TÉCNICO LISBOA

Building heat disaggregation with deep learning

Kaarlo Lauri Johannes Forsman

Thesis to obtain the Master of Science Degree in

Energy Engineering and Management

Supervisor: Prof. Carlos Augusto Santos Silva

Examination Committee

Chairperson: Prof. Edgar Caetano Fernandes
Supervisor: Prof. Carlos Augusto Santos Silva
Member of the Committee: Prof. Susana Margarida da Silva Vieira

October 2023

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to the supervisors of this work Professor Carlos Augusto Santos Silva, my supervisor from Instituto Superior Técnico, as well as Professor Risto Lahdelma from Aalto University for their insightful comments and suggestions, which have greatly contributed to the quality of this study.

I am grateful to Sweco Finland Oy for commissioning this thesis and for providing me with the necessary resources and support to carry out this research. I would like to express my special thanks to Joni Hilpinen, my company supervisor, for his invaluable guidance, encouragement, and support throughout the project.

Finally, I would like to extend my heartfelt thanks to my family, my mother, father, brother, and girlfriend, for their unwavering support, encouragement, and love.

ABSTRACT

Energy retrofit projects for existing buildings aim to improve energy acquisition costs and overall energy efficiency. Informed energy investment decisions require extensive data to achieve optimal results. Overall energy consumption provides only a partial picture of the information required and consumption data segmented by energy end-use is needed to make informed investment decisions. However, detailed consumption data is rarely available and only overall energy consumption time series are available. Engineering consultancies receive technical information, architectural drawings and building parameters as part of project initial data. Based on this information, a simulation model can be constructed to obtain the disaggregated consumption data. However, developing a simulation model is time and resource intensive.

In this study, a data-driven model for building heat disaggregation task was developed. A literature review of data-driven models in energy disaggregation and energy prediction and forecasting was conducted. Machine learning algorithms were found to achieve good performance in disaggregation and prediction tasks in the energy domain with neural networks exhibiting best results. A dataset was constructed from the simulations of the building design tool IDA ICE. The dataset was analyzed and engineered into a training dataset for data-driven model development.

A variety of neural network architectures were trained on the heat disaggregation task. Feedforward network with four hidden layers and 127 004 trainable parameters achieved the best validation performance with MAE of 0.0814, RMSE of 0.1388 and r2-score of 0.704. The performance of the model was evaluated with respect to energy industry model validation methodologies as well as with respect to the end-use case of the model. The model was assessed to perform the heat energy disaggregation task sufficiently. High maximum errors as well as a relatively low r2-score are key factor to reconsider if the model is to be deployed in other than preliminary assessment tasks.

Key-words: machine learning, deep learning, energy, building, disaggregation

RESUMO

Os projetos de modernização energética de edifícios existentes visam melhorar os custos de aquisição de energia e a eficiência energética global. Decisões informadas de investimento em energia requerem dados extensos para alcançar resultados ideais. O consumo global de energia fornece apenas uma imagem parcial da informação necessária e são necessários dados de consumo segmentados por utilização final de energia para tomar decisões de investimento informadas. No entanto, raramente estão disponíveis dados detalhados sobre o consumo e apenas estão disponíveis séries cronológicas globais do consumo de energia. As consultorias de engenharia recebem informações técnicas, desenhos arquitetônicos e parâmetros construtivos como parte dos dados iniciais do projeto. Com base nessas informações, um modelo de simulação pode ser construído para obter os dados de consumo desagregados. No entanto, desenvolver um modelo de simulação exige muito tempo e recursos. Neste estudo, foi desenvolvido um modelo baseado em dados para a tarefa de desagregação de calor em edifícios. Foi realizada uma revisão da literatura de modelos baseados em dados em desagregação energética e predição e previsão de energia. Descobriu-se que algoritmos de aprendizado de máquina alcançam bom desempenho em tarefas de desagregação e predição no domínio de energia, com redes neurais exibindo melhores resultados. Um conjunto de dados foi construído a partir de simulações da ferramenta de projeto de edifícios IDA ICE. O conjunto de dados foi analisado e transformado em um conjunto de dados de treinamento para desenvolvimento de modelo baseado em dados. Uma variedade de arquiteturas de redes neurais foi treinada na tarefa de desagregação de calor. A rede feedforward com quatro camadas ocultas e 127.004 parâmetros treináveis alcançou o melhor desempenho de validação com MAE de 0,0814, RMSE de 0,1388 e pontuação r2 de 0,704. O desempenho do modelo foi avaliado em relação às metodologias de validação do modelo da indústria de energia, bem como em relação ao caso de uso final do modelo. O modelo foi avaliado para realizar suficientemente a tarefa de desagregação de energia térmica. Erros máximos elevados, bem como uma pontuação r2 relativamente baixa, são fatores-chave a reconsiderar se o modelo for implementado em tarefas que não sejam de avaliação preliminar

Palavras-chave: aprendizado de máquina, aprendizado profundo, energia, construção, desagregação

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Energy consumption in buildings represent a large portion of overall energy consumption and the associated greenhouse gases. This is especially true for countries in the northern latitudes, where heating is critical for long periods of the year. In Finland, the energy consumption of buildings represents approximately 40 % of final energy consumption and approximately 30 % of overall greenhouse gas emissions [1]. Heating represents over 65 % share of energy use in buildings or 27 % of overall energy use in Finland, respectively [2]. Enhancing the energy efficiency of existing building stock and promoting energy efficient and renewable production solutions in new developments plays a significant role in reducing the national greenhouse gas emissions.

Accurate building energy consumption forecasting, and consumption profiles are a necessary requirement for buildings energy efficiency estimation. They play a vital role in evaluating the current and potential state of energy efficiency in buildings and thus are an integral part of energy efficiency investment decision making as inaccurate forecasts can lead to increased energy costs and suboptimal greenhouse gas emission savings [3]. Engineering and architecture consulting companies utilize detailed simulation software as a part of building energy system design process at various stages. The simulation software (i.e. IDA-ICE, Energy Plus, TRNSYS & VIP Energy) consist of white-box or physics-based simulation models that make the energy balance calculations by utilizing physics-based specifically engineered formulas. These models may provide accurate estimates but require a vast amounts of detailed input data about the buildings structure and the HVAC systems [4], [5]. However, in cases where detailed information about the building is not available or the nature of the design assignment does not require an extensive analysis of the building's energy balance, constructing a building simulation model might be impractical.

One of identified key tasks often faced by engineering consultancies in the building energy domain is to develop energy retrofit projects for existing buildings. Accuracy of the information about the building varies and a detailed consumption profile is rarely available. The overall heat and electricity consumption is often known, but a disaggregated consumption profile with different end-use cases differentiated is not available. This information is crucial especially for heat end-use in order to optimize the energy efficiency measures.

For new development projects, these disaggregated consumption series for heat end-use are available from the building simulation outputs. However, retrofit project for existing building do not have pre-existing simulation models in contrast to new developments, where a detailed simulation model is constructed as part of the building design process itself. Constructing a simulation model for

an existing building in order to obtain approximations of disaggregated heat consumption data is not ideal. In many cases, the time intensive simulation construction is too resource intensive. Retrofit projects also often suffer from a lack of detailed information regarding for example structural parameters and user interaction in the building. These factors combined often result in an approximation that is not as accurate as an accurately informed simulation can offer and yet is very time intensive to obtain. In this way, developing a model specialized in this disaggregation task would provide sufficiently accurate disaggregated heat consumption data and would significantly reduce the time intensity of the task [6].

Data-driven models have been proposed as an alternative for the described tasks, where the use of simulation software is suboptimal. As an alternative, data-driven models can be less intensive to implement for a specific task. In the domain of data-driven models, machine learning models are a powerful subset of algorithms that are used to leverage existing data. Machine learning models can be implemented for tasks, where sufficient data is available [4]. The task of disaggregating consumption data can be considered suitable for machine learning algorithms. As a part of SWECO´s (engineering & architectural consultancy) project portfolio, an extensive collection of simulation models for delivered projects is available [6]. Pre-existing simulation models and the respective accurate deconstructed heat consumption time-series can be used for the development of data-driven models for tasks, such as heat consumption disaggregation.

The objective of this study is to develop a data-driven model for building heat disaggregation in the context of the SWECO project. A comprehensive review of building heating systems and machine learning techniques is conducted. Additionally, a literature review to explore data-driven models used in energy prediction, forecasting, and heat disaggregation tasks is conducted. A synthetic dataset based on IDA ICE building energy simulations is constructed, analyzed, and engineered to be utilized as the training data for data-driven model development. Data-driven models are trained and evaluated with respect to the end-use case.

2 Foundations

The foundations of this work in both the fields of machine learning and energy technology, are examined in this chapter. Energy services provided by building heating and ventilation systems are discussed with respect to four heating end-use categories. The fundamental concepts behind traditional machine learning and deep learning are described focusing especially on the models used in this work. Metrics for evaluating the accuracy and performance of the models are described and discussed.

2.1 Heating in buildings

Energy use in buildings is a sum of complex interplay between the physical properties of the structure, the dynamically changing conditions such as weather and human actions and the buildings energy systems. As a part of buildings functionality, it needs to facilitate safe and comfortable conditions by utilizing resource efficiently. Good indoor air quality and stable indoor temperature are central to achieving functional operating conditions. The management of indoor air quality and temperature can be improved by architectural and structural design. One of the primary considerations is the geometry of the building itself. Orientation of the building on the site as well as material choices, especially amount of insulation, play a significant role in the properties of the building. Sustaining the suitable conditions is done with the buildings heating, ventilation, and air conditioning systems as well as water supply system. The overall final energy use in buildings consist of electrical energy and heat energy. The scope of this work focuses on the quantities and classification of supplied heat, but it is important to note that the electricity consumed in buildings will also partially or even entirely be converted into heat, thus having a strict correlation with heating and cooling demand [7].

