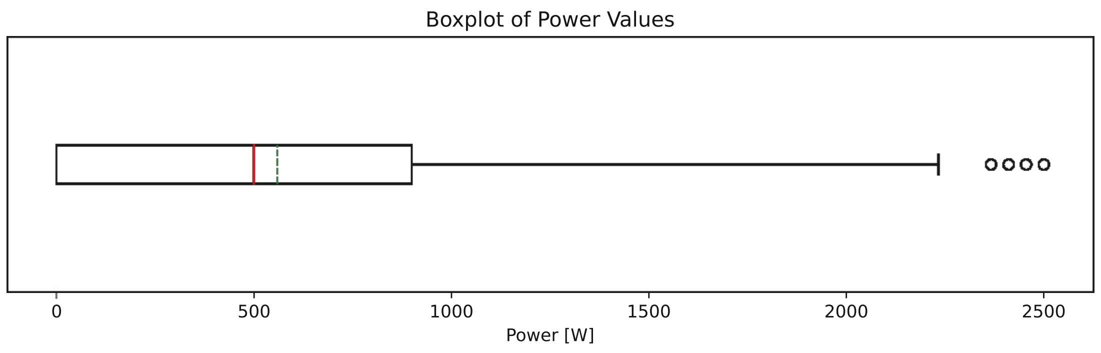
Figure 5.3: t-SNE plots of the input and output values of the synthetic dataset.

most recipe IDs' input values form a clearly defined group, unlike the ones from recipe ID -2 (idle), whose input values are scattered across the whole map. This is due to the variety of possible machines, durations, and preceding processes for the idle time. It can also be seen that recipe IDs 4 (gluing) and 6 (packing) have similar input values, as well as recipe IDs 1, 3, 5, 7 (transport processes between stations), and 8 (quality check). It is not surprising that the transport processes have similar input values. The similarity between the input values of the gluing and the packing process could possibly be explained by the identical duration and batch size range. The welding process, which also has the same duration and batch size range, also has the additional parameter "welding temperature" and, therefore, forms a distinct group in the t-SNE plot. Most recipe IDs' power values do not vary a lot, which can be seen from the tight groups (sometimes even only one spot) in Figure 5.3(b). The recipe IDs with the biggest variety in power profiles are -2 (idle), 2 (welding), 4 (gluing), 6 (packing), and 0 (stacking). The idle time has the widest variety in duration, which leads to a wide variety of profiles. Welding, gluing, packing, and stacking processes are the processes that have a repetition dependency on the batch size and, therefore, vary more than other processes with other profile dependencies.

Figure 5.4 shows the distribution in the dataset of process durations, which is important for the ratio between power and padding values, and the distribution of power values, which is important to evaluate MSE values. The durations are distributed quite evenly between 1 minute and 45 minutes, the mean of the power values is 558.8 W. While the median value is shown for both distributions, the mean value is only shown for the power values due to its importance for judging the MSE.



(a) Process duration



(b) Power values (green dashed line marks mean value)

Figure 5.4: Distributions of duration and power values within each experiment series.

Figure 5.5 shows the training history of the Ensemble LSTM architecture as an example. The training history includes the training loss (MSE), the validation loss (MSE), and the learning rate for each epoch of the training on logarithmic scales. The cyclical learning rate pattern that oscillates between the minimum and the decreasing maximum is clearly visible. The training loss value decreases more monotonically than the validation loss value, and from around epoch 35, the training reaches the state of overfitting, i.e., the model performs significantly better on the training dataset than on the validation dataset. It can also be seen that the loss values of both training and validation often increase temporarily with an increasing learning rate, but perhaps to the benefit of escaping a local minimum.

The training results and parameters that form the basis for comparing the different model architectures are three-fold: The MSEs with and without padding values are shown in Figure 5.6. The training durations are shown in Figure 5.7. And finally, the inference duration is compared in Figure 5.7. The comparison of MSEs shows that the RNN and Seq2Seq architectures are pretty much on par, with the Seq2Seq architecture performing slightly worse than the others. The ensemble learning LSTM architecture based on the recipe ID performs best, its prediction MSE is better than the single RNN architectures by a factor of almost 10. Its MSE of 345.1 W^2 compared to the mean power value of 558.8 W means that it achieves an average prediction error

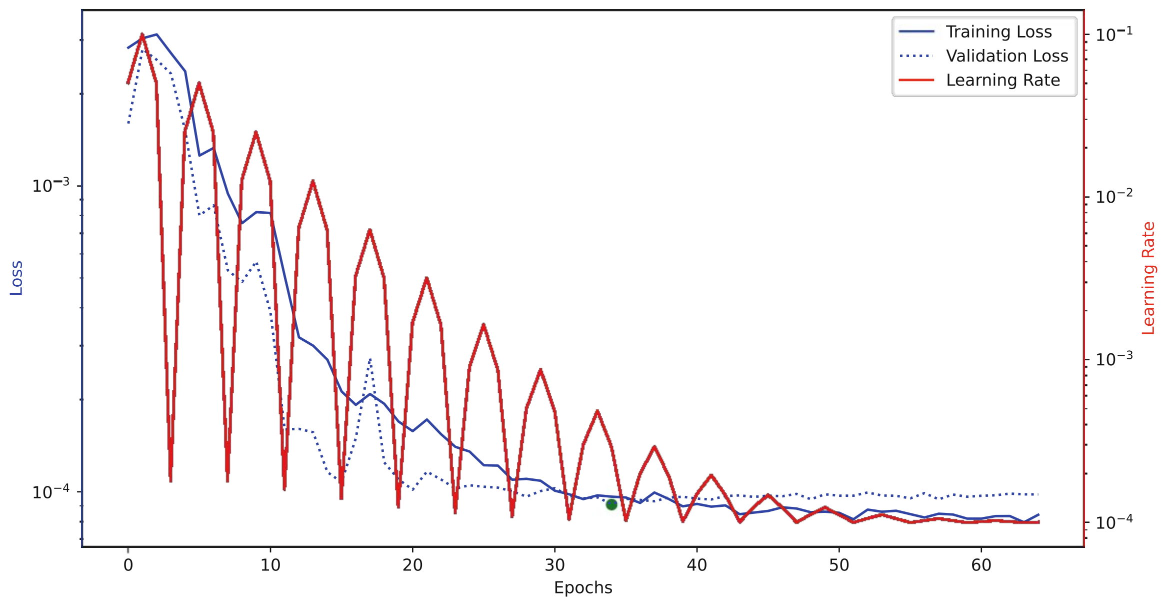


Figure 5.5: Training history of the Ensemble LSTM model.

of approximately 19 W, which is substantially lower than the mean power itself (3.3 %). The NN makes by far the worst predictions, the predictions from the test set reveal that the predictions by the NN are all identical, independently from the input parameters. A closer inspection of the consistent prediction vector indicates that it is the average of all power profiles from the training dataset, i.e. the NN does not learn the underlying relationships in the training data. This can be caused by an insufficient model complexity (e.g., too few layers or neurons) or by an inappropriate architecture. Since the MSE also does not improve with up to 40 layers, it can be concluded that the NN is not appropriate for predicting power profiles just from process parameters. The NN architecture has the strategic disadvantage to the RNN architectures in that it does not model any relationship between the output values, so each power value has to

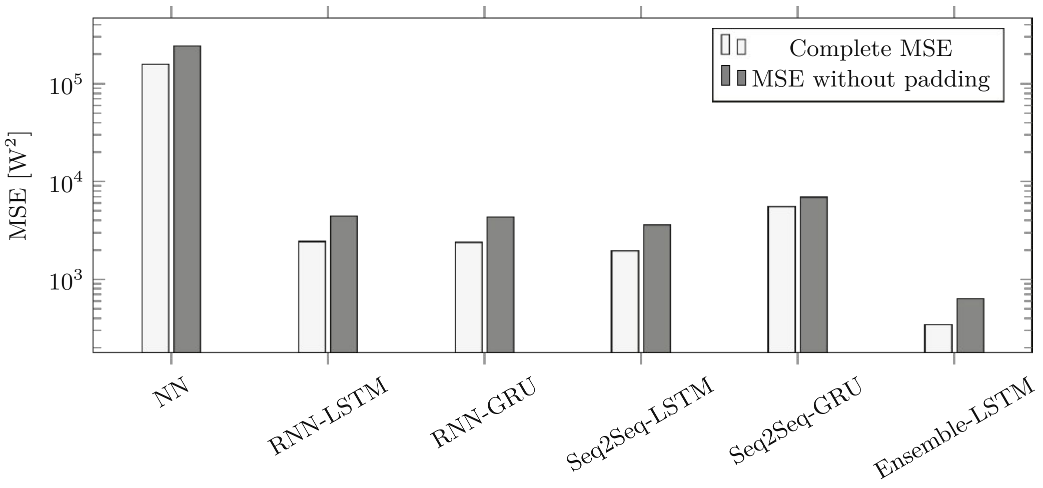


Figure 5.6: MSE of the different model architectures.

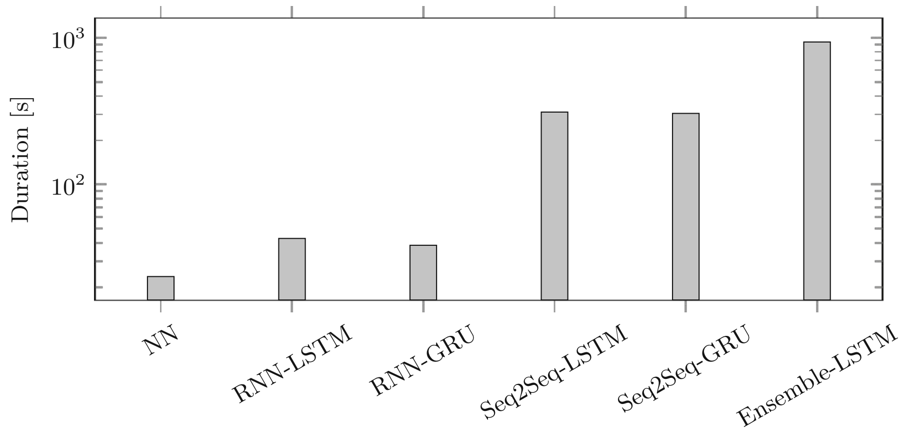


Figure 5.7: Training durations of the different model architectures.

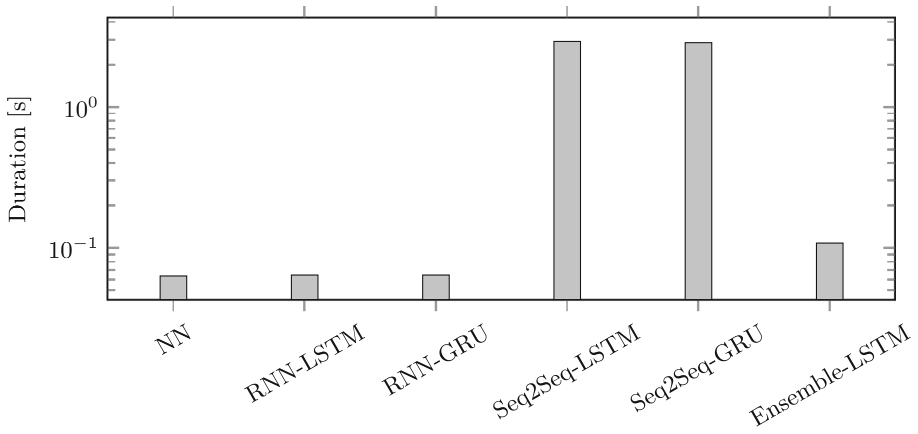


Figure 5.8: Inference durations of the different model architectures.

be determined just based on the input parameters, which leads to very complex functions.

Comparing the values of the prediction MSE with and without padding values in Figure 5.6 shows that the complete MSE including padding values is slightly better for all architectures, i.e., the padding values are predicted better than the power values. This is not surprising since the padding values are the most regular values in the power vectors. The factor between the MSE with and without padding values is roughly 2 for most architectures; for the NN, it is roughly 1.5, and for the Seq2Seq model with GRUs, it is roughly 1.2.

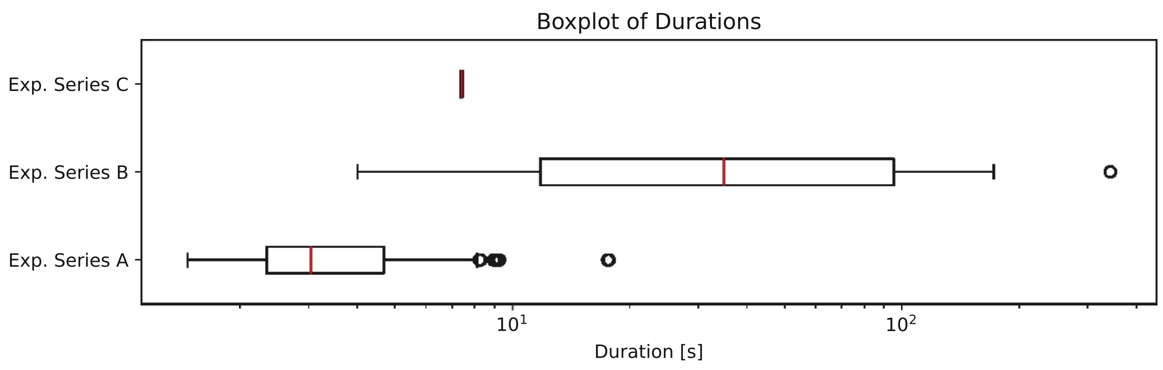
The training durations (see Figure 5.7) vary between 23.5 s for the NN and 939.1 s for the ensemble learning architecture. There is no clear relation between training duration and accuracy: the accuracy of the RNN and Seq2Seq architectures are similar, while their training durations differ by a factor of roughly 10. The training duration seems to depend more on the model architecture. Likewise for the inference durations (see Figure 5.8): while most models' inference durations are around 0.1 s or below, the inference durations of the Seq2Seq models are around 2 s. The longer inference duration of the Seq2Seq architectures can be explained by the recurrent nature of the inference process with encoder and decoder, and the less efficient inference with explicit Python code than with the efficient Tensorflow backend.

In conclusion, the ensemble learning architecture with models for each recipe ID seems to be most appropriate for use cases where the recipe ID is the most decisive parameter for the power profile, while the NN architecture is utterly inappropriate.

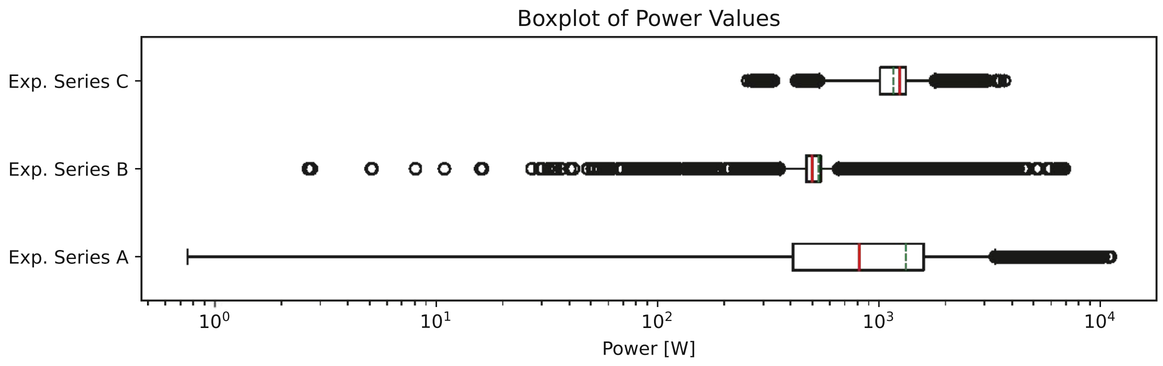
5.2 Architecture Comparison on Industrial Robot Dataset

The second experiment is dedicated to comparing the different model architectures on the datasets from all three experiment series of the IR use case as described in Section 4.2. The number of total samples for the different experiment series is 2400 (experiment series A), 384 (experiment series B), and 1200 (experiment series C). The datasets are split into training, validation, and test datasets with a ratio of 70/20/10 %. The normalized average minimal Euclidean distances $\|\overline{d_{\min}}\|$ between the test dataset and the training/validation datasets of the normalized parameter samples according to Equation 3.10 are 0.016 (experiment series A), 0.055 (experiment series B), and 0.0015 (experiment series C). Training and validation datasets are augmented by a factor of 5 with Gaussian noise ($\mu = 0\,\mathrm{W}$, $\sigma = 5.0\,\mathrm{W}$). The three datasets from the different experiment series differ mainly in the varying parameters (see Table 4.6). Also, the durations of the motions are distributed differently (due to varying velocity and acceleration parameters). These distributions are depicted in the boxplot in Figure 5.9(a) (note the logarithmic scale). While all motions of experiment series C have an identical duration of 7.45 s, the motion durations of experiment series A and especially experiment series B vary widely within $[1.4, 17.7]$ s and $[4.0, 346.4]$ s, respectively, due to varying acceleration and velocity parameters. On the one hand, a longer maximum duration means longer sequences to predict, on the other hand, a wide variety in durations means that shorter sequences have many padding values, sometimes even more than real values. In this experiment, all datasets have a temporal resolution of 500 ms, which results in sequence lengths of 37, 686, and 16 for experiment series A, B, and C, respectively. The distributions of power values are shown in Figre 5.9(b), the mean values are 1319.7 W for experiment series A, 532.5 W for experiment series B, and 1161.5 W for experiment series C.

In Figure 5.10, the correlations between parameters and the power profile of all experiment series of the IR dataset are shown. As for the synthetic dataset, the correlation (explanation of $dCor$ and η in Section 3.5) is calculated between all process parameters and both the average power and the energy of the power profile. In experiment series A, the parameter with the strongest correlation to the average power is the motion duration, while the parameter with the



(a) Motion duration

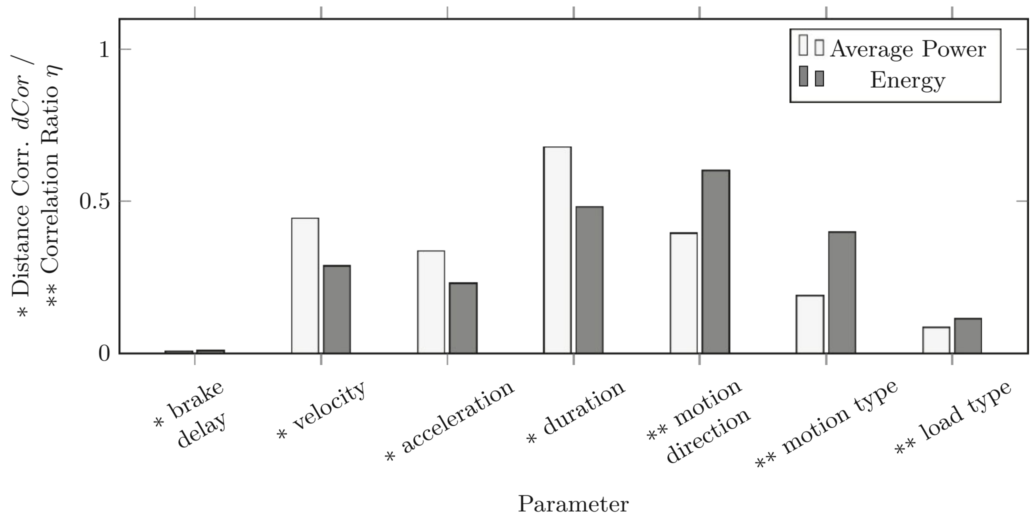


(b) Power values (green dashed line marks mean value)

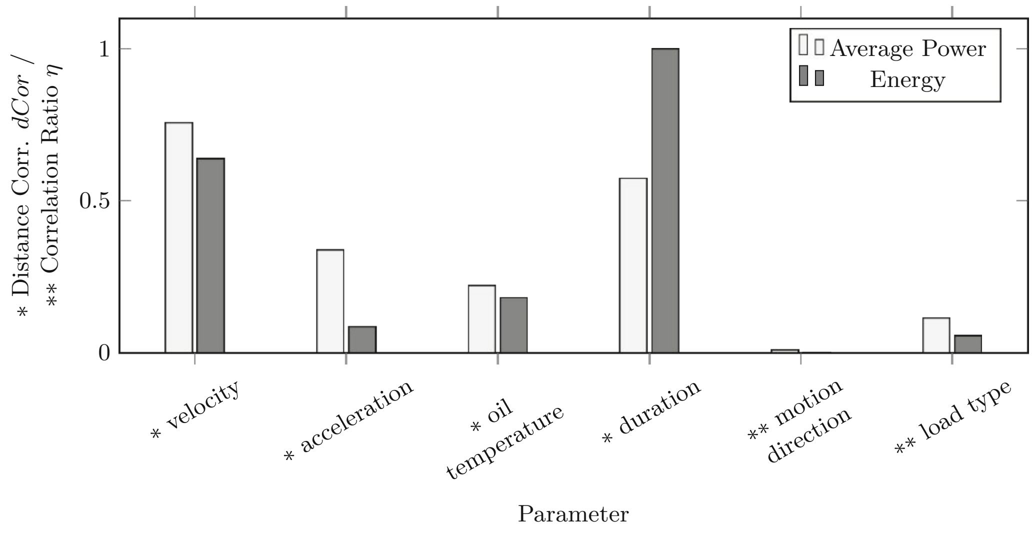
Figure 5.9: Distributions of duration and power values within each experiment series.

strongest correlation to the consumed energy is the motion direction, followed by the motion duration. The reason for the correlation between the consumed energy and the motion direction in experiment series A is that the start and end points have different y coordinates and, therefore, different potential energy; hence, in experiment series B and C (360° rotation of one axis), there is barely any correlation with the motion direction.

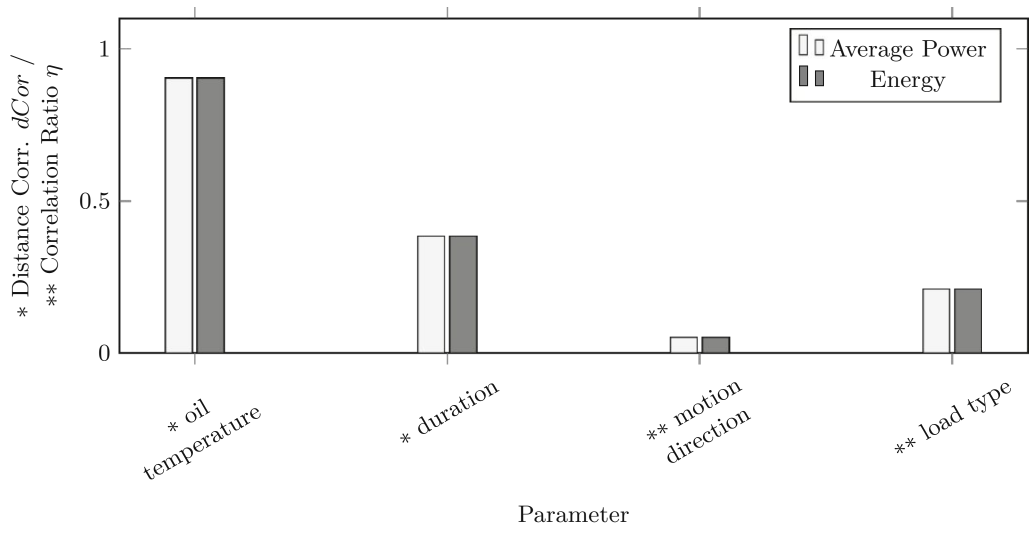
In both experiment series A and B there is quite a difference between the correlations with the average power and the correlations with the consumed energy. In some cases, e.g., velocity and acceleration, the correlation with the average power is higher since higher acceleration and velocity imply faster movements and, therefore, higher power values, though for a shorter time, which is why the correlation with the consumed energy is lower. For the duration, the relation between average power correlation and consumed energy correlation is the reverse in series A and series B. This might be because, in experiment series B, one joint just performs a circular motion, and the acceleration only plays a role in the very beginning of the motion, while in experiment series A, six axes are moved, and a higher acceleration plays a role throughout the motion. The source of the correlation between the oil temperature and the average power and energy consumption in experiment series C is visible from the negative correlation between the oil temperature and friction in Figure 4.7(e).



(a) Experiment Series A



(b) Experiment Series B



(c) Experiment Series C

Figure 5.10: Correlation between parameters and power profile of the IR dataset.