An energy system of larger buildings can and often is a complex array of various HVAC technologies and support systems that use both supplied electricity and heat to sustain determined conditions in the buildings distinctive spaces. The heat source of a building can consists of various technologies. In 2018 the most common heating method used for buildings in Finland was district heating representing a share of over 45 %. District heating is even more common for larger residential buildings as well as commercial and public buildings. In 2021 district heating represented a share of over 85 % in residential apartment buildings in Finland [2], [8]. In addition to district heating, heat pumps, electrical heating, biofuel based, and fossil fuel based heating are used to lesser extent [2]. Recently, heat pumps and especially geothermal heat pumps have increased their share as an

economically and environmentally competitive alternative to district heating. Also, hybrid systems are common, with for example geothermal heat pump providing majority of the annual heat demand and district heating connection assisting in peak load scenarios, where the power of the heat pump system is not sufficient [9].

Generally, the heat demand of the entire HVAC systems can be divided into four categories: domestic water heating, domestic hot water circulation losses, space heating and ventilation system intake air heating. These specific categories sum to the overall heat demand of the building, which is provided by the prevalent heat supply technology. Heat consumption data is often collected on the system level [10]. This level of accuracy is generally not optimal when examining distinctive energy efficiency measurements or energy system investments. Additionally, this level of accuracy differentiates from the energy analysis data provided by simulation and design software used. The heat demand categories utilize distinctive parts of the buildings overall energy system. In the following chapter, the four categories are examined and defined. It is important to note, that the output classes of the AI models developed in this work are representations of these heat consumption categories.

2.1.1 Domestic water heating & circulation

The domestic hot water (DHW) system of a building delivers both cold and hot water for the users. Since generally only cold water is supplied to the system, a water heating system is needed. DHW energy demand refers to the amount of energy that is needed to increase the temperature of the cold water to a suitable level for the buildings end-users. The hot water temperatures vary from system to system, but generally temperatures lower than 55 °C are avoided to prevent Legionella growth [11]. The temperature difference between cold and hot water is generally approximately 50 °C [12].

The main factor affecting the DHW energy usage is the user water usage. The water usage profile of building users depends highly on the function of the building itself – water consumption for a resident in a residential building is completely different from the water usage of a student in a public elementary school. The Finnish building energy and heating power requirements evaluation legislation (Rakennuksen energiakulutuksen ja lämmitystehotarpeen laskenta, [12]) provides guidelines for DHW energy usage evaluations for residential as well as commercial & public buildings. The guidelines can be used if a more comprehensive evaluation is not feasible. Water usage in commercial and public buildings can be evaluated with respect to gross square area if the building use-case is known. In residential buildings, hot water usage can be evaluated with respect to the number of residents. The guideline values for the estimations are presented in Table 1.

Table 1. DHW energy use estimation guidelines [12]**.**

Building type	**Hot water use per capita (litres/day)**
Residential building (individual metering & billing)	50
Residential building (other)	60
Building type	**Hot water use per gross square area (litres/gross-m² per year)**
Residential building	600
Office building	100
Hospital / healthcare facility	520
Day-care facility	460
Theatre / library	120
Swimming hall	1800
School / teaching facility	180
Business space	65

The hot water is distributed with a closed-loop circulation system, that allows the heated water to circulate inside the buildings water distribution system. With a circulation system, hot water is available faster for the end-user even if the distance between the water heating unit and end-user is large. Without the circulation, hot water would potentially sit and cool down between hot water demand periods. The overall consumption of water is greatly reduced when there is no need to flush cooled water from the pipes from the way of sufficiently heated water. When hot water use is low, the hot water circulates through the loop uninterrupted. The heat that's transferred from the hot water during the circulation process is referred to as DHV circulation losses. The share of DHV circulation losses varies with respect to building use-cases and the resulting hot water demand profile. In buildings, where the hot water consumption relative to building size is the lowest, the circulation losses share of overall DHW heat usage is highest. Examples of such cases are office and commercial buildings, where the building operation times cover only part of the day and hot water demand is relatively low. In contrast, circulation losses are lowest for buildings with high hot water demand and even hot water consumption profile, for example residential buildings. DHW circulation losses can represent a significant share of 50 – 90 % of DHW system heat demand in office buildings, while the share only approximately 30 % in residential buildings. It is important to note that the circulation losses cannot be defined as lost energy in cases, where space heating throughout the building is needed. The dissipated heat from the DHW circulation network is partially out to used spaces therefore mitigating the equivalent energy in space heating demand. However, the effect of this impact can be fairly limited – the heat is only utilized during cold periods, the circulation pipes are often located outside the building envelope for utilitarian reasons. The utilization share of the DHW circulation losses can be as low as 25 %. However, determining accurate evaluations for DHW losses and DHW system efficiency can be problematic, if the building geometry and prevailing dynamic conditions are not taken into account [11].

According to the Finnish building energy and heating power requirements evaluation legislation (Rakennuksen energiakulutuksen ja lämmitystehotarpeen laskenta, [12]) the DHW circulation losses can be evaluated with respect to gross square area or circulation water mass flow. The legislation provides guidelines for evaluating the DHW losses to use in case there is no possibility of conducting a comprehensive evaluation. The guideline values are presented in Table 2.

Table 2. DHW circulation energy use estimation guidelines [12]**.**

Building type	**Specific DHW circulation losses (kWh/ gross-m² per year)**
Residential buildings	
- wet room heating circuits not connected to DHW circulation	15
- wet room heating circuits connected to DHW circulation	30
Other buildings	
- wet room heating circuits not connected to DHW circulation	7
- wet room heating circuits not connected to DHW circulation	15

DHW energy use is a dynamic attribute that is affected by the building user's hot water demand. The hot water demand profile is the sum of all buildings user's hot water use at given times. The demand profile is often well represented by a weekly cyclic distribution. In the IDA ICE simulation software, the daily cycles are represented by defining a daily profile, that is extended further for a weekly cycle, where Saturday and Sunday profiles can be explicitly set to either match or differ from the daily cycle. DHV circulation losses remain nearly constant throughout the year. However, as can be seen from Table 2, the circulation losses are higher for buildings with high hot water use. Unlike space heating and
ventilation intake air heating, the DHW heating and circulation losses are not strictly correlated to outside temperatures [13].

2.1.2 Space heating

Space heating & cooling systems are used to control temperature in the buildings. In this work, the focus is on heat energy use and thus the space temperature control is assumed to only be capable of adding energy into the system. Space heating aims to supply the system with equivalent amount of energy that the overall system losses are. In situations where outside temperature is higher than the temperature inside the buildings, buildings lose heat energy as the heat is conducted and dissipated away from the structures.

Temperature control is a dynamic task that depends not only on environmental factors but also on building user actions. Factors such as the number of people utilizing the space and their distribution throughout given time frames. In addition to heat generated by the people themselves, the use of all

electrical appliances & other energy use inside the buildings affects the space heating power required at any given instance.

According to the Finnish building energy and heating power requirements evaluation legislation (Rakennuksen energiakulutuksen ja lämmitystehotarpeen laskenta, [12]) space heating energy requirements can be evaluated if the thermal transmittance of structures and their respective areas are known in addition to the temperature difference between the space and the adjacent spaces (i. e. other spaces or outside temperature). The number of users and waste heat generated by electrical appliances decrease the space heating demand. Heat radiation from sunlight also decreases space heating demand and the effect is affected by buildings orientation, geometry and window area.

Energy for space heating can be supplied to the building as either electricity, heat or fuel depending on the buildings heating system. Electric heating and boilers can be used, but district heating is the most common heat supply for large buildings in residential, public and commercial use. The prevalent heating technology is used to heat up water (or potentially in rare cases other heat transfer fluid) in a closed loop. Closed loop system transports the heat to all required spaces. In the case of electric heating, no closed loop circulation system is required. The heat is dissipated into the space most commonly by using radiators and/or underfloor heating. Radiator inlet temperatures are most commonly in the range of 45 – 90 °C and outlet temperatures range from 35 – 70 °C, respectively. Underfloor heating temperatures are often lower with inlet temperatures of 40 °C and outlet temperature of 35 °C. The inlet temperatures are generally increased when outside temperature decreases and vice versa. Losses in the space heating system are comprised of distribution losses, delivery losses, control losses and heat development losses. Distribution losses account for the heat lost in the closed loop circulation, delivery losses account the efficiency of the heat delivery technology (i.e. radiator or underfloor heating) and control losses account for the heat lost due to operation when the system reacts to dynamically changing environment unoptimally. Heat development losses account for the efficiency of the prevalent heat development technology, such as district heating heat exchangers. Potential buffer tank losses should also be included if such technology is included in the system [12].

2.1.3 Ventilation system intake air heating

Ventilation systems ensure the quality of indoor air. Sufficient ventilation improves comfort, reduces negative effects on health and prevents structural problems caused by moisture. Passive ventilation methods are ventilation systems that rely solely on pressure differences, wind and density differences occurring inside the structure with respect to conditions outside of the structure.

Mechanical ventilation is utilized for larger buildings and offers higher control over ventilation and is often utilized in larger buildings due to the higher ventilation requirements. Mechanical ventilation can be arranged by either exhaust ventilation, exhaust-intake ventilation or gravity driven ventilation.

Heating of intake air is an efficient way to reduce energy consumption. In low temperatures, the intake air is cold and without pre-heating, the cold air needs to be heated by increasing space heating load to counter the decrease of inside temperature that the flow of low temperature air causes. To further increase overall efficiency, excess heat from the warm exhaust ventilation can be captured and used to pre-heat the intake air. If no heat capture is implemented, the exhaust ventilation flow wastes significant amounts of energy as the heat energy captured in the room temperature exhaust air exists the building [12].

The overall flow of ventilation is one of key factors when describing building ventilation system. The maximum ventilation capacity does not necessarily represent the overall ventilation needs, since the ventilation system schedule is often altered with respect to the building use profile. An apartment building for example can have the ventilation systems operating at a steady rate throughout the day and over weekends and vacation days since the user profile is relatively uniform. In contrast, an office space ventilation can be active only when the building is used at a significant capacity during office working hours for example, from 6.00 to 19.00 from Monday to Friday. In addition to differences in full ventilation schedule, ventilation system can remain operating at decreased capacity even though building use would be low to ensure suitable conditions for structural factors and indoor air quality handling [6].

2.2 Machine Learning

Building heat consumption modelling and forecasting has been a popular field of study, due to its large importance. Since 1980s, various models with highly differentiating structures and operational philosophies have been proposed, and especially data-driven machine learning based models have gained popularity. Data-driven models consist of a family of algorithms that aim to learn the patterns from the given data. Recent years have seen a rapid development in deep learning algorithms across various fields in general, as well as in consumption models [14].

An ML algorithm is an algorithm capable of acquiring knowledge from data. Learning in this context involves a computer program that acquires knowledge through experience in a particular set of tasks compared to specific performance metrics. As experience accumulates, the algorithm's proficiency in executing tasks improves. Various kinds of tasks, performance measures, and experiences can be incorporated into machine learning algorithms. The experience the model accumulates is related to the used data the model receives, which can be continuous and dynamic or a predefined set of data instances i.e., examples. Classifications such as "pre-trained" and "dynamic" models are often used to describe the way model gains experience [15].

2.2.1 The tasks

Generally, some tasks are very easy to perform admirably with fixed computer programs. Similarly, some tasks that are trivial for humans are excruciatingly difficult to program using traditional fixed computer algorithms. Machine learning models can be thought of to approach problems similarly to what humans do – by learning to do a task by learning from experience. The overall experience the machine learning model is exposed to is the overall set of examples that the model learns from during the training phase. A single example or data instance is typically presented as a vector $\boldsymbol{x}$, which consists of all the properties and characteristics related to that instance – features. Machine learning tasks are usually categorized and described in terms of what the model is expected to process an example [16].

- **Classification** model aims to group the given inputs into a set of classes. Binary classification divides the instances between two classes, while multi-class classification problem aims to divide the instances to more numerous sets of classes. A very common variate of a classification task is one where the resulting f outputs a probability distribution over the given classes. Energy consumption forecasts tasks, where overall consumption is divided between sources, is a classification tasks.
- **Regression** is a task, where the model outputs a numerical value given the input $\boldsymbol{x_i}$. Regression is fairly similar to classification, but the output format is different. Ordinal regression is a regression task, where one or more of the features is an ordinal variable. Ordinal variables are categorical variables for which the specified values can be arranged and ordered, thus having elements from both quantitative and categorical variables.
- **Transcription** are tasks where machine learning system aims to transform an unstructured set of data into textual form. Speech recognition is an example of a transcription task.
- **Machine translation** is a task where an input sequence s converted into another sequence.

- **Anomaly detection** looks through a set of examples and tries to identify if some of them are atypical.
- **Synthesis and sampling** are a form of generative tasks in which the algorithm creates new instances that are similar to the instances. If the model aims to generate only some missing values or instances, this task is often referred to as **imputation of missing values**.
- **Denoising** is a task, where the model aims to distinguish, which datapoints are corrupted and which are not.
- **Density estimation** is a task, where the algorithm aims to capture the structure of the given data and learn a probability distribution that represents it. This task can be used as a step to solve otherwise difficult tasks, such as imputation of missing values.

It is important to note that the spectrum of tasks that machine learning algorithms can learn is vast and might not fit the categories specified below. To monitor how the model is performing the task it was specified to do, one needs to define a performance metric to match the specified task. [16]Energy consumption forecasting are commonly regression tasks, which is one of the most prevalent and classic machine learning tasks. The problem and therefore task examined in this work, is however considered to be a classification task since model predicts the share of consumption from a known overall consumption instead of predicting the real consumption values. As can be seen later, this distinct quality of the task affects the model architecture and performance metric considerations.

To establish how well a model is executing the given task, a performance metric needs to be specified. In general, performance of the model is evaluated by comparing predictions and results provided by the model to labeled data, if labeled data is available. In cases, such as density estimations, this method is not suitable and often the performance is approached from a different perspective, for example by examining the average log-probability the model assigns. The performance measures are evaluated using a distinct set of examples, that have not been used in the training of the model. With this separated test set it is possible to address the performance of the model in general use, in the real-world application [16].

As energy consumption forecast models focused on classification and regression tasks, there are various established performance metrics. Accuracy is one of the most common performance metrics for classification tasks is to simply examine the share of examples that the model classifies right. The inverse, share of examples that are classified wrong, is referred to as error rate as is used in place of accuracy very often. More complex metrics that provide class-specific insight into the model's

behavior, such as precision, recall and F1 score, can be used in multi-class classification tasks. Metrics that measure distances between predictions and actual values are often used in regression type tasks. Different loss functions place a different emphasis to different mistakes – for example single large mistakes (outlier) or several small/medium mistakes [15].

The overall suitability of a performance metric differs from task to task. The suitability can differ from one distinct application to another, even though the high-level task description stays the same. One can examine a trained model with various metrics, but to facilitate training, one metric needs to be defined to construct a loss function for the optimization problem. Different loss functions based on different performance metrics result in different optimization problems. Thus, even with identical settings, model architecture and examples, the choice of performance metric will affect the training and therefore resulting model significantly [15].

2.2.2 Experience

Machine learning algorithms are typically classified into three main categories: "supervised," "unsupervised," and "reinforcement learning." This categorization helps to organize and distinguish between different types of algorithms and portray the way they gather experience.

Supervised learning is a category of machine learning algorithms and models that utilize pre-labelled data, in which the input-output pars are known. In other words, the model experiences a dataset, where each example of features is provided with a label in the training phase. The resulting dataset $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{\boldsymbol{M}}$, where M is the number of annotated training samples. The $\boldsymbol{x_i}$ is the i[th] annotated input vector and corresponds to a label $\boldsymbol{y_i}$. This dataset of input-output pairs is utilized as a training dataset, on which the model learns a mapping, $f: x \to y$. In practice, the overall available dataset is further split for testing and validation, that are used as the basis of performance evaluation [17].

Unsupervised learning, on the other hand, aims to discover interesting patterns in the data, without the use of annotated datasets. Model aims to gather patterns in the unannotated dataset $D^u = \{\boldsymbol{x}\}_{i=1}^{N}$, even though the specific meaning of different patterns and distinctions between data-instances is not predefined and therefore previously invisible to the model. In practice, the specific task of the model is often to learn the entire probability distribution that represents the dataset [17].

Examples of traditional supervised machine learning tasks are classification, regression, and structured output problems. Density estimation, synthesis, and sampling and denoising are traditionally unsupervised machine learning tasks. Supervised and unsupervised learning can be combined Semi-Supervised Learning (SSL) approaches, where the dataset is only partially labeled. By

providing some labeled data together with unlabeled data, the model is hoped to learn the underlying patterns more efficiently. SSL has proved to be especially effective in applications, where gathering labeled data is extremely expensive [15].

Reinforcement learning describes situations, where the model does not experience a fixed dataset, but actively interacts with the environment. When the operational environment of the model is especially complex and difficult to model, some tasks can be easier to execute as a reinforcement learning problem. Reinforcement learning is used to determine how to act or behave when presented with reward or punishment signals. An agent interacts with its environment and learn a way of operating, a policy, that maximizes a defined set of rewards. By trial and error, the agent can learn actions that correlate to rewards and avoid actions which correlate to penalties [17].

Generally, the most relevant machine learning models for energy consumption forecasting are typically supervised learning algorithms. This is because energy consumption data typically has a clear target variable (i.e., the actual energy consumption) that can be used for training and testing a supervised learning model. Time series forecasting models are particularly well-suited to energy consumption forecasting, as they can capture the seasonality and trends in the data. Unsupervised algorithms are also used in the energy consumption domain, but generally for slightly different applications compared to supervised models. Unsupervised learning is used in pattern detection and especially anomality detection but is not typically used as a primary forecasting model. (Runge & Zmeureanu, 2019) Reinforcement learning is not generally suited for energy consumption forecasting since the target variable is not a consequence of the agent's actions. The lack of direct relationship between the agents' actions and the energy consumption conflicts with the base principles of reinforcement learning. However, in energy sector in general, reinforcement learning algorithms have plenty of applications, just not specifically as primary forecast models [16].

2.2.3 Linear & Logistic Regression

Linear and logistic regression algorithms are one of the simplest machine learning algorithms. Linear regression is used for supervised regression tasks while logistic regression is used for supervised classification tasks [16].

Linear regression takes as input a vector $\boldsymbol{x}$ and outputs a scalar $\boldsymbol{y}$. In other words, aim is to fit a linear line or plane, so that it best captures the representations of $\boldsymbol{y}$ with respect to $\boldsymbol{x}$. The output of the linear model is the following:

$$\hat{y} = w^T x + b \tag{1}$$

Where $\hat{y}$ is the transformed output, x is the input vector ($1\ x\ n$), w is a vector of parameters, often referred to as the weight vector, and b as the bias vector. Linear regression, by definition, only implements a linear operation on the input vector by multiplying each element x_i it by the respective weight vector element y_i, and by adding a respective intercept term b [15].

In addition to the underlying model definition, the used performance metric needs to be specified. The mean squared error is one the most common loss functions used with regression tasks and is specified as follows:

$$MSE = \frac{1}{m}\ (\hat{y} - y)_i^2 \tag{2}$$

Mean squared error decreases, when the Euclidean distance of both the predicted scalar $\hat{y}$ and the label of the test or validation set y decreases, and vice versa. (Zhou, 2021) Since the goal is to obtain an optimal result of the linear model with respect to the defined loss function, the aim is to minimize the loss function. This can be achieved by solving where the gradient of MSE is zero or approaching local loss function minima step-by-step. This technique is called iterative optimization and is in some cases the only feasible optimization algorithm, especially with complex models and datasets [16].

Other metrics accuracy more focused on other aspects are also generally used. Root mean-squared error (RMSE) is a commonly used metric that presents the quadratic mean of the differences in predicted values and predicted scalar. R2-score is often used to describe model overall determination. R2-score is defined as the squared value of the correlation coefficient. In the case of multivariate regression, the coefficient of multiple correlation. Different metrics can be calculated for the same model test and validation in order to obtain a broader view of model overall performance [16].