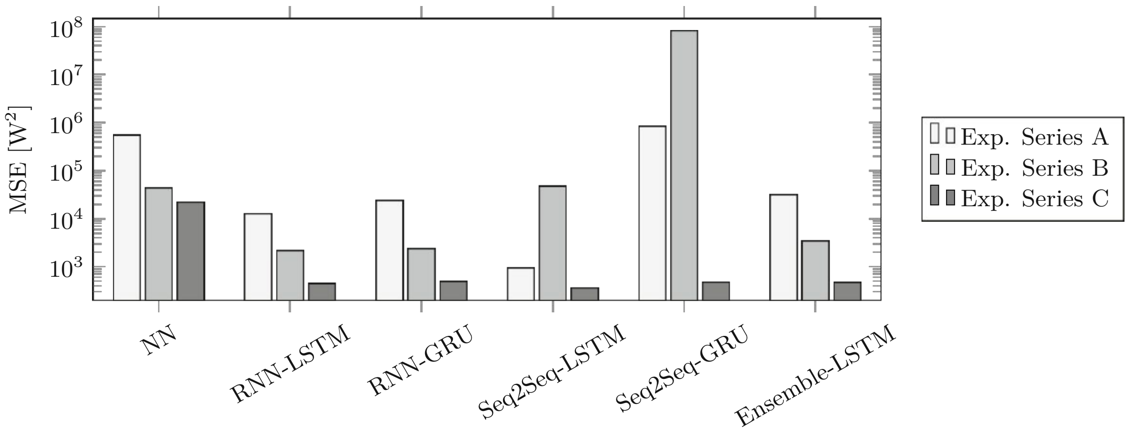


Figure 5.11: Complete MSE of the different model architectures.

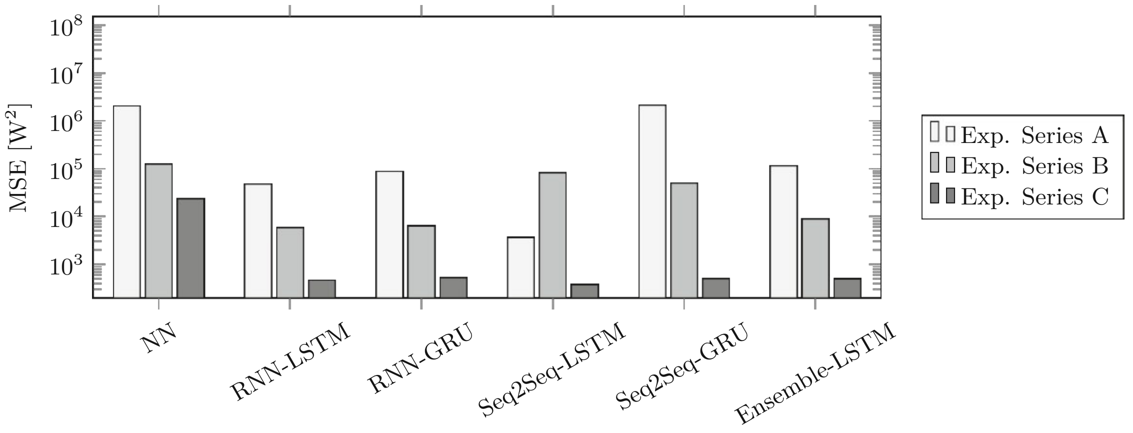


Figure 5.12: MSE without padding values of the different model architectures.

Just like in Section 5.1, the training results and parameters to compare the different model architectures are the following three: The MSEs with and without padding values are shown in Figure 5.11 and 5.12, respectively. The training durations are shown in Figure 5.13. And finally, the inference duration is compared in Figure 5.14. For all model architectures, the MSE (as shown in Figure 5.11) is highest for experiment series A and lowest for experiment series C, except for the Seq2Seq architectures. The cause of sequence is the decreasing variety of parameters and, hence, also the decreasing variety of power profiles. The Seq2Seq architectures have considerable difficulties handling the long sequences of experiment series B. Also, the GRU-Seq2Seq architecture has a much higher MSE than its LSTM counterpart, except for experiment series C. The MSE without padding values (see Figure 5.12) in this case is a better metric to compare the architectures since the effect of a wide variety of durations and long padding sequences at the end of some sequences is not included in the metric. Just like for the synthetic dataset, the MSE without padding values is higher for most model architectures, except for the GRU-Seq2Seq architecture in experiment series B, which indicates that the main

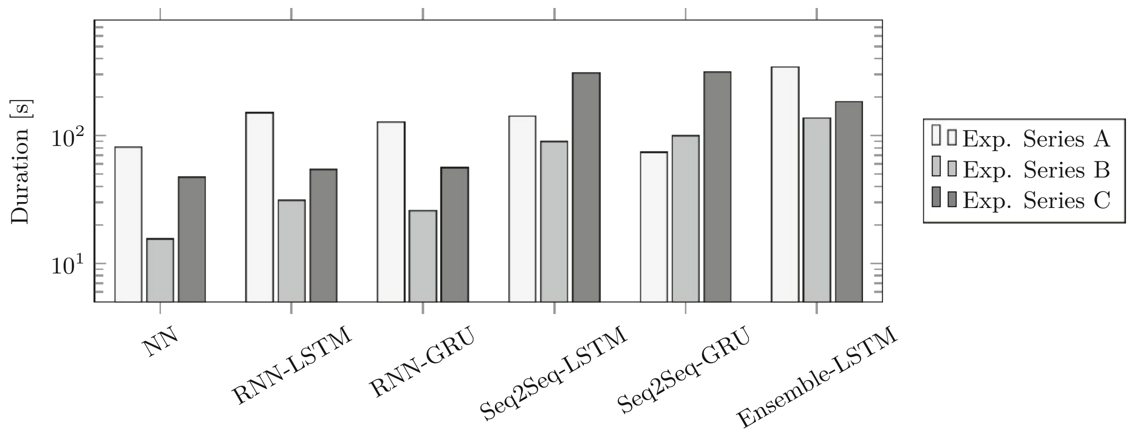


Figure 5.13: Training durations of the different model architectures.

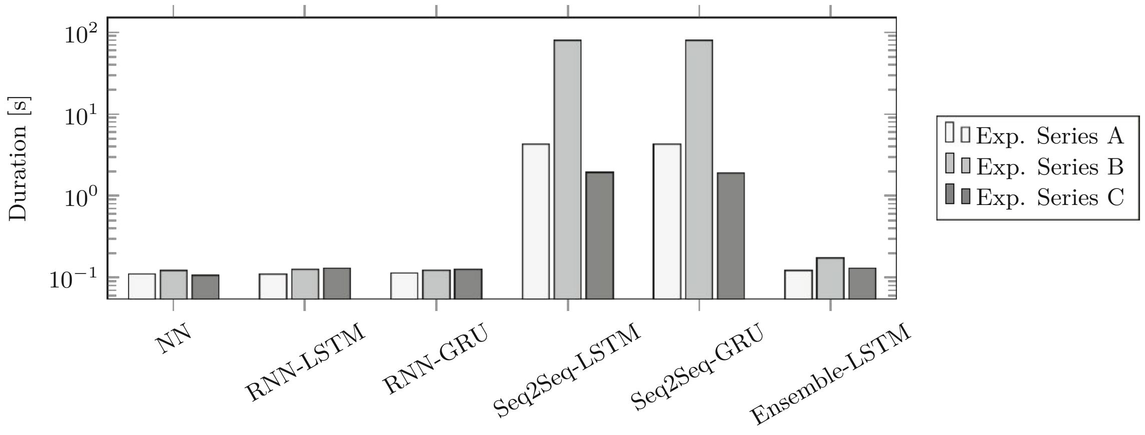


Figure 5.14: Inference durations of the different model architectures.

driver for the exceptionally high MSE including padding values are the padding values. The LSTM-Seq2Seq architecture is the best-performing architecture for experiment series A (MSE without padding: 3598.4 W^2, hence average prediction error 60.0 W, 4.5 % of the mean power value) and C (MSE without padding: 375.3 W^2, hence average prediction error 19.4 W, 1.7 % of the mean power value), while the regular LSTM-RNN architecture has the lowest MSE for experiment series B (MSE without padding: 5783.7 W^2, hence average prediction error 76.1 W, 14.3 % of the mean power value).

The variety in training durations across the experiment series and architectures (see Figure 5.13) is smaller than for the synthetic dataset, and the durations are generally shorter; they range from 15.4 s to 342.5 s. There is no clear trend in the training durations depending on the experiment series; however, for all architectures apart from the Seq2Seq architectures, the training takes the longest for experiment series A and the shortest for experiment series B, which correlates with the number of training samples. The inference duration is more or less constant across the experiment series for all architectures except the Seq2Seq architectures,

Table 5.1: Parameters of the synthetic energy dataset. Parameters with a "∗" character are linked to the recipe ID.

Parameter Name	**Number of Possible Values**	**Applicable Recipe IDs**
Recipe ID	10	$-2, [0 \dots 8]$
Machine	according to Table 4.5 any from Table 4.5	$[0 \dots 8]$ -2
Duration	$*$ $[0 \dots \text{max\_duration}]$	$[0 \dots 8]$ -2
Preceding Recipe ID	2 5 10	$0, 2, 4, 6, 8$ $1, 3, 5, 7$ -2
Preceding Duration	$[0 \dots \text{max\_duration}]$	$-2, [0 \dots 8]$
Batch Size	5	$0, 2, 4, 6, 8$
Welding Temperature	$[500 \dots 1500]$	2

whose inference duration is extremely sensitive to the profile length. Again, as for the synthetic dataset, the reason for this might be the manual inference in a for loop due to the custom model instead of the efficient Tensorflow backend.

5.3 Influence of Number of Samples

Before setting up a series of experiments to test the influence of the number of training samples on the model architecture, a reasonable number of total samples has to be determined based on the number of possible parameter combinations. The synthetic dataset contains the parameters listed in Table 5.1. The parameters with a "∗" character are linked to the recipe ID. Likewise, the preceding duration is linked to the preceding recipe ID but since the preceding recipe ID can be "-2" (idle, i.e., arbitrary duration) for all recipe IDs, the preceding duration can be arbitrary. For recipe IDs $[0, 2, 4, 6, 8]$, the preceding recipe ID can either be the same as recipe ID (because they can only be run on that specific machine type) or "-2". For recipe IDs $[1, 3, 5, 7]$, the preceding recipe ID can be either "-2" or any recipe ID of the set $[1, 3, 5, 7]$, since these are run on the same machine type. Finally, depending on the machine type, the preceding recipe ID for recipe ID "-2" can be any recipe ID.

Considering only the discrete and not the continuous values (i.e., not the welding temperature, the preceding duration, and the duration of recipe ID "-2"), the theoretical number of parameter combinations per recipe ID is

$$N_{\text{ID}} = n_{\text{machines}} \cdot n_{\text{durations}} \cdot n_{\text{prec. IDs}} \cdot n_{\text{batch\_sizes}} \tag{5.1}$$

Table 5.2: Values for the number of training, validation, and test samples for the experiment series with a varying number of training samples.

Experiment	Training Samples	Validation Samples	Test Samples
1	700 (70 %)	200 (20 %)	100 (10 %)
2	650 (65 %)	186 (19 %)	164 (16 %)
3	600 (60 %)	171 (17 %)	229 (23 %)
4	500 (50 %)	143 (14 %)	357 (36 %)
5	400 (40 %)	114 (11 %)	486 (49 %)
6	200 (20 %)	57 (6 %)	743 (74 %)

Substituting the numbers of machines, durations, preceding recipe IDs, and batch sizes for each recipe ID with the values from Table 5.1 yields

$$\begin{aligned} N_{\mathrm{ID}=-2} &= 28 \cdot 1 \cdot 10 \cdot 1 = 280 \\ N_{\mathrm{ID}=0} &= 6 \cdot 1 \cdot 2 \cdot 5 = 60 \\ N_{\mathrm{ID}=\{1,3,5,7\}} &= 8 \cdot 1 \cdot 5 \cdot 1 = 40 \\ N_{\mathrm{ID}=\{2,4,6,8\}} &= 4 \cdot 1 \cdot 2 \cdot 5 = 40 \end{aligned}$$

Hence, the total theoretical number of parameter combinations is

$$N = N_{\mathrm{ID}=-2} + N_{\mathrm{ID}=0} + 4 \cdot N_{\mathrm{ID}=\{1,3,5,7\}} + 4 \cdot N_{\mathrm{ID}=\{2,4,6,8\}} = 600 \tag{5.2}$$

To also account for the continuous parameter values, the number of samples is set to 1000 for the source power profile dataset (without duplicate sets of parameter values). To train the model with different numbers of samples, this source dataset is split with varying ratios into training, validation, and test datasets. To keep the ratio between the training and validation dataset constant, the ratios with a given number of training samples N_{train} are calculated as follows:

$$\begin{aligned} r_{\mathrm{train}} &= \frac{N_{\mathrm{train}}}{1000} \cdot 100\,\% \\ r_{\mathrm{val}} &= r_{\mathrm{train}} \cdot \frac{20}{70} \\ r_{\mathrm{test}} &= 100\,\% - r_{\mathrm{train}} - r_{\mathrm{val}} \end{aligned} \tag{5.3}$$

The values of the number of training samples for the different training runs are given in Table 5.2.