When linear regression can be used for regression tasks, logistic regression is a simple machine learning algorithm used for classification tasks. Logistic regression, despite its name, is a classification model that is used for binary classification tasks. The aim is to fit a sigmoid curve to examples, so that the curve/plane represents the probability of event $\boldsymbol{y}$ happening given the value of $\boldsymbol{x}$ at the point. In a more detailed form, the probability of y given x is given by the sigmoid function with a weighted sum of the input features [15].

$$p\ (\ y = 0|\ \boldsymbol{x}) = \frac{1}{1 + e^{\boldsymbol{w^T x + b}}} \tag{3}$$

Where x is the input vector ($1\ x\ n$), w is a vector of parameters, often referred to as the weight vector, and b as the bias vector. The binary cross-entropy loss is most commonly used as the

performance metric for logistic regression. It measures the difference between the model predicted probabilities for y given x and the true labels [16]. Binary cross-entropy loss can be expressed as the following:

$$Loss_{Binary\ Cross-Entropy}\ (y_i, \hat{y}) = y_i \times \log(\hat{y}_i) + (1 - y_i) \times \log\ (1 - \hat{y}_i) \quad (4)$$

The goal of the training process is to minimize the loss function, which can be achieved by utilizing various optimization algorithms, both analytical and iterative. Even though the trained model predicts probabilities and therefore obtains values between 0… 1, often the model can be used as a strict classifier by defining a decision threshold (for example 0.5) [15].

One of the key limiting factors of standard logistic regression is the fact that it can only learn binary classification tasks. Often the task requires the model to classify the examples into more than two classes. Multi-class classifiers can be constructed by creating an ensemble of binary classifiers. Depending on the application, they can be constructed as One-vs-One, One-vs-All or One-vs-Rest binary classifier systems. In one-vs-one systems, one binary classifier is trained for each pair of classes, resulting in a total of $[n * (n-1)]/2$ trained binary classifiers for n -number of classes. In the One-vs-All approach, one binary classifier for each class vs. the rest of the classes is trained, resulting in a total of n -number of binary classifiers for n -classes. Similarly in the One-vs-Rest approach, one binary classifier is trained for each class vs. the rest of the classes resulting in n -number of binary classifiers for n -classes. Generally, One-vs-All and One-vs-Rest classifiers are preferred, due to the smaller computational complexity. However, in some cases the individual class-to-class specific classifiers can perform better, and the best approach varies from application to application [15].

Multinomial logistic regression is a classifier that is used to perform multiclass classification tasks and is a generalization of the binary logistic regression. The goal of the model is to predict the probability of an input data point belonging to each class out of n possible classes. This is achieved by expanding the amount of output nodes. In binary logistic regression, there is a single output node that represents the probability of the input data point belonging to one class (positive class), and the complementary probability represents the probability of belonging to the other class (negative class). In multinomial logistic regression, there are n output nodes, each representing the probability of the input data point belonging to a specific class. To achieve a standardized sum of probabilities across the output nodes, a softmax activation function is used [16]. The softmax activation function is defined below:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \ , for \ = 1 \dots K \ \& \ z = (z_1 \dots \ z_K) \tag{5}$$

The softmax activation applied to the weighted sum of input features of each class converts the output into probabilities that sum up to 1. The probabilities of different classes can be further refined into a strict classifier by applying a decision threshold. Loss functions such as cross-entropy loss can be used to train a multinomial logistic regression model. The minimization of the loss function can be carried out by various optimization techniques, such as gradient descent. Cross-entropy loss is a generalized version of the binary cross-entropy loss [16].

In addition to linear regression and logistic regression, there are several other traditional machine learning algorithms commonly used in various fields. Decision trees are based on the principle of recursively splitting data into subsets to make decisions or predictions. Random forests, an ensemble method that combines multiple decision trees, are used for classification and regression tasks. Support vector machines are used for binary and multi-class classification and are based on finding optimal hyperplanes that best separate data points. k-Nearest Neighbors is a simple instance-based algorithm that classifies new data points based on the majority class of its k-nearest neighbors. Naive Bayes is a probabilistic classifier based on the Bayes theorem, commonly used for text classification tasks. Principal Component Analysis (PCA) is a dimensionality reduction technique used for feature extraction and data visualization. Traditional machine learning algorithms are suitable for a wide array of tasks and are used in various real-life applications. More complex models, neural networks, rely on interconnected layers of simple computing units and are often significantly more complex systems. These are further discussed in the following chapters.

2.2.4 Performance metrics

Several performance metrics can be used to evaluate model overall fit and generalization performance. In general, use of distinctive validation dataset for evaluating the model generalization performance is widely regarded as good practice compared to using performance metrics calculated on test set predictions. Validation set or any parts of it should not be part of the model training experience. Different performance metrics give differing insights to the model performance and metrics should be chosen with respect to the task and the end-use of the model [15].

For regression tasks, mean squared error and mean absolute error (MAE) are generally used. Mean absolute error is defined by the sum of absolute errors between individual predictions divided by the sample size. MAE is widely used in regression tasks due to its simplicity and interpretability but is not very sensitive to outliers and might not be suitable for a comprehensive evaluation of a model performance [16]. The MAE is defined formally below:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{m} \tag{6}$$

Root mean-squared error (RMSE) is widely used in regression tasks and penalizes large errors more relative to MAE. It is defined by the sum of the root of the sum of squares of errors between individual predictions divided by the sample size [18]. RMSE is defined formally below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{m}} \tag{7}$$

MAE and RMSE do not treat under, or over-predictions distinctively and therefore do not give an indication is of the positive or negative magnitude of a set of errors. Normalized mean biased error (NMBE) gives the average direction and magnitude of the bias of model's predictions. Unlike MAE and RMSE, NMBE can attain negative or positive values. The NMBE describes the bias of the predictions and is therefore not a comprehensive accuracy measure. For example, in cases where a model is prone to making significant errors in predictions, but the errors are not biased as positive or negative, the NMBE might obtain very small values even though significant inaccuracy is present [18]. NMBE is defined formally below:

$$NMBE = \frac{1}{\bar{y}} \times \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)}{n} \times 100\ (\%) \tag{8}$$

Model goodness of fit can be estimated with specific performance metrics that focus on the overall model fit over a dataset. R2-score measures the proportion of variance in the target variable that is represented in the model. R2-score can be used to intuitively assess how well a given model fits a dataset compared to a simple mean baseline. R2-ranges from 0 – 1 and higher values indicate better fit. R2-score is defined formally below:

$$r2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{9}$$

Distinctive performance metrics give differing insights into model performance. A combination of suitable metrics is often used in order to obtain a comprehensive view of the model generalization performance and the nature of the errors the model makes. Assessing performance should always be

done by reflecting the metrics and results to the end-use case of a given model. It is important to note that the quality of test and validation datasets determines the underlying ruggedness of model performance evaluation.

2.3 Deep learning

Simple traditional machine learning algorithms are suitable for a wide array of tasks in diverse domains. However, some tasks require a model with higher level of complexity and capacity to produce commendable results. Neural networks are a family of machine learning algorithms that currently represent state-of-the-art performance in various benchmark tests. Neural networks were first proposed in 1943 by Warren McCulloh. Albeit the underlying philosophy was the same, modern networks employ different and more sophisticated mechanisms. Inspiration behind neural networks is the biological brain - multitude of computationally simple units interconnected to form a system with uncanny abilities to capture representations of highly complex inputs. Neural networks suffered a decline in interest during 1970s and 1980s, which gave away after significant advancements in the neural network performance emerged [19]. Various neural network architectures have been proposed to address differentiating tasks, but different architectures share key components and fundamentals, as well as several design aspects.

2.3.1 Artificial Neural Networks

Neural networks are a group of machine learning algorithms, that consist of interconnected layers of perceptrons. Perceptrons are simple computing units that take an input vector, produce an output, and pass it through an activation function to achieve non-linearity. The output y of a perceptron is computed as follows:

$$y = g\left(\sum_{i=1}^{m} \boldsymbol{w_i x_i} + \boldsymbol{b}\right) \tag{10}$$

Where y is the transformed output, x is the input vector ($1\ x\ n$), w is the weight vector, b is the bias, g is an activation function and m is the number of inputs. Perceptrons are critical components of artificial neural networks, serving as their fundamental building blocks. Artificial neural networks consist of interconnected layers of perceptrons, with each layer typically comprising of multiple perceptrons. Combining multiple perceptrons forms a layer, which is responsible for processing input data and producing output signals. Artificial neural networks have one input layer, one or more hidden layers, and one output layer. The input layer receives the input data, while the hidden layers

process the data through interconnected perceptrons to learn representations. The complexity of the models allows neural networks to learn increasingly complex patterns from the data and thus perform difficult and even abstract tasks admirably. The output layer produces the final output of the model, which is either a linear regression or logistic regression model, depending on the task [15].

Fully connected neural networks are a type of neural network architecture that uses layers, where each output signal of the previous layer is connected to every node in the following layer. An illustration of a fully connected layer can be found below [17].

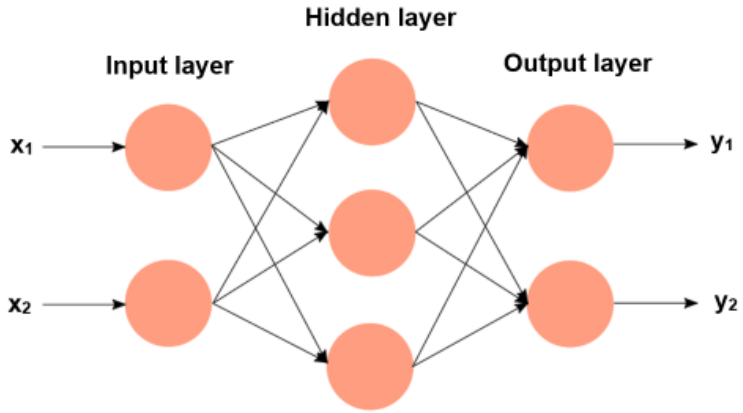

Figure 1. Simple fully connected neural network architecture.

Networks that have fully interconnected layers and no recurrent feedback connections are often referred to as feedforward networks. The input is manipulated by learnable parameters inside layers of perceptron units.

2.3.2 Specialized neural network architectures

In addition to fully connected feedforward networks, a wide array of varying network architectures has been proposed. Generally, different architectures perform better on specific category tasks. For example, convolutional networks are utilized especially in image processing tasks [16].

- **Convolutional neural networks** are a type of deep neural network commonly used for image recognition tasks, where they use convolutional layers to automatically learn hierarchical representations from input images.
- **Recurrent networks** are a type of neural network that can process sequential data by using recurrent connections, allowing them to capture temporal dependencies and handle variable-length inputs.

- **LSTM (Long Short-Term Memory)** units are a type of recurrent neuron that can effectively model long-term dependencies in sequential data, making them well-suited for tasks such as language modeling and speech recognition.
- **GRUs** are a type of recurrent neuron that are similar to LSTM units but with a simpler structure, making them computationally efficient and suitable for tasks that require capturing short-term dependencies in sequential data.
- **Encoder-Decoders** are a type of architecture used for sequence-to-sequence tasks, where an encoder captures input sequences into a fixed-size representation, and a decoder generates output sequences from that representation, making them suitable for tasks such as machine translation.
- **GANs** are a type of architecture that consist of a generator and a discriminator, which are trained in an adversarial manner to generate realistic data samples, making them powerful for tasks such as image synthesis and data generation.
- **Transformers** are a type of architecture that use self-attention mechanisms to process sequential data in parallel, making them highly scalable and effective for tasks such as machine translation, text generation, and language understanding.
- **Capsule Networks** are a type of architecture that use "capsules" to model the spatial relationships between different parts of an input image, allowing them to handle image recognition tasks with increased robustness to changes in viewpoint and pose.
- **Spatial Transformer Networks** are a type of architecture that use differentiable spatial transformation operations to enable neural networks to learn to spatially transform input data, making them useful for tasks such as image alignment, object detection, and image warping. [20]

In this work, we will focus mainly on feedforward networks with various regularization mechanisms as well as transformer architectures. In addition to architecture type, the network layout and other regularization methods as well as hyperparameter definitions affect the performance of a neural network model greatly. These design concepts, considerations and methods are further examined in the following chapters.

2.3.3 General model design concepts

The dataset used for training of a model is often a very little glimpse to the overall set of examples that the actual phenomena that the model aims to capture consists of. The aim for a deployed final

model is to perform well in a real-world application. This means that the model must generalize well to new datasets and input examples, that have been previously invisible to it. The model is trained on the training set, but since the paramount objective is to perform well with the unseen data, optimizing the model further with respect to the training data does not guarantee a better real-life performance. The error calculated during the training of a machine learning model is often referred to as the training error. In order to evaluate how well the model performs with real-world data, the full dataset $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{\boldsymbol{M}}$ is split into two or three parts. Only a part of the preselected data is used for training so that part of the preselected data can be used to obtain test or validation performance. Training set and test set are assumed to be identically distributed [15].

The two main tasks for machine learning model training procedure are to minimize the training error, while also minimizing the gap between the training and test error. Small training error does not strictly lead to a small test error. The capacity of a model can be thought to represent how complex functions can a model represent. If the capacity of a model is too high for the given task, the model can learn to fit the training data specifically and fail to represent the overall phenomena as a result. On the other hand, too small capacity leads to the model not being able to capture all required intricacies of the phenomena. These two concepts are often referred to as overfitting and underfitting. The concepts of overfitting g and underfitting are demonstrated below [15].

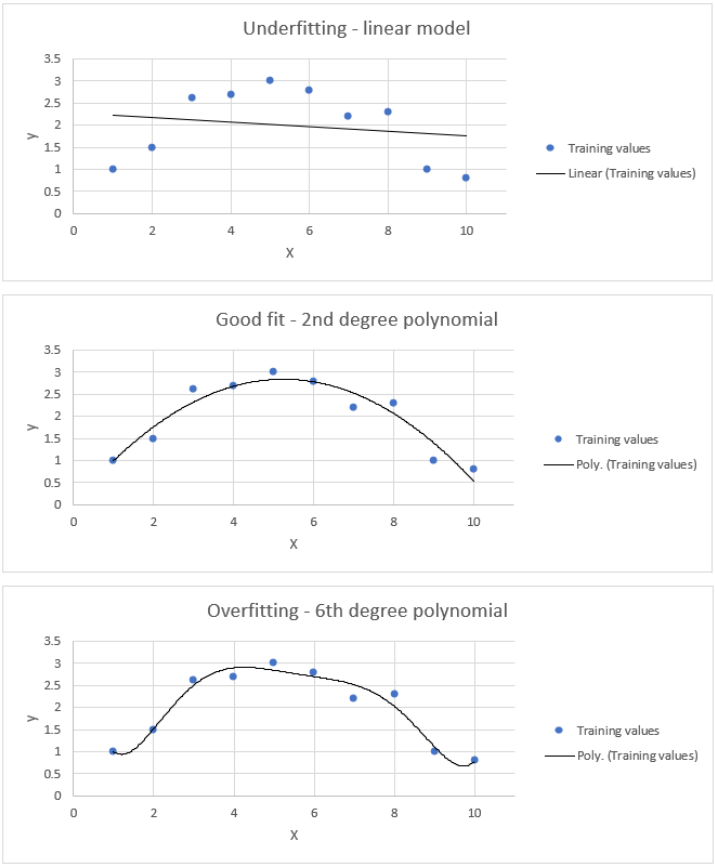

Figure 2. Underfitting & overfitting concepts in a simple regression task.

The complexity of the model depends on the task. Sufficient model capacity is necessary to enable the model to learn complex patterns and representations from the data. It is generally observed that both training and test errors tend to decrease with increasing model capacity, as the model becomes more capable of capturing the underlying patterns in the data. However, there is a point where the test error starts to increase, even though the training error continues to decrease. This phenomenon is commonly associated with overfitting, where the model may start to memorize the training data instead of generalizing well to unseen data. Traditionally, the capacity at which the local minimum of the test error is achieved is considered as the optimal capacity for the model, striking a balance between underfitting and overfitting. Careful consideration and analysis of the trade-off between model capacity and generalization performance are essential when selecting a machine learning model for a given task [15].

Overfitting phenomena is very common especially for complex neural networks and it can be observed by plotting the test error and training error by epoch. As previously discussed, test error is utilized as a measure of model generalization outside of the limited training set. Overfitting situation presents itself as an increase in test error, while training error continues to decrease. After this stage

the model can be thought to learn the distinct examples from the dataset and not the underlying patterns. An example of this phenomena is illustrated in the figure below.

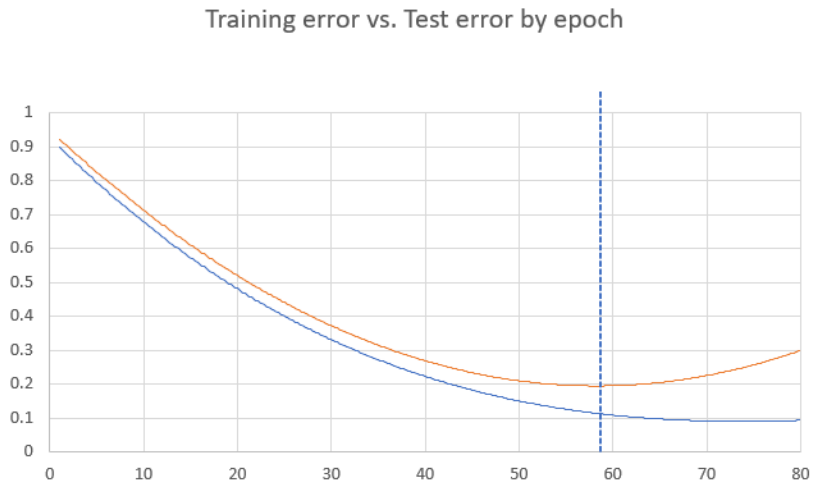

Figure 3. Training & test error development by epoch.

The optimal quantity of training epochs can be found, where the test error reaches its lowest point, even though the training error continues to decrease.

Occam's razor, a principle rooted in philosophy and widely applied in machine learning, suggests that simpler models should be preferred over more complex ones when all else is equal. This principle is often used as a guiding principle for model complexity selection, as simpler models are generally considered to be more interpretable, easier to understand, and less prone to overfitting. In the context of statistical learning theory, Occam's razor is closely related to the concept of model generalization, which refers to a model's ability to perform well on unseen data. Simpler models with fewer parameters are often believed to have higher generalization performance, as they are less likely to overfit the training data [16]. However, recent research has shown that with the advent of deep neural networks and other complex models, the relationship between model complexity and generalization performance may not always adhere to the principles of Occam's razor. In fact, more complex models with a higher number of parameters, such as deep neural networks, have been shown to achieve superior performance in various tasks, challenging the traditional notions of model simplicity and Occam's razor. One interesting phenomenon contradicting these classical machine learning model design philosophies, is the over-parameterization paradox. Over-parametrization refers to the use of models with a higher number of parameters compared to the size of the training data. Classical approach would deem this ineffective and one of the main reasons for high overfitting

risk. However, recent studies have revealed that over-parametrization may lead to improved generalization performance in certain cases [21]. This paradox arises due to the complex interplay between model capacity, overfitting, and generalization, and has important implications for model selection, regularization techniques, and the interpretability of deep neural networks. These empirical findings are still not comprehensively understood and are a topic of ongoing research in the field [22].

2.3.4 Training neural network models

Similar to traditional machine learning algorithms, the model aims to learn a set of parameter values that minimize the chosen loss function. Due to the complexity of multi-layer networks, analytical optimization is not possible and neural networks are trained iteratively. Most common optimization algorithm for training neural networks is backpropagation. Backpropagation is a widely used and highly effective neural network learning algorithm. The backpropagation algorithm has been successfully applied in real-world applications, training feedforward networks as well as other network architectures [16].

The backpropagation algorithm operates on a training set, which consists of input samples and their corresponding output labels. Each input sample is described by a set of attributes, which may be discrete or continuous. Discrete attributes require pre-processing, such as converting them into continuous values or representing them as k-dimensional vectors, where k is the number of attribute values. The backpropagation algorithm updates the connection weights and thresholds of the neurons in the network based on the error calculated at the output layer. The workflow of the backpropagation algorithm involves feeding each input sample to the input layer neurons, forwarding the signals layer by layer until the output layer produces results and calculating the error of the output layer. After these steps, we have obtained a result for the i-th epoch of the iterative training loop. Backpropagation then computes the gradient of the error with respect to the learnable parameters over the entire depth of the neural network. This is done by utilizing chain-rule principles, that allow the algorithm to examine the derivatives over multiple layers, while being reasonably computationally efficient. Backpropagation as a term defines the method how the gradient is obtained, but it is often used to describe also the following steps, where the parameters themselves are modified [15].