Figure 5.15 shows the prediction MSE for all model architectures over the varying number of samples, Figure 5.16 shows the prediction MSE without padding values. Surprisingly, a bigger

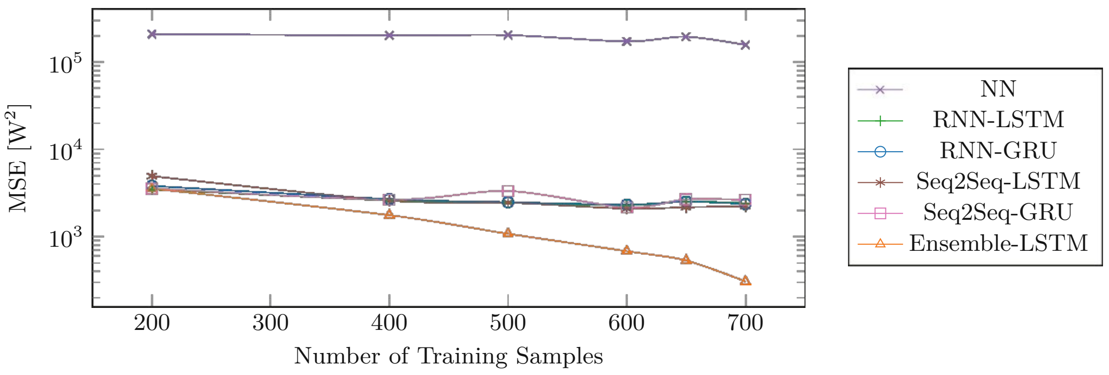


Figure 5.15: Complete MSE of the model architectures with varying number of samples.

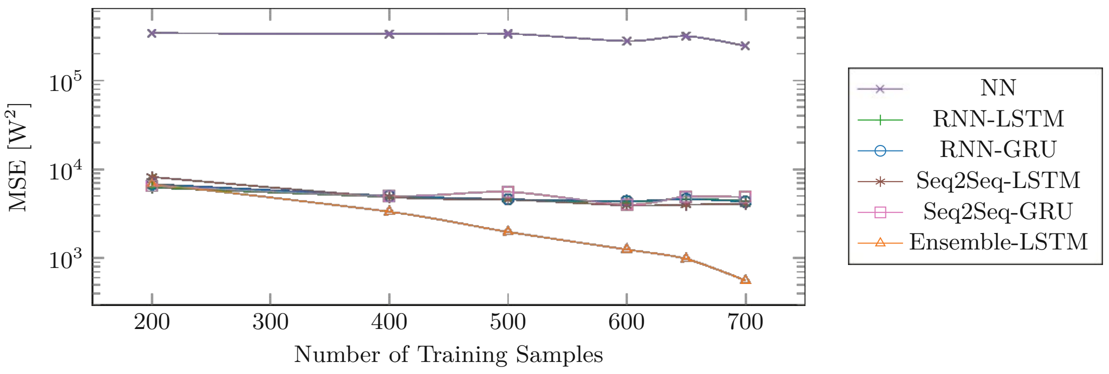
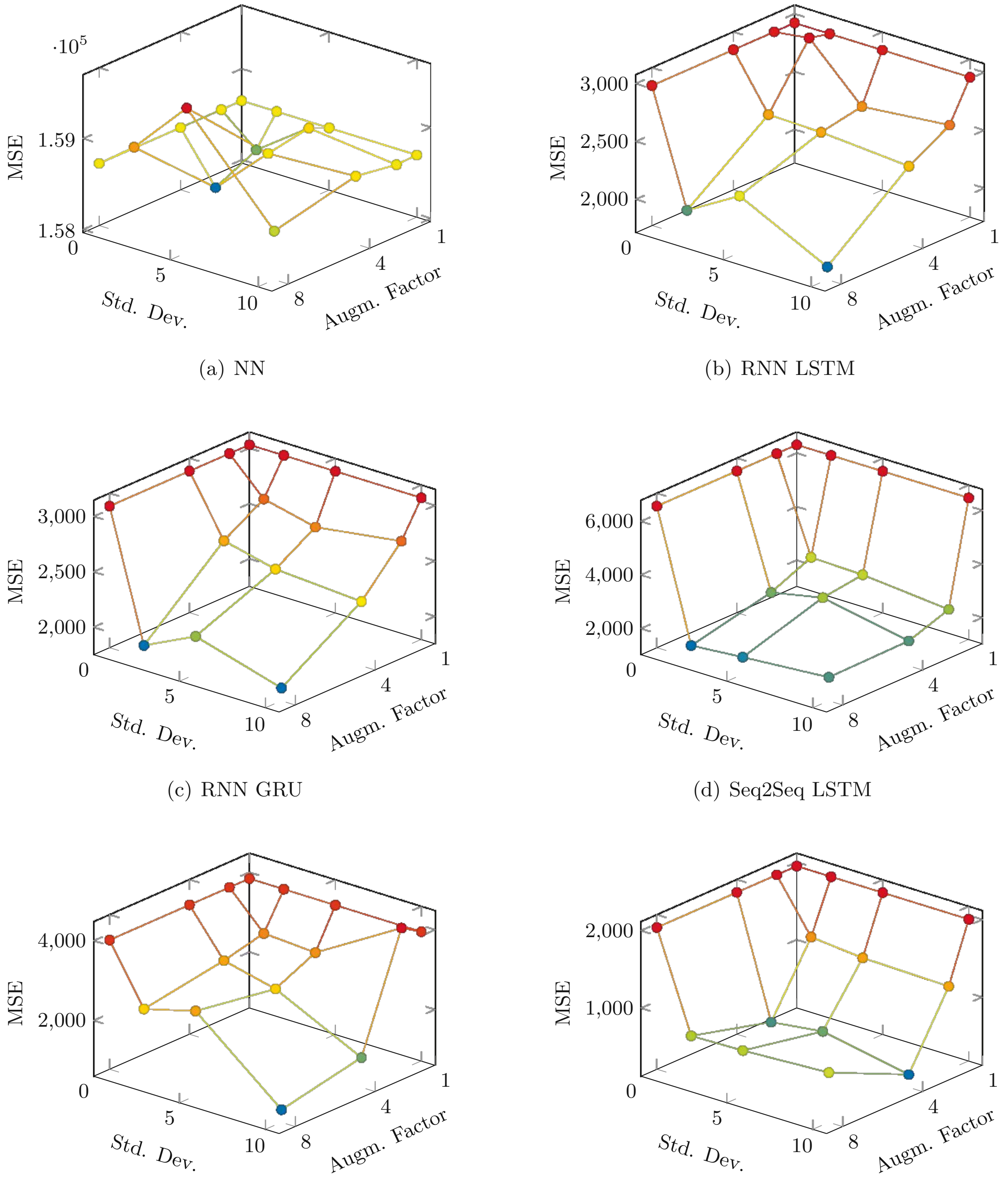


Figure 5.16: MSE without padding of the model architectures with varying number of samples.

number of samples only has a significant effect on the resulting MSE for the ensemble learning architecture. The MSE of the other architectures only decreases minimally with a growing number of samples. An explanation for this might be that each model of the ensemble learning architecture only gets a fraction of the training samples, so the single models profit more from the increase in samples. A growing number of samples also presents the "monolithic" architectures with more seemingly complementary information, i.e., the same parameter value might have different implications for the profile if the meaning of one parameter value depends on another parameter value (e.g., the preceding recipe ID has different implications depending on the current recipe ID). Compared to that, the single models of the ensemble architecture get pre-sorted samples with fewer dependencies and can learn from them more quickly.

5.4 Influence of Augmentation

To examine the influence of different augmentation parameters on the model accuracy, the synthetic dataset with 1000 samples is split into training, validation, and test datasets with a ratio of 70/20/10 %, and the training and validation datasets are augmented with Gaussian noise ($\mu = 0$) with varying standard deviations ($\sigma = [0.0, 2.0, 5.0, 10.0]$ W). Also, the factor by which the data is augmented varies ($f = [1, 2, 4, 8]$), $f = 1$ meaning that the number of samples remains



(a) NN (b) RNN LSTM

(c) RNN GRU (d) Seq2Seq LSTM

(e) Seq2Seq GRU (f) Ensemble LSTM

Figure 5.17: Complete MSE (in W^2) of the architectures with varying augmentation parameters.

the same. One scenario is created for each combination of the augmentation parameters σ and f. However, there is one exception: since $f = 1$ means that the dataset remains unchanged and $\sigma = 0$ means that samples are just duplicated, which would just distort the stopping condition (minimal improvement of the validation MSE of 10^{-6}) by duplicating data to extend each epoch, no scenarios for the combinations $(f, \sigma) \in [(1, 2.0\,\mathrm{W}), (1, 5.0\,\mathrm{W}), (1, 10.0\,\mathrm{W}), ((2, 0.0\,\mathrm{W}), (4, 0.0\,\mathrm{W}), (8, 0.0\,\mathrm{W})]$ are created. For visualization purposes, the MSE values for these combinations are copied from the scenario with $(f, \sigma) = (1, 0.0\,\mathrm{W})$.

The resulting prediction MSEs are presented as mesh plots for each architecture in Figure 5.17. Except for the NN, a clear tendency is visible for a lower MSE with more augmentation. While the exact value of the standard deviation σ does not have too big an influence in most examples, an increasing augmentation factor in the tested range decreases the prediction MSE. This is because the standard deviation would only have a detrimental effect on the MSE if $\sigma^2 > \mathrm{MSE}$. However, even the lowest MSE value ($285.2\,\mathrm{W}^2$, Ensemble LSTM architecture) is greater than the biggest σ^2 ($100\,\mathrm{W}^2$).

5.5 Influence of the Profile Length

The influence of the profile length on the model prediction accuracy is tested by resampling the power series of experiment series A from the IR use case with different temporal resolutions. As explained in Section 4.2, the resampling method does not distort the total energy consumption. The values for the temporal resolutions are $[100, 150, 250, 500, 750, 1000]\,\mathrm{ms}$, which is equivalent to profile lengths of $[19, 25, 37, 72, 119, 178]$. The datasets are split into training, validation, and test datasets with a ratio of $70/20/10\,\%$. Training and validation datasets are augmented by a factor of 5 with Gaussian noise ($\mu = 0\,\mathrm{W}$, $\sigma = 5.0\,\mathrm{W}$).

Figure 5.18 and Figure 5.19 show the complete prediction MSE and the MSE without padding, respectively, for varying profile lengths. For the majority of the tested architectures, the MSE only slightly increases with an increasing profile length, while for the Seq2Seq architectures, the increase of the MSE is more drastic (for GRU-Seq2Seq even more than for LSTM-Seq2Seq). However, the Seq2Seq architectures' MSEs also start from a lower level. For the GRU-Seq2Seq architectures, two surprising observations can be made: first, there is an exception to the growing MSE tendency for a profile length of 119, which is likely due to coincidentally good training, and second, the MSE with padding values is higher than the one without padding values for a profile length of 37, which suggests that the padding values are predicted worse than the power values in this example, and is likely a consequence of coincidentally bad training.

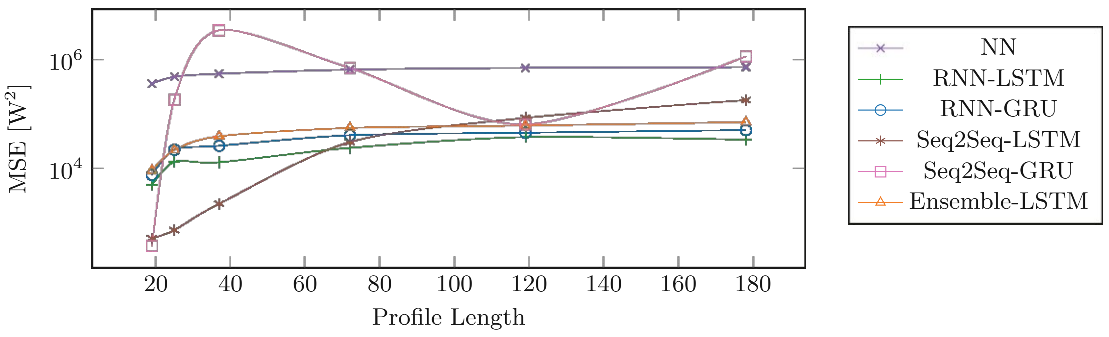


Figure 5.18: Complete MSE of the model architectures with varying profile lengths.