Backpropagation is often used with the stochastic gradient descent optimization method. Stochastic gradient descent (SGD) approximated the gradient for the whole training set by calculating the

gradient based on a randomly selected batch of training examples. This reduces the computational requirements while still achieving admirable results. After obtaining this approximation of the gradient, the algorithm updates the learnable parameters, for example by a previously defined step often referred to as the learning rate. This process is repeated for each training sample until a termination condition is met. Termination condition can be a defined set of training epochs, predefined value for training error or a rule that examines the development of training error and test rule [15].

2.3.5 Regularization

Backpropagation with SGD performs is often able to converge to a sufficient minimum and achieve small training losses. However, risk of overfitting is often high with the complex network models. Regularization methods are used to prevent overfitting and guide the model towards a set of parameter weights that improve overall generalization of the model. Various regularization methods are available, and they can be used either exclusively or together to achieve better regularization.

Early stopping is one of the simplest forms of regularization. In early stopping, the training of the model is stopped before the maximum defined number of epochs. The stopping point is determined by monitoring the performance on separate validation set with respect to the training error and validation error. This prevents the mode from overfitting, as shown in Figure 3 by stopping the training process, when the validation set error starts deteriorating [16].

L1 and L2 regularization methods add a penalty term to the loss function during training phase. The L1 regularization encourages sparse weight values and promotes feature selection. The penalty term inflicts a penalty proportional to the absolute value of the weights. L2 regularization, often referred to as weight decay, adds a penalty proportional to the square of the weights, which encourages small weight values. Both L1 and L2 regularization help prevent overfitting and restrict the complexity of the model thus promoting generalization [15], [16].

Dropout is a regularization method, where a specified share of the neural network perceptron neurons are set to zero on each training iteration. This in effect means that on that training epoch, the neurons that have been dropped out do not experience any parameter alterations i. e. learning. Dropout regularization forces the model to learn more robust representations on the underlying patters, since the model cannot rely on specific neurons doing significant transformations on the inputs. This improves generalization performance in total [16].

Batch normalization is a regularization technique that normalizes the inputs to each layer of the neural network. This is done by normalizing the outputs of the previous layer within each training batch. This helps stabilize and accelerate the training process, as well as reduce the sensitivity of the network to the scale and distribution of the inputs [16].

Data augmentation techniques are used to increase diversity in the training dataset. Techniques such as rotation, scaling, and flipping, can be used to generate augmented training data from the original dataset. This helps increase the diversity and amount of training data, which can improve the generalization performance of the neural network and reduce overfitting. Similarly, injecting artificial noise to the training data can help the model learn more robust patterns and also improve model generalization performance especially in real life applications, where noise can be present [16].

In addition to distinctive regularization techniques, the model architecture and hyperparameter values affect the generalization performance of the model greatly. Architecture of neural networks includes fundamental model configuration aspects, such as number and type of layer, number of neurons in each layer and the activation functions in the neurons. Changing these fundamental design considerations can have a significant impact on the model generalization as well as the computational requirements of the training and use of the model [16].

Hyperparameter tuning refers to the optimization of the non-learnable model parameters. These include learning rate, number of training epochs, regularization related parameters (regularization strengths, dropout probability, early stopping parameters) and optimizer choice. In addition, some architecture choices are often also tuned like any other hyperparameters. For example, tuning the activation function from a set of potential candidates is common practice and allows the final model to obtain better results. Hyperparameter tuning is considered widely an important step in all machine learning implementations, since it is impossible to state the actual potential performance of the model without careful hyperparameter considerations. A model that is capable of good results can easily perform poorly with unoptimal hyperparameter values. Hyperparameters, model architecture and regularization methods are in the end application specific, even though some methods have been proven to often perform better in specific tasks [16].

2.4 Energy consumption models – review

Building energy consumption prediction and forecasting has been an active field of research, with various model types being applied to various building prediction tasks. The performance of different models with respect to the specific applications is examined in this chapter. With varying tasks and available data, the optimal model choices found in different studies vary greatly. This speaks more of the applicability of distinctive models and distinctive applications than the overall performance of different models across the entire energy prediction domain.

2.4.1 Energy forecasting & prediction models

Energy estimation models have become an increasingly important area of research, particularly with the advancements in artificial intelligence and machine learning. Such models have found widespread application in energy systems, buildings, and HVAC systems. ASHRAE (American Society of Heating, Refrigeration and Air-conditioning Engineers) has classified energy estimation models into two main categories: physics-based/forward models and data-driven/inverse models [4]. Alternatively, energy prediction models have been classified into white-, grey- and black-box models. Both classifications categorize models with same principles [14].

White-box models or physics-based models make the energy load predictions by utilizing physics-based formulas. Often utilized in high-accuracy prediction tasks, such as simulation software, the models require a significant set of inputs and data from the modelled building, in order to obtain accurate results. Physics-based models are often utilized in building design stages, where the trade-off between excellent accuracy and transparency of results and the time-consuming process of defining input parameters with extensive detail is beneficial. However, in situations where there is no access to the detailed building parameters or plans and manual evaluation of such details is not feasible, data-driven models can provide a sufficient and more efficient solution [4].

Data-driven models have emerged as a powerful alternate to the physics-based models in the last decades. These models can be broadly categorized into two main classes: mathematically based and statistically based models. Mathematically based models typically apply a pre-set mathematical function to the input data to produce the output, whereas statistically based models rely on the data itself to determine the mathematical relationships between the input and output variables. As a further classification, black-box and grey-box models are two subtypes of data-driven models that describe the model architecture in further detail. Grey-box models incorporate a partial physics-based model along with a data-driven model, while black-box models are purely data-driven. The

popularity of data-driven models is due to their ease of development, high accuracy, and suitability for a wide array of applications. This has been further fueled by the growing interest in machine learning and deep learning models, which have been successfully applied to energy forecasting tasks. Machine learning algorithms, including the neural network models, are classified as black-box models, since they are purely data-driven [14].

Energy consumption forecasting & prediction tasks are often regression tasks. Energy consumption is dependent on multiple features and relationships between feature values and final consumption are complex. Due to the complexity of the task of energy consumption prediction & forecasting, neural networks have performed well in this domain. The complexity of representations that neural networks can learn give the models ability to learn the inherently complex interplay between features and their respective effect on the final consumption.

2.4.2 Energy disaggregation & NILM -tasks

Energy disaggregation tasks involve deconstructing an aggregated data, such as overall consumption, to the original components it is originally derived from. This task can be executed in real-life scenarios intrusively, by metering the end-use appliances and aggregating the respective energy flows to the distinctive categories. This approach is however expensive to implement due to the acquirement and installation costs for the increased number of meters and sensors. Meters are also a pre-emptive method, since the benefits of deconstructed data are evident only in future projects. Energy disaggregation in buildings with data-based models is an active field of study. The tasks vary from one building to general models and are implemented to disaggregate consumption data in various granularities and resolutions. Both electricity and heat consumption can be disaggregated.

Zaeri et al. (2022) employed a multiple linear regression model for overall energy consumption disaggregation. Model was employed for one building and three output classes – lighting & plug loads, AHU heating and perimeter heating. It was found that the machine learning mode successfully learned the representations and was able to approximate the categories with the worst performing category achieving an R^2 -score of 0.73 with 60 min granularity task [23]. Deuk-Woo et al. (2022) implemented a weather sensitive energy disaggregation model for 11 building datasets. Change point regression was compared to a non-parametrical model. It was found that the parametrical regression model was able to reach similar results to the white-box model [24].

The complexity of an energy disaggregation problem is increased when model is trained on a set of buildings and then generalized to perform the task on similar but unseen buildings. Batra et al. (2015) studied non-invasive energy disaggregation with big data and examined the suitability of applying

energy disaggregation models based on buildings to similar but previously unseen buildings. The study focused on disaggregating overall consumption to electrical appliance use and consumption. Batra et al. (2015) found that models generalize well between similar buildings and that the assumptions related to similar consumption patterns across similar households and buildings is valid [25].

Paresh et al. (2019) studied deep learning approaches to energy disaggregation tasks. In appliance specific energy disaggregation tasks computational costs of implementing complex deep learning models are high since each appliance needs to be learned individually and the model needs to be re-trained if more appliances are added to the portfolio. Deep learning provides good performance especially in complex problems, but the computational requirements prove too significant for variety of tasks [26]. Rahman et al. (2018) studied HVAC load disaggregation using neural network based extreme learning machines. Extreme learning machines are a class of deep feedforward neural networks that utilize random analytically optimized weights and are often used in applications, where computational requirements of deep neural network training is not feasible. The HVAC disaggregation task was a pure classification task in contrast to the previously mentioned tasks [27].

Recurrent neural networks that utilize feedback connections have the capability to leverage temporal patterns in addition to relationships between variables in the current data instance. Tongta et al. (2020) studied LSTM neural network architectures in individual electrical appliance energy disaggregation tasks. LSTM network was found to perform better compared to feedforward deep network and achieved a 48.8 % smaller mean absolute error with unseen validation data [28]. Kaselimi et al. (2020) also found that LSTM models perform better compared to various traditional machine learning algorithms as well as deep feedforward networks. In addition to leveraging the long-term dependencies in the data by using the LSTM architecture, the energy disaggregation task was defined as a regression problem instead of a classification task. Energy disaggregation as a regression task allows the model to output not only the distinctive on/off events but also reproduce the shape of consumption loads [29]. He et al. (2016) used an LSTM architecture as well as convolutional layers for more efficient feature extraction. LSTM without the convolutional layer-based feature extraction performed well and the performance was enhanced slightly with the use of convolutional layers. To achieve good performance, the LSTM models require a large amount of data, which is typical for all complex neural network models. LSTM training is also computationally heavy even compared to other neural network models with large complexity [30].