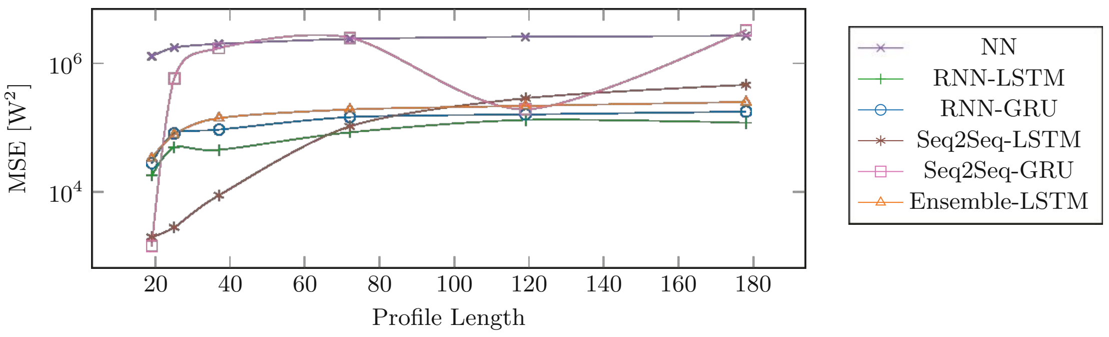


Figure 5.19: MSE without padding of the model architectures with varying profile lengths.

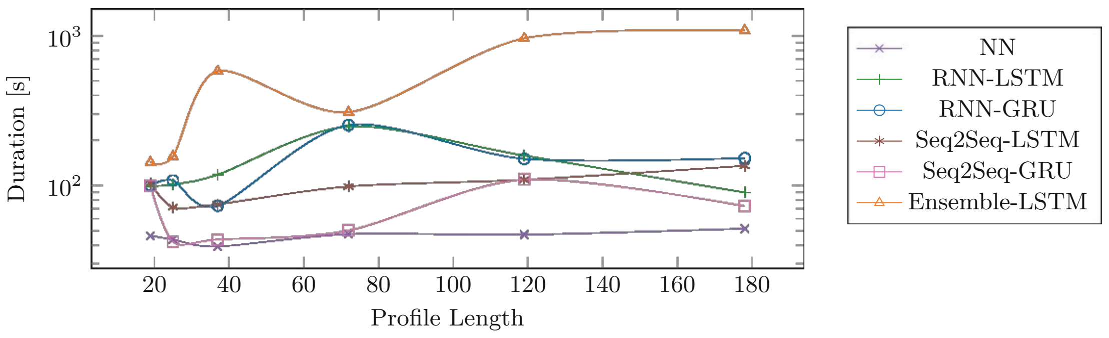


Figure 5.20: Training duration of the model architectures with varying profile lengths.

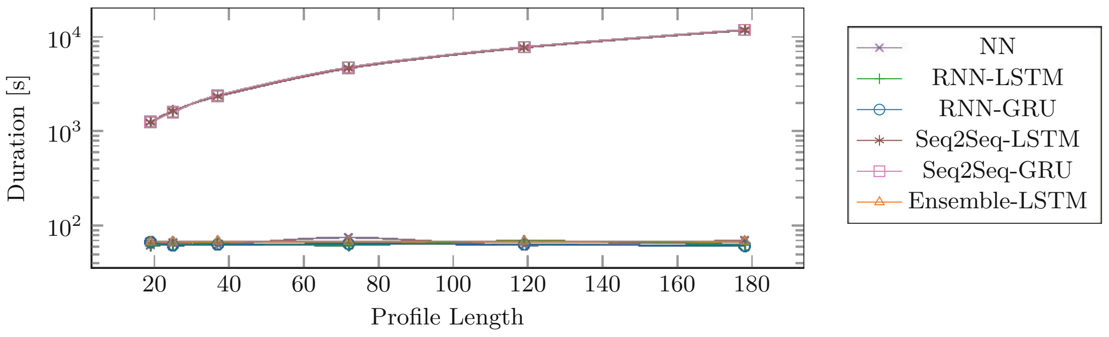


Figure 5.21: Inference duration of the model architectures with varying profile lengths.

Figure 5.20 shows the training duration of the model architectures with varying profile lengths, which does not seem to depend on the profile length. Just the training duration of the Ensemble LSTM architecture seems to increase with longer profiles.

Figure 5.21 shows the training duration of the model architectures with varying profile lengths. While the inference duration of most model architectures is independent of the sequence length, the inference duration of the Seq2Seq architectures strongly depends (linearly) on the length of the sequence to be predicted because, due to the differing training and inference architecture, they use a custom inference loop instead of the efficient Tensorflow backend.

5.6 Influence of Training Data Sparsity

One of the prime reasons for a dedicated energy model instead of a lookup table with historical data is the ability to interpolate between parameter values. To test this ability, the dataset of experiment series C of the IR use case is split into training, test, and validation datasets according to the parameter “oil temperature”. Temperature values present in the test dataset are not present in either the training or the test dataset. After determining the samples for the test dataset, the remaining samples are split randomly with a ratio of 78/22 % into training and validation data to preserve the ratio between the number of training and validation samples from previous experiments (70/20 %). The varying parameter to test the interpolation capability of the model architectures is the gap between known temperature values from the training data, which ranges from 1 °C to 5 °C in steps of 1 °C. Figure 5.22 illustrates the distribution of temperature values across training, validation, and test data in the different scenarios. The relative average distance (RAD) introduced in Section 3.5 for the different scenarios is 1.2 % ($\Delta T = 1\,°\text{C}$), 1.7 % ($\Delta T = 2\,°\text{C}$), 5.0 % ($\Delta T = 3\,°\text{C}$), 3.7 % ($\Delta T = 4\,°\text{C}$), and 10.0 % ($\Delta T = 5\,°\text{C}$). The main reason for the fluctuations in the RAD despite a monotonically increasing ΔT is the varying range of test set values at the end of the whole parameter range. This effect can be seen in Figure 5.22: the range of test set temperature values down from 50 °C varies also relative to ΔT.

To assess the MSE predictions with interpolation, they have to be compared to the power profile from the closest parameter combination. For this purpose, for each sample from the test set, the MSE between the power profile of the test sample and the power profile of the closest sample from either the training or validation set (i.e., the closest temperature value and identical other parameter values) is calculated and an average MSE of such a naive lookup table is calculated.

Figure 5.23 shows the complete prediction MSE with varying parameter gaps in the training

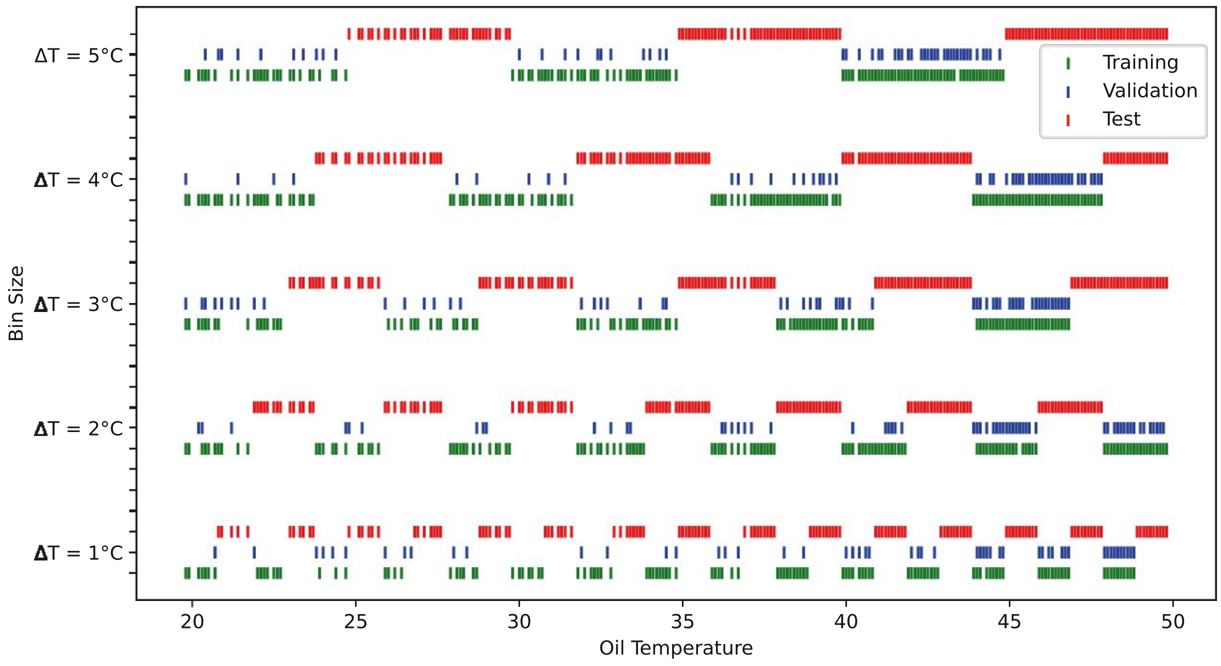
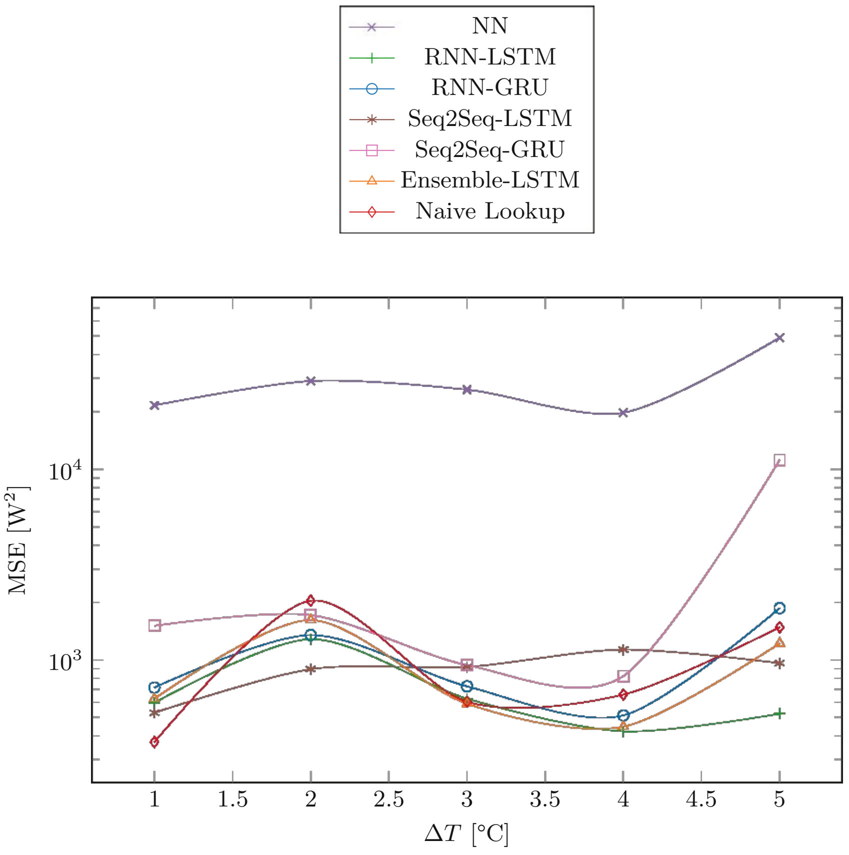


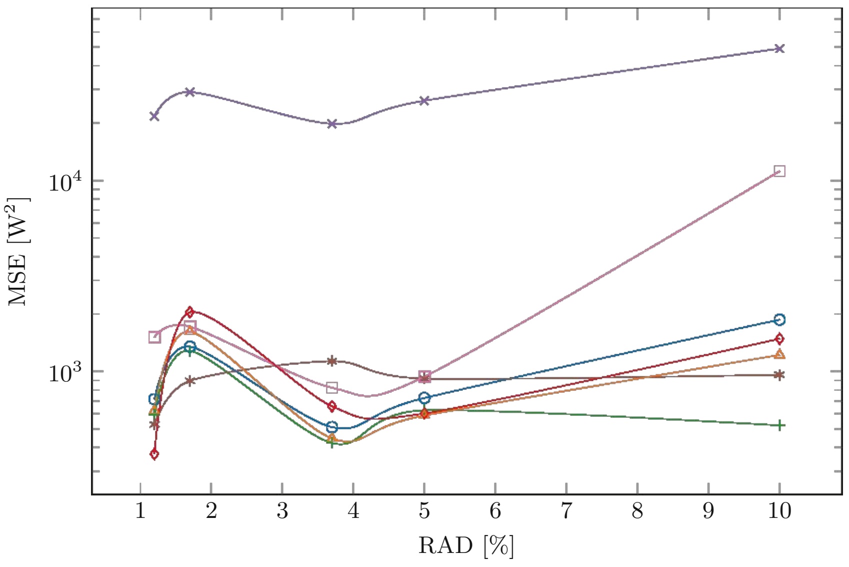
Figure 5.22: Distribution of the temperature values across training, validation and test set for different models.

and validation data of the model architectures and the hypothetical naive lookup table, both over ΔT (Figure 5.23(a)) and over the RAD (Figure 5.23(b)). The MSE without padding values is not evaluated because, in experiment series C from the IR use case, all power profiles have an identical length. For all model architectures, the curve of the MSE follows a similar shape as the naive lookup table, whose MSE surprisingly does not just grow with bigger parameter value gaps: it rises slightly from $\Delta T = 1\,°\text{C}$ to $\Delta T = 2\,°\text{C}$, before falling to $\Delta T = 4\,°\text{C}$, and more or less sharply rising at $\Delta T = 5\,°\text{C}$. An exception to this rule is the LSTM-Seq2Seq architecture, whose MSE rises monotonically from $\Delta T = 1\,°\text{C}$ to $\Delta T = 4\,°\text{C}$, before slightly dropping again at $\Delta T = 5\,°\text{C}$. The curve shape over the RAD is also not continuous. The reason for the surprising shape of the MSE curve of the naive lookup table is probably that the lookup table just chooses the closest temperature value, while for non-linear relations between parameter value and power profile, a different profile can be more similar, e.g., the function value of $f(x) = x^2$ of $x = 5$ is closer to $x = 2$ than to $x = 7$. For $\Delta T = 1\,°\text{C}$, no model architecture is better than the naive lookup table; for $\Delta T = 2\,°\text{C}$, all model architectures are better, and for the other values of ΔT, only some model architectures are better.

Overall, more model architectures perform worse than the naive lookup table. Nevertheless, in cases with non-linear parameter dependencies of the power profile, the architectures can still outperform a naive lookup table. Therefore, especially in cases when the exact non-linear dependency is not known, which might be used to improve the lookup table, it can still be



(a) Varying ΔT



(b) Varying RAD

Figure 5.23: Complete MSE of the model architectures with varying parameter gaps.

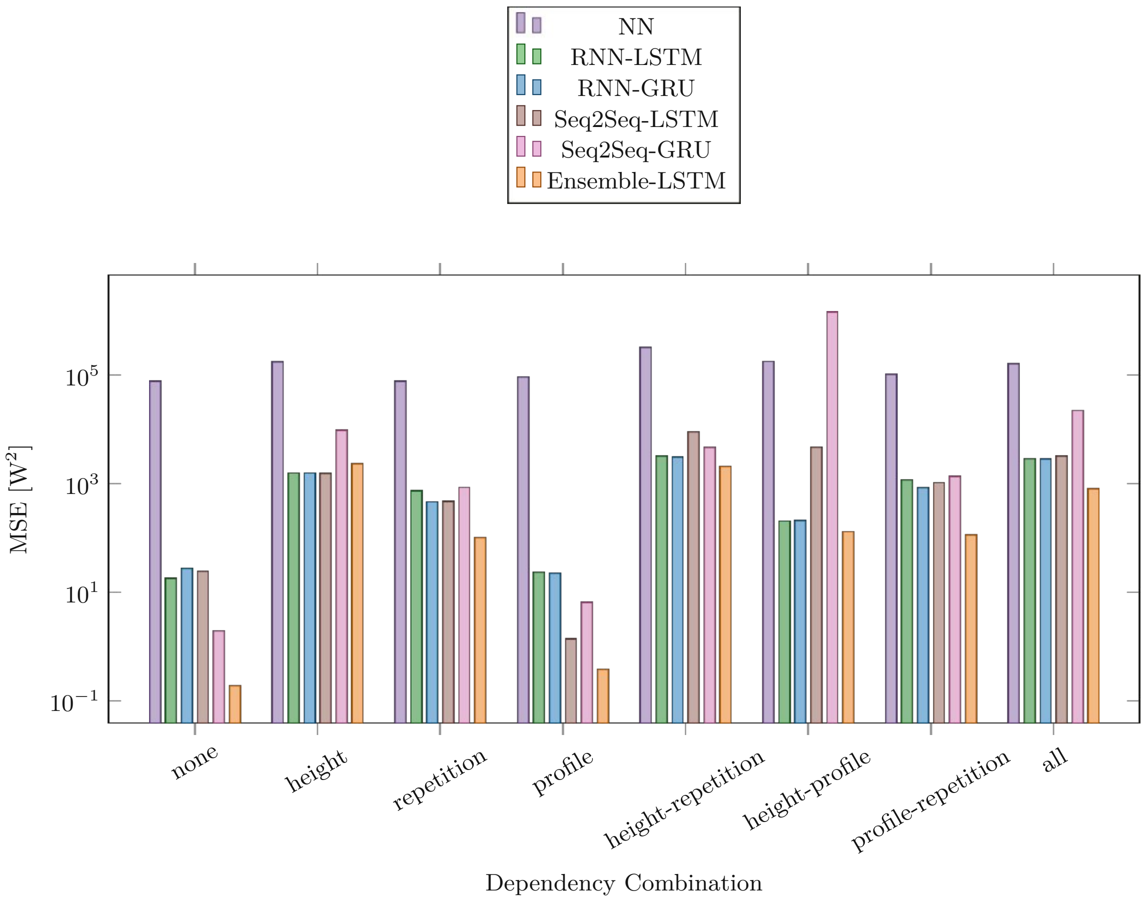
worthwhile to prefer machine learning over a lookup table. The model architectures that perform most consistently better than the naive lookup table on this interpolation task are the LSTM-RNN and the Ensemble LSTM.

5.7 Influence of Parameter Dependencies

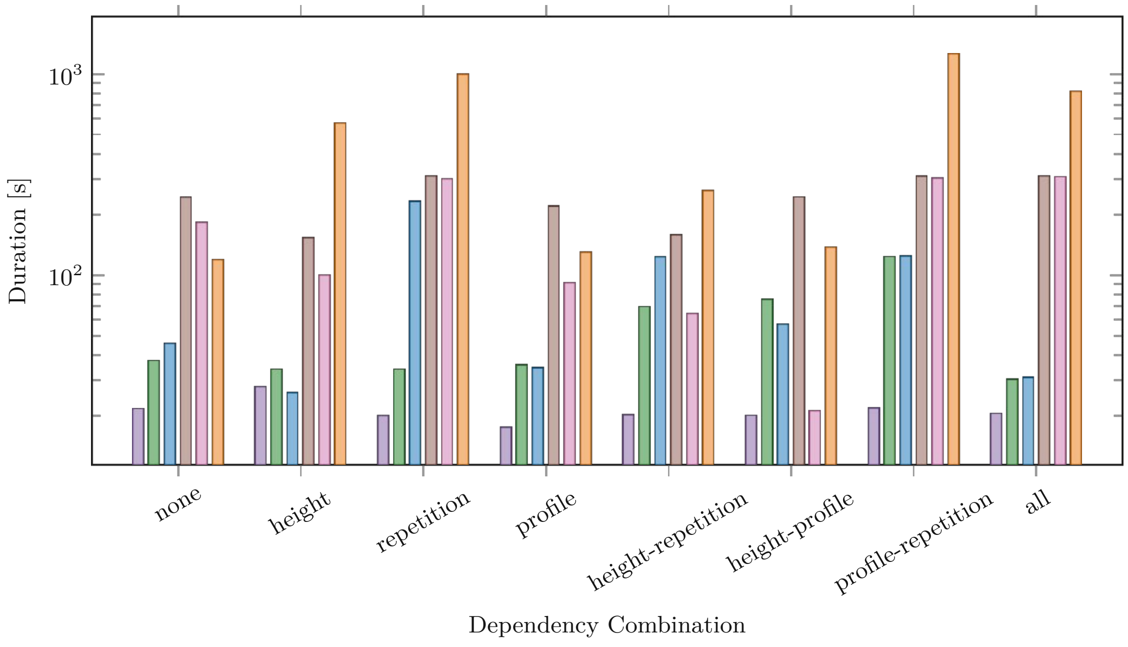
Different synthetic datasets are created for all combinations of height, repetition, and profile dependencies (including no dependencies at all) to assess which dependencies are most challenging for the model architectures and how the architectures compare to each other on the single dependency combinations. For each dependency combination, the respective dependencies, as described in Section 4.1.2 and in Tables A.2–A.4, are either switched on or off for the data generation and 1000 samples are generated. The datasets are split into training, validation, and test datasets with a ratio of 70/20/10 %. Training and validation datasets are augmented by a factor of 5 with Gaussian noise ($\mu = 0\,\text{W}$, $\sigma = 5.0\,\text{W}$).

Figure 5.24(a) shows the MSE for the different dependency combinations and model architectures. The following ranking is based on the median of the MSEs to exclude outliers from the ranking. It can be seen that the scenario with only profile dependencies yields the highest model accuracy, closely followed by the scenario with no dependencies at all. Actually, there is only the profile dependency for idle consumption, so the difference between the profiles without any dependencies and the profiles with just profile dependencies is not really big. Next up are repetition dependencies and then the combination of profile and repetition dependencies. The fifth combination in the row is the scenario with height dependencies, followed by the combination of height and profile dependencies. The last combinations are the combination of all dependencies and, roughly on par, the combination of height and repetition dependencies. A general tendency towards a higher MSE with more dependencies can be observed, but there are single dependencies that seem to pose more difficulties for the model architectures than combinations of other dependencies. The most complicated dependencies to predict in this experiment are the height dependencies, then the repetition dependencies, and lastly the profile dependencies. The profile dependencies are also inherently tested with the recipe ID, except for the Ensemble LSTM, which has a separate model for each recipe ID.

The training durations of the architectures for different dependency combinations are shown in Figure 5.24(b). Ensemble LSTM and the Seq2Seq architectures are (apart from combined height and repetition dependencies) the three architectures with the longest training duration. The NN generally has the shortest training duration but also has the highest prediction MSE.



(a) Complete MSE



(b) Training duration

Figure 5.24: MSE and training duration of the model architectures with varying profile dependencies.

5.8 Discussion

Having conducted a series of tests to find the characteristics of the model architectures, conclusions about the strengths and weaknesses of each architecture can be drawn. The NN architecture proved to be unsuitable for predicting power profiles from process parameters alone. It lacks the capability of modeling temporal relationships between the values of the power profile and the relations between each single power value and the process parameters apparently are too complicated for the NN to capture. Both classic RNN architectures exhibited similar characteristics: a good prediction accuracy (rarely the best, but consistently in the middle field of the tested architectures) which proved to be relatively independent of the profile length, short training durations, and an inference duration which was independent of the profile length. The ensemble version of the LSTM-RNN showed even better prediction accuracy in cases when one categorical parameter had a disproportionate influence on the power profile compared to the other parameters. However, it needed more training data because it was split into several models and the training took longer (factor 2–10). Of the Seq2Seq architectures, the LSTM version showed better prediction accuracy than the GRU version. For the basic synthetic dataset and the dataset from the IR use case, the prediction accuracy was among the two best, except for experiment series B from the IR use case. This is an example of the major weakness of the Seq2Seq architectures: the prediction accuracy decreases with a growing profile length. In the conducted experiment, the accuracy decreased quite drastically up to a profile length of roughly 80, and after that, the decrease in accuracy was similar to the other scenarios.

The real-world applicability of this type of model depends on the available data, the required accuracy, the complexity of the processes, and their adaptability. If a process does not have any parameters to configure or runs on a specialized machine and is always the same, like in continuous production or mass production, it is sufficient to use historical load profiles as forecasts and not necessary to develop a dedicated energy model. This type of energy model aims at discrete production types like batch production or job production which run on adaptable machines with adaptable process parameters. If the underlying physical processes can be easily modeled or the model must be so accurate that it justifies the increased development effort, a physical model is to be preferred. However, if processes have many varying parameters, historical data is available, and the prediction error does not need to be much lower than 5 %, the proposed energy model is a suitable method to forecast the load profile of a process before execution.

Chapter 6

Conclusion

This thesis proposed a novel data-based energy model architecture to predict the power profile of manufacturing schedules and processes prior to their execution. The targeted application area is discrete production, more specifically, production types like batch production or job production, with processes of smaller quantities that run on adaptable machines with adaptable process parameters.

The tested machine-learning architectures were a Neural Network (NN), the Recurrent Neural Network (RNN) types Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), an ensemble learning model consisting of one LSTMs model for each category of a specified categorical parameter, and sequence-to-sequence (Seq2Seq) models with LSTMs and GRUs. Of these architectures, the Ensemble LSTM architecture and the LSTM-Seq2Seq architecture generally showed the best prediction accuracy while the NN proved to be unsuitable for the task. A weakness of the Seq2Seq architectures is their dependency on the profile length regarding the prediction accuracy and the inference duration.