Different deep learning network architectures have been proposed to further enhance deep learning-based energy disaggregation models. Piccialli et al. (2021) propose a self-attention based deep neural network model for deconstructing appliance consumption shares from an aggregated overall consumption. Study found that self-attention mechanism improves the performance compared to various sequential networks [31]. Many sequential tasks have been improved by utilizing self-attention-based transformer networks in other domains, such as natural language processing [32].

2.4.3 Energy forecasting & prediction regression tasks

Neural networks have been utilized extensively in energy forecasting & prediction tasks. One of the key differences between studies and the respective neural network model tasks is the granularity of the consumption input time-series. Models can be trained to estimate consumption for example 1 minute intervals to one day intervals or even more. Model architectures are variations of neural network architectures discussed in previous sections. Table 3 presents a summary of studies that employ a neural network model to a time-series related regression task.

Table 3. Machine learning model research in energy consumption prediction & forecasting tasks.

Author	Data source	Granularity (min)	Network type	n. hidden layers	Activation function
Park et al.	Calendar data, weather data	60	FFNN, RNN, NARX	1	tanh
Mordjaoui et al	Calendar data, historical data	30	Dynamic NN	1	
Qiu et al.	Calendar data, weather data, economic factors, random effects	30	RVFL network	1	logistic sigmoid
Reddy	Historical data, weather data	60	BPNN	1	Sigmoid
Hu et al.	Historical data, weather data	60	GRNN		
Ertugrui	Calendar data, weather data, historical data	15	ELMNN	1	Sigmoid
Zeng et al.	Calendar data, weather data, historical data	60	ELMNN	1	tanh
Li et al.	Calendar data, weather data, historical data	60	ELMNN ensemble	1	sigmoid
Mocanu et al.	Historical data, weather data	1	RBM	1	Sigmoid
Raza et al.	Calendar data, historical data	60	LM-based NN	1	
Elgarhy et al.	Calendar data, weather data, historical data	60	LM-based NN	1	Logistic sigmoid
Singh et al.	Calendar data, weather data, historical data	60	BPNN	1	Sigmoid
Dedinec et al.	Calendar data, weather data, historical data	60	DPN	2	tanh
Qiu et al.	Calendar data, weather data, random effects, economic factors	30	DPN	2	
Ryu et al.	Calendar data, weather data, historical data	60	DPN, DNN	4	sigmoid, ReLU
Fan et al.	Calendar data, weather data, chilled water temperature	60	DAE, DNN	DAE 3, DNN 2	DAE tanh, DNN ReLU
Din & Marnerids	Calendar data, weather data, historical data	60	FFNN, RNN	RNN: 1	ReLU
Kuo & Huang	Historical load	60	CNN	3	ReLU
Amarasinghe et al.	Historical load		CNN	5 + 2	ReLU
Saquid et al.	Historical load	5	Transformer with stationary wavelet transform		
Yuan et al.	Historical load, weather data, holiday (binary)	1 day	LSTM, self-attention		Leaky ReLU

Neural networks are found to perform well in various regression tasks in the energy prediction & forecasting domain. It is important to note that forecasting & prediction tasks differentiate from the task that this paper addresses. Although the examined papers employ all represent regression tasks in the building energy domain, disaggregating the heat consumption differs significantly in the available features. Many of the presented studies use only the realized consumption time-series as the basis of the predictions. The models that are employed in this paper have the overall consumption as one of the explanatory features not only in training, but in deployment setting.

Additionally, the heat consumption disaggregating task is multinomial with four outputs, in contrast to the presented prediction & forecast studies that only predict one output. The distribution is also not strictly limited by task definition, since consumption quantities can theoretically vary on an arbitrary scale. In the case of heat consumption disaggregation task, the distinct outputs vary on a distribution between 0… 1 and their sum cannot exceed 1 by task definition [33]–[45].

Examples for energy prediction & forecasting found in literature review found that proposed networks had relatively small number of hidden layers with 12 of the identified time-series forecasting models employing only one hidden layer. 8 studies utilized a deeper model, with hidden layers up to 7. The granularity of the time-series data varied from 1 min intervals to 1 day intervals, with majority of the models employed for 60 min granularity. Sigmoid activation function was found to be the most widely used. ReLU activation was also utilized especially in models with multiple hidden layers. Tanh activation was used in significantly fewer examples. Best performing architectures generally employed a recurrent unit that enables the model to extract trend patterns in previously predicted and realized consumptions [33]–[45].

3 SWECO building heat disaggregation problem and data

As is the case with many engineering consultancies, the available data on customers' existing buildings is rarely fully comprehensive. The heat segment of the energy consumption is optimally available in the four categories: domestic water heating, domestic water heating circulation losses, space heating & ventilation system intake air heating. The categories are further discussed in section 2.1. This level of consumption data accuracy allows for optimal energy retrofit investment decision making.

When considering new development projects, this level of accuracy is obtained as product of simulations. The construction of the simulation models is an integral part of the workflow of designing a given building and therefore acquiring the disaggregated heat consumption data is time efficient. SWECO predominantly uses IDA ICE is used for dynamic multizone energy simulations.

For energy retrofit projects for existing buildings, the workflow does not necessarily require construction of an IDA ICE simulation model. Especially in the preliminary cases such as preliminary analysis or exploratory investment calculations, the construction of a simulation model is relatively time intensive. In these cases, generally only the overall heat and electricity consumption time-series with one hour granularity are generally provided. To disaggregate the consumption, the simulation model needs to be constructed for existing buildings and then calibrated to match the measured consumption data. Motivation to develop a model for disaggregating the heating data is due to the time intensive process of constructing and calibrating a simulation model in a situation, where it cannot be fully leveraged.

Training a statistical machine learning model to alleviate the problem requires data. As a key factor in the presented problem, no disaggregated real consumption data is available for existing buildings. This presents a problem in obtaining a labelled dataset that is needed for a supervised learning process. Despite the lack of real disaggregated data, the dataset can be constructed by utilizing the IDA ICE simulation model results from previous projects. Since simulation models have had to be constructed for the previous new developments but also existing buildings retrofit projects, a simulated time series can be acquired for each project where the heat consumption is segmented in the four necessary categories. In addition, the structural parameters as well as user interaction parameters are documented for each project. This allows to construct a labelled dataset from the simulated "virtual twins" of past development projects to be used as a training, testing and validation dataset for a supervised learning task.

3.1 Synthetic dataset properties & pre-processing

The dataset is constructed from 16 buildings that are apartment buildings, offices, and schools. Each building has at least one year of simulated consumption at the hourly scale, with several buildings simulated on multiple different years. In total, 24 distinctive one year time-series were included in the dataset with hourly resolution. All simulations have been carried out with respect to varying years, with the varying temperatures in the time-series as well as the varying holidays and temporal effects representing a wider range of environmental conditions throughout the data. The dataset was constructed by extracting structural constants and parameters from the simulation model and combining them with the simulation results, thus obtaining a fully synthetic dataset based on the existing SWECO project portfolio.

3.1.1 Feature selection and engineering

From each building, 22 features were gathered. Two of them, the overall heating consumption as well as the temperature are obtained from the simulation results. Seven of the features describe the use profile of the building. The dataset comprises of 210240 data entries in total, where each entry represents one hour in the consumption time-series in distinctive buildings. Data points include the consumption time-series for the four categories discussed in section 2.1 temperature time-series as well as the structural parameters of the buildings. The dataset was audited as a part of the data collection process to ensure the quality of the data. Projects that included problematic datapoints or incomplete information were discarded to ensure the quality of the dataset to ensure a solid foundation for robust model training and evaluation. There are no outliers, missing values or otherwise problematic datapoints in the entire dataset. The features gathered from simulation models are presented in Table 4.

Table 4. Dataset model features.

Variable	Type	Source	Description
Heating overall	Float	Simulation results	The overall heat consumption time-series. Aggregated from the simulation results for the four distinctive heat consumption categories.
Temperature	Float	Simulation input	Outside temperature time-series
Area	Float	Structural parameter	Area of the building
Volume	Float	Structural parameter	Overall volume of the building
Max users	Integer	Use profile parameter	Maximum number of occupants/users in the building at a given time
Outside walls overall area	Float	Structural parameter	The overall area of the outside walls in the building
Window area	Float	Structural parameter	The overall area of outside windows in the building
Envelope area	Float	Structural parameter	The overall envelope area of the building
U-value	Float	Structural parameter	Approximated thermal conductivity of the building structure
Lighting energy consumption	Float	Structural parameter	Yearly lighting energy consumption
Appliance energy consumption	Float	Structural parameter	Yearly appliance energy consumption
Type of a building	Categorical	Structural parameter	The type of the building. Categorial variable (Apartment building, school, office)
Active use time starts	Float	Use profile parameter	The start of active use. Values between 0-23
Active use ends	Float	Use profile parameter	The end of active use. Values between 0-23.
Active use on Saturdays	Binary	Use profile parameter	Is use profile on Saturdays similar to weekdays
Active use on Sundays	Binary	Use profile parameter	Is use profile on Sundays similar to weekdays
AC -use during inactive hours	Float	Use profile parameter	AC -use during the inactive use profile.
Supply air	Float	Structural parameter	Maximum overall AC supply air volume.
AC -system heat recovery efficiency	Float	Structural parameter	Average efficiency of AC -system heat recovery
AC -system type	Categorical	Structural parameter	Type of the AC -system. Categorical variable (heat recovery, no heat recovery, gravitational)
Q50 -value	Float	Structural parameter	Air leakage value. Standardized metric.
DHW circulation value	Float	Structural parameter	Water heating system circulation loss constant.
Hot water use	Float	Use profile parameter	Hot water use constant.

The different variables have significantly different value ranges. Dataset consists of large scalar variables with high standard deviation, such as overall consumption. Also, continuous variables vary within ranges bounded by known values by definition, such as heat recovery efficiency. There are also binary and categorical variables. The basic descriptive statistics for the continuous features in the dataset can be found in Table 5.