The interpolation performance of the architectures depends more on the nature of the parameter dependency than on the parameter value gaps in the training data. In some tested cases, just taking the historical profile with the most similar parameters yielded better results than the machine-learning models, in other cases, all architectures except for the NN performed better than this lookup table. The architectures that exhibited a high interpolation accuracy most consistently were the LSTM-RNN and the Ensemble LSTM.

Training data augmentation with Gaussian noise was shown to be effective for all model types, i.e., the prediction MSE was lower with augmented training data. The augmentation factor had a bigger influence on the resulting accuracy than the standard deviation of the noise.

The parameter dependencies of the power profile that were the most challenging for the model architectures to capture and predict were height dependencies, then repetition dependencies, and lastly profile dependencies. Altogether, an absolute prediction error of 5 % with the most suitable architectures in the respective cases can be expected.

Following this work, the next step would be a real-world application in a manufacturing context, for example in schedule optimization to minimize energy costs in markets with real-time pricing. An energy model of the proposed architecture can give a forecast of the energy consumption of each process and the processes can be scheduled according to their energy consumption in cheap (energy-intensive) or more expensive (less energy-intensive) time slots. In a specific use case, it could also be tested how much the prediction accuracy could be improved with hyper parameter tuning. In the context of the growing research field of Explainable Artificial Intelligence (XAI), further development of the proposed architectures could contribute to drawing conclusions from the captured dependencies about energy-efficient scheduling and energy hotspots.

Bibliography

Print Resources

[AB15] Ronay Ak and Raunak Bhinge. “Data analytics and uncertainty quantification for energy prediction in manufacturing”. In: (2015), pp. 2782–2784. DOI: `10.1109/BigData.2015.7364081`.

[Agg18] Charu C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, 2018. ISBN: 9783319944630. DOI: `10.1007/978-3-319-94463-0`.

[Bam+16] Mario Bambagini et al. “Energy-aware scheduling for real-time systems: A survey”. In: *ACM Transactions on Embedded Computing Systems (TECS)* 15.1 (2016). ISBN: 1539-9087 Publisher: ACM New York, NY, USA, pp. 1–34.

[Bän+21] Kristian Bänsch et al. “Energy-aware decision support models in production environments: A systematic literature review”. en. In: *Computers & Industrial Engineering* 159 (Sept. 2021), p. 107456. ISSN: 03608352. DOI: `10.1016/j.cie.2021.107456`.

[Ber+22] Miguel Angel Bermeo-Ayerbe, Carlos Ocampo-Martinez, and Javier Diaz-Rozo. “Data-driven energy prediction modeling for both energy efficiency and maintenance in smart manufacturing systems”. en. In: *Energy* 238 (Jan. 2022), p. 121691. ISSN: 03605442. DOI: `10.1016/j.energy.2021.121691`.

[BM13] Vincent Aizebeoje Balogun and Paul Tarisai Mativenga. “Modelling of direct energy requirements in mechanical machining processes”. In: *Journal of Cleaner Production* 41 (2013), pp. 179–186. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2012.10.015`.

[BMWi2015] Federal Ministry for Economic Affairs and Energy (BMWi). *An electricity market for Germany's energy transition.* Tech. rep. Berlin, Germany: Federal Ministry for Economic Affairs and Energy (BMWi), July 2015.

[Box+16] George E. P Box et al. *Time series analysis : forecasting and control.* eng. Fifth Edition. Wiley series in probability and statistics. Hoboken: Wiley, 2016. ISBN: 1118675029.

[Bra+21] Aleksey Bratukhin et al. "Integrating uncertainty of available energy in manufacturing planning". In: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).* 2021, pp. 1–4. DOI: `10.1109/ETFA45728.2021.9613343`.

[Bri+21] Markus Brillinger et al. "Energy prediction for CNC machining with machine learning". en. In: *CIRP Journal of Manufacturing Science and Technology* 35 (Nov. 2021), pp. 715–723. ISSN: 17555817. DOI: `10.1016/j.cirpj.2021.07.014`.

[Cha+92] Anantha P Chandrakasan, Samuel Sheng, and Robert W Brodersen. "Low-power CMOS digital design". In: *IEICE Transactions on Electronics* 75.4 (1992), pp. 371–382.

[Che+14] Xuyue Chen et al. "Real-Time Prediction and Optimization of Drilling Performance Based on a New Mechanical Specific Energy Model". en. In: *Arabian Journal for Science and Engineering* 39.11 (Nov. 2014), pp. 8221–8231. ISSN: 1319-8025, 2191-4281. DOI: `10.1007/s13369-014-1376-0`.

[Che+22] Jinchao Chen et al. "Energy-aware scheduling for dependent tasks in heterogeneous multiprocessor systems". en. In: *Journal of Systems Architecture* 129 (Aug. 2022), p. 102598. ISSN: 13837621. DOI: `10.1016/j.sysarc.2022.102598`.

[Del+16] Vasco Delgado-Gomes, José A. Oliveira-Lima, and João F. Martins. "Energy consumption awareness in manufacturing and production systems". en. In: *International Journal of Computer Integrated Manufacturing* (May 2016), pp. 1–12. ISSN: 0951-192X, 1362-3052. DOI: `10.1080/0951192X.2016.1185154`.

[DG04] *An Environmental Analysis of Machining.* Vol. Manufacturing Engineering and Materials Handling Engineering. ASME International Mechanical Engineering Congress and Exposition. Nov. 2004, pp. 643–652. DOI: `10.1115/IMECE2004-62600`.

[DV08] Anton Dietmair and Alexander Verl. "Energy consumption modeling and optimization for production machines". In: *2008 IEEE International Conference on Sustainable Energy Technologies*. 2008, pp. 574–579. DOI: `10.1109/ICSET.2008.4747073`.

[E-C19] E-Control. *Ausfall- und Störungsstatistik für Österreich 2019 [Failure and disruption statistics for Austria 2019]*. Wien, June 2019.

[E-C22] E-Control. *Ausfall- und Störungsstatistik Strom für Österreich 2022. Ergebnisse für das Jahr 2021 [Electricity outage and disruption statistics for Austria 2022. Results for 2021]*. Energie-Control Austria für die Regulierung der Elektrizitäts- und Erdgaswirtschaft, 2022. URL: `https://www.e-control.at/documents/1785851/1811582/Strom-AuSS-Veroeffentlichung-2022-fuer-Berichtsjahr-2021.pdf/bd965c19-d8f2-926f-b715-3b7c351ac75e?t=1663220603901` (visited on 01/17/2024).

[Eur18] Council of European Energy Regulators. *CEER Benchmarking Report 6.1 on the Continuity of Electricity and Gas Supply. Data update 2015/2016*. Brussels, July 2018.

[Far+22] Glen G Farivar et al. "Grid-connected energy storage systems: State-of-the-art and emerging technologies". In: *Proceedings of the IEEE* (2022).

[Gad+21] Michele Gadaleta et al. "Extensive experimental investigation for the optimization of the energy consumption of a high payload industrial robot with open research dataset". In: *Robotics and Computer-Integrated Manufacturing* 68 (Apr. 2021), p. 102046. ISSN: 0736-5845. DOI: `10.1016/j.rcim.2020.102046`.

[Gah+16] Christian Gahm et al. "Energy-efficient scheduling in manufacturing companies: A review and research framework". In: *European Journal of Operational Research* 248.3 (2016), pp. 744–757. ISSN: 0377-2217. DOI: `10.1016/j.ejor.2015.07.017`.

[Gon+15] Xu Gong et al. "An Energy-Cost-Aware Scheduling Methodology for Sustainable Manufacturing". en. In: *Procedia CIRP* 29 (2015), pp. 185–190. ISSN: 22128271. DOI: `10.1016/j.procir.2015.01.041`.

[Gud+12] Nikhil Gudi, Lingfeng Wang, and Vijay Devabhaktuni. "A demand side management based simulation platform incorporating heuristic optimization for management of household appliances". In: *International Journal of Electrical Power & Energy Systems* 43.1 (2012), pp. 185–193.

[Gut+06] Timothy Gutowski, Jeffrey Dahmus, and Alex Thiriez. “Electrical energy requirements for manufacturing processes”. In: *13th CIRP international conference on life cycle engineering*. Vol. 31. 1. Leuven, Belgium. 2006, pp. 623–638.

[He+20] Yan He et al. “A generic energy prediction model of machine tools using deep learning algorithms”. en. In: *Applied Energy* 275 (Oct. 2020), p. 115402. ISSN: 03062619. DOI: 10.1016/j.apenergy.2020.115402.

[Her+21] Juan Heredia, Christian Schlette, and Mikkel Baun Kjaergaard. “Data-Driven Energy Estimation of Individual Instructions in User-Defined Robot Programs for Collaborative Robots”. In: *IEEE Robotics and Automation Letters* 6.4 (Oct. 2021), pp. 6836–6843. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3094781.

[HS23] Simon Howind and Thilo Sauter. “Modeling Energy Consumption of Industrial Processes with Seq2Seq Machine Learning”. In: *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*. IEEE, June 2023. DOI: 10.1109/isie51358.2023.10228118.

[Kel+12] Karel Kellens et al. “Methodology for systematic analysis and improvement of manufacturing unit process life cycle inventory (UPLCI) CO2PE! initiative (cooperative effort on process emissions in manufacturing). Part 2: case studies”. In: *The International Journal of Life Cycle Assessment* 17.2 (2012), pp. 242–251. ISSN: 0948-3349. DOI: 10.1007/s11367-011-0352-0.

[Kub17] Miroslav Kubat. *An Introduction to Machine Learning*. Springer International Publishing, 2017. ISBN: 9783319639130. DOI: 10.1007/978-3-319-63913-0.

[Kum+20] Ankur Kumar, Apratim Bhattacharya, and Jesus Flores-Cerrillo. “Data-driven process monitoring and fault analysis of reformer units in hydrogen plants: Industrial application and perspectives”. en. In: *Computers & Chemical Engineering* 136 (May 2020), p. 106756. ISSN: 00981354. DOI: 10.1016/j.compchemeng.2020.106756.

[Le+13] Cao Vinh Le et al. “Classification of energy consumption patterns for energy audit and machine scheduling in industrial manufacturing systems”. In: *Transactions of the Institute of Measurement and Control* 35.5 (2013), pp. 583–592. DOI: 10.1177/0142331212460883.

[LH23] Dominik Leherbauer and Peter Hehenberger. “Modeling the Energy Flexible Job Shop with a Disaggregated Load Approach for Changeable Manufacturing”. en.

In: *Procedia Computer Science* 217 (2023), pp. 1225–1233. ISSN: 18770509. DOI: `10.1016/j.procs.2022.12.321`.

[Li+11] Wen Li et al. “An Investigation into Fixed Energy Consumption of Machine Tools”. In: *Glocalized Solutions for Sustainability in Manufacturing*. Ed. by Jürgen Hesselbach and Christoph Herrmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 268–273. ISBN: 978-3-642-19692-8.

[Liu+15] Z.Y. Liu et al. “Energy Consumption Characteristics in Finish Hard Milling of Tool Steels”. In: *Procedia Manufacturing* 1 (2015). 43rd North American Manufacturing Research Conference, NAMRC 43, 8-12 June 2015, UNC Charlotte, North Carolina, United States, pp. 477–486. ISSN: 2351-9789. DOI: `10.1016/j.promfg.2015.09.007`.

[Liu20] Hui Liu. *Non-intrusive Load Monitoring: Theory, Technologies and Applications*. en. Singapore: Springer Singapore, 2020. ISBN: 978-981-15-1860-7. DOI: `10.1007/978-981-15-1860-7`.

[May+17] Gökan May et al. “Energy management in manufacturing: From literature review to a conceptual framework”. In: *Journal of Cleaner Production* 167 (2017), pp. 1464–1489. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2016.10.191`.

[MB18] Cristina Morariu and Theodor Borangiu. “Time series forecasting for dynamic scheduling of manufacturing processes”. In: *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. Cluj-Napoca: IEEE, May 2018, pp. 1–6. ISBN: 978-1-5386-2205-6. DOI: `10.1109/AQTR.2018.8402748`.

[Men+19] Roberto Menghi et al. “Energy efficiency of manufacturing systems: A review of energy assessment methods and tools”. en. In: *Journal of Cleaner Production* 240 (Dec. 2019), p. 118276. ISSN: 09596526. DOI: `10.1016/j.jclepro.2019.118276`.

[MH20] Victoria Jayne Mawson and Ben Richard Hughes. “Deep learning techniques for energy forecasting and condition monitoring in the manufacturing sector”. en. In: *Energy and Buildings* 217 (June 2020), p. 109966. ISSN: 03787788. DOI: `10.1016/j.enbuild.2020.109966`.

[PG17] Josh Patterson and Adam Gibson. *Deep Learning*. en. Sebastopol, CA: O’Reilly Media, Sept. 2017.

[Rib+20] Andrea Maria N. C. Ribeiro et al. “Short-Term Firm-Level Energy-Consumption Forecasting for Energy-Intensive Manufacturing: A Comparison of Machine Learning and Deep Learning Models”. en. In: *Algorithms* 13.11 (Oct. 2020), p. 274. ISSN: 1999-4893. DOI: `10.3390/a13110274`.

[Ric11] John T.E. Richardson. “Eta squared and partial eta squared as measures of effect size in educational research”. In: *Educational Research Review* 6.2 (2011), pp. 135–147. ISSN: 1747-938X. DOI: `https://doi.org/10.1016/j.edurev.2010.12.001`.

[SB18] Luzi Sennhauser and Robert Berwick. “Evaluating the Ability of LSTMs to Learn Context-Free Grammars”. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Ed. by Tal Linzen, Grzegorz Chrupała, and Afra Alishahi. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 115–124. DOI: `10.18653/v1/W18-5414`. URL: `https://aclanthology.org/W18-5414`.

[Sch+15] Christopher Schmidt et al. “A methodology for customized prediction of energy consumption in manufacturing industries”. In: *International Journal of Precision Engineering and Manufacturing-Green Technology* 2.2 (2015), pp. 163–172. ISSN: 2288-6206. DOI: `10.1007/s40684-015-0021-z`.

[See+19] J-P. Seevers et al. “Automatic Time Series Segmentation as the Basis for Unsupervised, Non-Intrusive Load Monitoring of Machine Tools”. en. In: *Procedia CIRP* 81 (2019), pp. 695–700. ISSN: 22128271. DOI: `10.1016/j.procir.2019.03.178`.

[Shr+14] Fadi Shrouf et al. “Optimizing the production scheduling of a single machine to minimize total energy consumption costs”. In: *Journal of Cleaner Production* 67 (2014), pp. 197–207. ISSN: 0959-6526. DOI: `10.1016/j.jclepro.2013.12.024`.

[SN19] Dipayan Sarkar and Vijayalakshmi Natarajan. *Ensemble Machine Learning Cookbook*. Birmingham, England: Packt Publishing, Jan. 2019.

[SR11] Y. Seow and S. Rahimifard. “A framework for modelling energy consumption within manufacturing systems”. In: *CIRP Journal of Manufacturing Science and Technology* 4.3 (2011). Production Networks Sustainability, pp. 258–264. ISSN: 1755-5817. DOI: `10.1016/j.cirpj.2011.03.007`.

[Sut+14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to sequence learning with neural networks”. In: *Proceedings of the 27th International Conference on*

Neural Information Processing Systems - Volume 2. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 3104–3112.

[Szé+07] Gábor J. Székely, Maria L. Rizzo, and Nail K. Bakirov. "Measuring and testing dependence by correlation of distances". In: *The Annals of Statistics* 35.6 (2007), pp. 2769–2794. DOI: 10.1214/009053607000000505. URL: https://doi.org/10.1214/009053607000000505.

[Tan+21] Daniel Tan et al. "A three-step machine learning framework for energy profiling, activity state prediction and production estimation in smart process manufacturing". In: *Applied Energy* 291 (2021), p. 116808. ISSN: 03062619. DOI: 10.1016/j.apenergy.2021.116808.

[Vas19] Ivan Vasilev. *Advanced Deep Learning with Python*. Birmingham, England: Packt Publishing, Dec. 2019.

[Wei+11] Nils Weinert, Stylianos Chiotellis, and Günther Seliger. "Methodology for planning and operating energy-efficient production systems". In: *CIRP Annals* 60.1 (2011), pp. 41–44. ISSN: 0007-8506. DOI: 10.1016/j.cirp.2011.03.015.

[WW21] Jessica Walther and Matthias Weigold. "A Systematic Review on Predicting and Forecasting the Electrical Energy Consumption in the Manufacturing Industry". en. In: *Energies* 14.4 (Feb. 2021), p. 968. ISSN: 1996-1073. DOI: 10.3390/en14040968.

[Zha+17] G. Y. Zhao et al. "Energy consumption in machining: Classification, prediction, and reduction strategy". In: *Energy* 133 (2017), pp. 142–157. ISSN: 0360-5442. DOI: 10.1016/j.energy.2017.05.110.

[ZS15] Behnam Zakeri and Sanna Syri. "Electrical energy storage systems: A comparative life cycle cost analysis". In: *Renewable and sustainable energy reviews* 42 (2015), pp. 569–596.

Online Resources

[AVL] AVL List GmbH. *Optimized Battery Industrialization Manufacturing: From Prototypes to Production Process*. URL: https://www.avl.com/en/engineering/e-mobility-engineering/battery-development/battery-industrialization-and-manufacturing (visited on 01/07/2024).

[Cho23] François Chollet. *Character-level recurrent sequence-to-sequence model.* 2023. URL: https://keras.io/examples/nlp/lstm_seq2seq/#run-inference-sampling (visited on 02/01/2024).

[Dun21] Michael Dunlop. *Laser welding 101.* 2021. URL: https://fsmdirect.com/laser-welding-101/ (visited on 01/13/2024).

[Gad+20] Michele Gadaleta et al. *DATA of the paper: "Extensive Experimental Investigation for the Optimization of the Energy Consumption of a High Payload Industrial Robot with Open Research Dataset".* 2020. URL: https://data.mendeley.com/datasets/hdhchngky8/1 (visited on 02/02/2024).

[KUKA20] KUKA Deutschland GmbH. *KR 210 R2700 prime.* 2022. URL: https://www.kuka.com/-/media/kuka-downloads/imported/8350ff3ca11642998dbdc81dcc2ed44c/0000181027_de.pdf (visited on 02/02/2024).

[LH18] Anodyne Lindstrom and Sara Hoff. *Average U.S. electricity customer interruptions totaled nearly 8 hours in 2017.* 2018. URL: https://www.eia.gov/todayinenergy/detail.php?id=37652 (visited on 01/19/2022).

[Mei+23] Gerhard Meister, Wolfgang Novak, and Tom Horvat. *AVL Battery Development Solutions.* Aug. 2023. URL: https://www.avl.com/en/system/files?file=2023-09/avl_battery_solutions_0.pdf (visited on 02/12/2024).

[VSC5] Austrian initiative on high performance computing. *VSC-5 – newest system featuring CPU and GPU.* URL: https://vsc.ac.at//systems/vsc-5/ (visited on 01/15/2024).

Appendix A

Synthetic Data Configuration

Here, the exact profile configurations for the synthetic data generation are shown. Table A.1 lists the interpolation method and basic profile for each recipe ID. The interpolation method indicates whether the "pad" or "nearest" method was used to fill in missing values. The "pad" method replicates the value of the last available element before the gap, while the "nearest" method uses the value closest to the missing value, regardless of its location. The other tables list all the configured dependencies (except for offset dependencies, which were not configured): Height dependencies (Table A.2), repetition dependencies (Table A.3) and profile dependencies (Table A.4). The application of this configuration is described in detail in Section 4.1.2.

Table A.1: Basic profile configuration for synthetic data generation.

recipe_id	**step_name**	**interpolation_ method**	**profile**
0	stacking	nearest	[750, 900]
1	stacking_ to_welding	nearest	[0, 0, 0, 0, 0, 0, 0, 0, 900]
2	welding	nearest	[500, 500, 650, 500, 500, 650, 500, 500, 650]
3	welding_ to_gluing	nearest	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 900]
4	gluing	pad	[500]
5	gluing_ to_packing	nearest	[0, 0, 0, 0, 0, 0, 900]
6	packing	nearest	[200, 500, 200, 500, 200]
7	packing_ to_quality	nearest	[0, 0, 0, 0, 0, 0, 0, 900]
8	quality	pad	[340]
-2	Idle	pad	[1]

Table A.2: Height dependencies configuration for synthetic data generation.

Recipe ID	Step Name	Height Dependencies
0	stacking	–
1	stacking_ to_welding	[{'parameter_name': 'parameter_categorical_preceding_recipe_id', 'factor_dict': {1: 1, 3: 1.1, 5: 0.9, 7: 1, -2: 1}]
2	welding	[{'parameter_name': 'parameter_numerical_welding_temperature', 'min_factor': 1, 'max_factor': 2}]
3	welding_ to_gluing	[{'parameter_name': 'parameter_categorical_preceding_recipe_id', 'factor_dict': {1: 1, 3: 1.1, 5: 0.9, 7: 1, -2: 1}}]
4	gluing	[{'parameter_name': 'parameter_numerical_preceding_duration', 'min_factor': 1, 'max_factor': 5}, {'parameter_numerical_batch_size', 'min_factor': 1, 'max_factor': 5}]
5	gluing_ to_packing	[{'parameter_name': 'parameter_categorical_preceding_recipe_id', 'factor_dict': {1: 1, 3: 1.1, 5: 0.9, 7: 1, -2: 1}}]
6	packing	–
7	packing_ to_quality	[{'parameter_name': 'parameter_categorical_preceding_recipe_id', 'factor_dict': {1: 1, 3: 1.1, 5: 0.9, 7: 1, -2: 1}}]
8	quality	–
-2	Idle	[{'parameter_name': 'parameter_categorical_machine', 'factor_dict': {'stacking_robot_0': 440, 'stacking_robot_1': 440, [...], 'welding_robot_0': 380, 'welding_robot_1': 380, [...], 'gluing_robot_0': 280, 'gluing_robot_1': 280, [...], 'packing_robot_0': 200, 'packing_robot_1': 200, [...], 'testing_station_0': 220, 'testing_station_1': 220, 'trolley_0': 0, 'trolley_1': 0, [...]}}]

Table A.3: Repetition dependencies configuration for synthetic data generation.

Recipe ID	Step Name	Repetition Dependencies
0	stacking	[{'parameter_name': 'parameter_numerical_batch_size', 'min_factor': 1, 'max_factor': 5}]
1	stacking_ to_welding	–
2	welding	[{'parameter_name': 'parameter_numerical_batch_size', 'min_factor': 1, 'max_factor': 5}]
3	welding_ to_gluing	–
4	gluing	[{'parameter_name': 'parameter_numerical_batch_size', 'min_factor': 1, 'max_factor': 5}]
5	gluing_ to_packing	–
6	packing	[{'parameter_name': 'parameter_numerical_batch_size', 'min_factor': 1, 'max_factor': 5}]
7	packing_ to_quality	–
8	quality	[{'parameter_name': 'parameter_numerical_batch_size', 'min_factor': 1, 'max_factor': 5}]
-2	Idle	–

Table A.4: Profile dependencies configuration for synthetic data generation.

Recipe ID	**Step Name**	**Profile Dependencies**
0	stacking	–
1	stacking_ to_welding	–
2	welding	–
3	welding_ to_gluing	–
4	gluing	–
5	gluing_ to_packing	–
6	packing	–
7	packing_ to_quality	–
8	quality	–
-2	Idle	[{'parameter_name': 'parameter_categorical_machine', 'factor_dict': {'stacking_robot_0': [1, 1, 1, 0, 0], 'stacking_robot_1': [1, 1, 0, 0], [...], 'welding_robot_0': [1, 1, 0, 0], 'welding_robot_1': [1, 0, 0], [...], 'gluing_robot_0': [1, 1, 0, 0], 'gluing_robot_1': [1, 1, 0, 0], [...], 'packing_robot_0': [1, 0, 0], 'packing_robot_1': [1, 0, 0], [...], 'testing_station_0': [1, 0, 0], 'testing_station_1': [1, 0, 0], [...], 'trolley_0': [0], 'trolley_1': [0], [...]}}]

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Copyright Statement

I, Simon Howind, hereby declare that this thesis is my own original work and, to the best of my knowledge and belief, it does not:

- Breach copyright or other intellectual property rights of a third party.
- Contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
- Contain material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.
- Contain substantial portions of third party copyright material, including but not limited to charts, diagrams, graphs, photographs or maps, or in instances where it does, I have obtained permission to use such material and allow it to be made accessible worldwide via the Internet.

Signed by:	Simon Howind
Date:	28.02.2024 15:12:01

TRUST
This document is digitally signed!
This document bearing a qualified electronic signature has the same legal standing as a handwritten document in accordance with Article 25 (2) of Regulation (EU) No. 910/2014 of 23 July 2014 ("eIDAS VO")
Validation: Information for checking the electronic signature can be found at: www.a-trust.at/pdf

Signature: ______________________

Vienna, Austria, February 2024 Simon Howind