Overall heating consumption has the largest mean of 92632 with a standard deviation of 112019. The three largest scalars have also the three largest standard deviations. However, standard deviation scaled proportional to the size of the mean is largest for temperature with a mean scaled standard deviation of 163 %. The overall heating as well as the Q-50 value have a large mean scaled standard deviation. Min-max scaled standard deviation, the standard deviation scaled to the delta between minimum and maximum values, is highest for structural parameters such as heat recovery efficiency, supply air, appliance energy consumption and U-value. The min-max scaled standard deviation is lowest for heating overall and temperature.

Descriptive statistics do not give full insight into the data. However, the scale of differences in the value ranges and means of the numerous variables suggest than the dataset should be normalized or scaled before data-driven model training.

Table 5. Descriptive statistics of continuous variables.

Variable	Mean	Standard deviation	Mean scaled standard deviation	Min-max scaled standard deviation
Heating overall	92632	112019	121 %	9.2 %
Temperature	5.62	9.14	163 %	16.7 %
Area	7351	5212	71 %	26 %
Volume	22242	16901	76 %	31 %
Max users	840	747	89 %	22.9 %
Outside walls overall area	3355	2519	75 %	23.3 %
Window area	968	717	74 %	28.5 %
Envelope area	8789	5626	64 %	25.4 %
U-value	0.41	0.18	45 %	31.8 %
Lighting energy consumption	11.58	4.61	40 %	24.0 %
Appliance energy consumption	7.86	5.62	71 %	31.8 %
Supply air	1.64	1.45	88 %	36.8 %
Heat recovery efficiency	0.58	0.32	55 %	39.4 %
Q-50 value	3.20	3.52	110 %	21.6 %
DHW circulation value	1.64	0.67	41 %	25.5 %
Hot water use	35.73	33.92	95 %	20.9 %

In addition to the features acquired from the simulation model and simulation results, several other features were aggregated. The aggregated features are designed to allow for models with no feedback connections to learn representations of temporal patterns. They are constructed by utilizing the date and time information associated with the simulation year. The features are described in the Table 6.

Table 6. Aggregated features.

Variable	Type	Description
id	Integer	Specific *id* for every building in the training set. Used for miscellaneous use, not a trainable parameter.
Hour	Integer	Hour of the day (0-23). Aggregated from datetime variable of the simulation year date.
Number of the day	Integer	Number of the day of the year (1-365). Aggregated from datetime variable of the simulation year date.
Number of the day in the week	Integer	Number of the day of the week (1-7). Aggregated from datetime variable of the simulation year date.
Holiday	Binary	Is the day a national holiday in Finland. Aggregated from datetime variable of the simulation year date.

The labels used for the supervised learning task are acquired from the simulation results. Due to the original description of the problem, the model was trained to approximate the shares of heat consumption between the four distinct categories. The shares of consumption, aggregated from the simulation results for the four categories, were used as the labels in the training phase. Labels used for the supervised learning task can be found in Table 7.

Table 7. Labels used for the supervised learning.

Labels	Description
Domestic water heating	Energy consumption used for domestic water heating service.
Domestic water heating circulation losses	The energy losses in the domestic water heating circulation system.
Space heating	Energy consumed for space heating.
Ventilation system intake air heating	Energy consumed for the pre-heating of the intake air in the ventilation and air conditioning systems of the building.
Aggregated labels	
Share of domestic water heating	The share calculated with respect to the overall energy consumption at given instance.
Share of domestic water heating circulation losses	The share calculated with respect to the overall energy consumption at given instance.
Share of space heating	The share calculated with respect to the overall energy consumption at given instance.
Share of ventilation system intake air heating	The share calculated with respect to the overall energy consumption at given instance.

The different four consumption categories follow clear but different trends. The space heating consumption as well as intake air heating consumption follow outdoor temperature tendencies indicating a strong inverse correlation. Seasonal cycles are apparent for both categories. The volatility of ventilation intake air heating is larger than space heating in the examined example. The findings are intuitive, since the ventilation unit runtime is in the IDA ICE simulation model itself dependent on the use profile parameters such as active use. Additionally, the space heating and ventilation intake heating demand is strongly dependant on the outside temperature, since the service provided by these systems is to maintain suitable indoor climate [6]. The space heating and ventilation intake air heating hourly consumption time-series can be found in Figure 4.**Error! Reference source not found.**

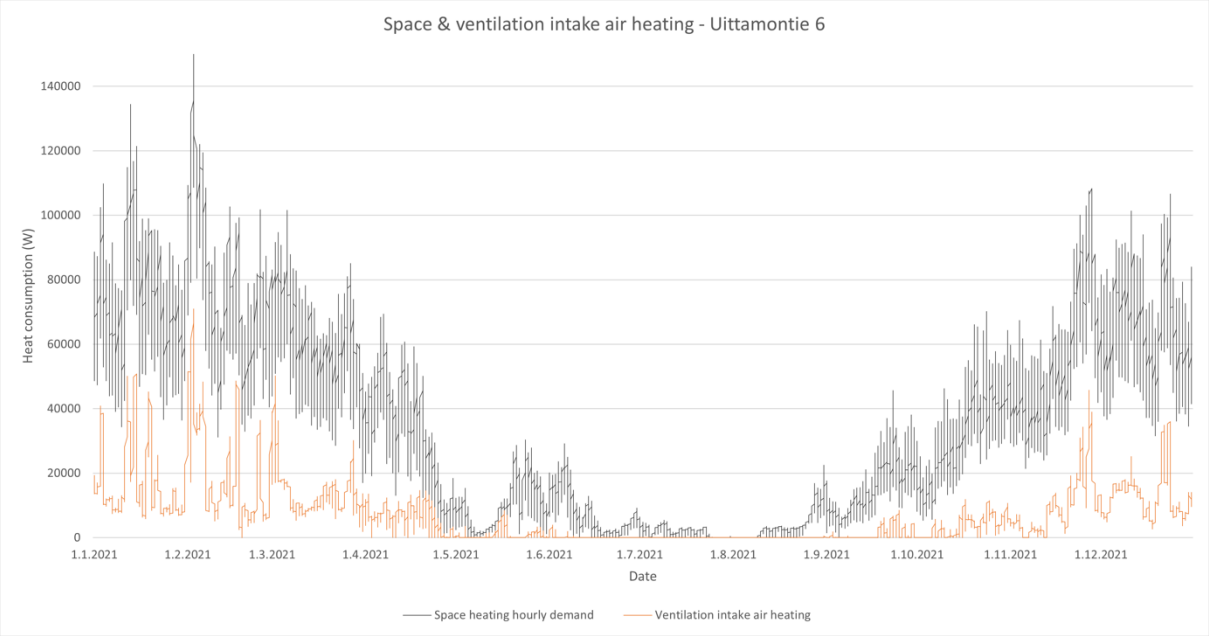

Figure 4. Space and ventilation intake air heating time-series - Uittamontie 6.

The domestic hot water usage follows a shorter 24-hour cycle with the domestic hot water circulation losses remaining constant for this cycle. Domestic hot water heating has a larger scale and volatility compared to the circulation losses. Domestic hot water usage or circulation losses do not exhibit similar seasonal patterns to the overall heating or ventilation intake air heating. Findings are intuitive, since hot water usage is determined largely on use profile parameters as well as building type and the use patterns related to the use of the building type in the simulation model. The circulation losses are determined in the simulation model by the related constants and other

structural parameters, which are not dependent on temperature or any other variable with clear seasonal pattern. [6] The domestic hot water heating and circulation losses can be found in Figure 5.

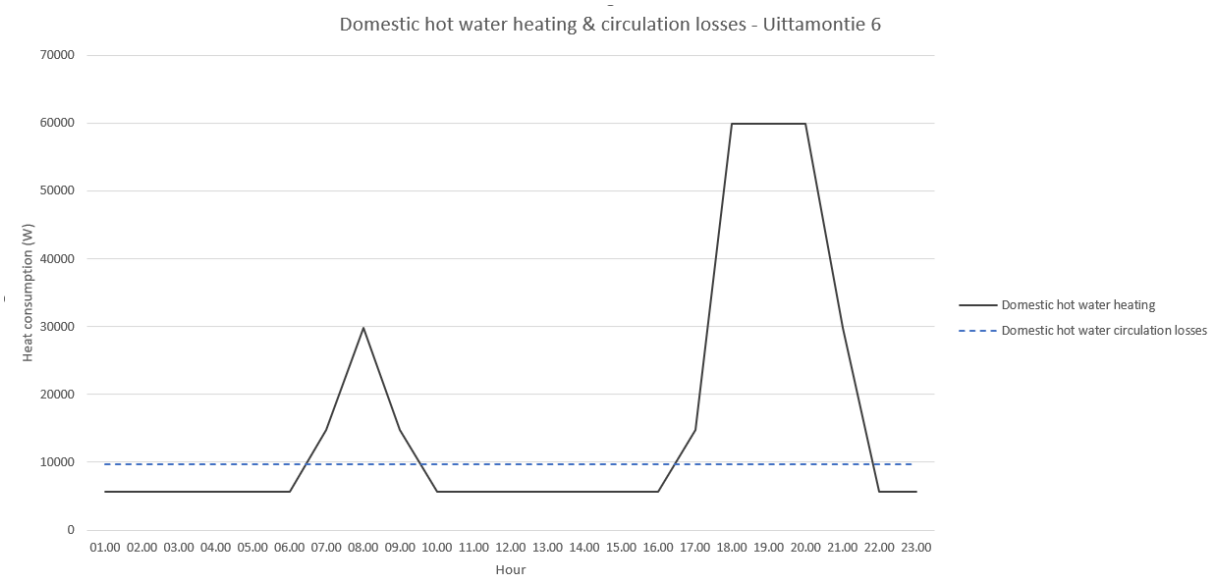

Figure 5. Domestic hot water heating and circulation losses - Uittamontie 6.

The presence of temporal patterns in the consumption data is not unexpected, since energy consumption is often autocorrelated. Autocorrelation describes the dependencies between a variable at a current time and its lagged values. Autocorrelation is a measure of the correlation between a current state value and lagged values. High positive autocorrelation at a given lag indicates the presence of temporal patterns and trends, while high negative autocorrelation indicates a presence of alternating patterns with inverse correlation relationships.

3.1.2 Temporal connections

With respect to data-driven model development, identifying the presence of temporal connections is important. Models with high complexity but no feedback connections are at risk of underperforming with highly autocorrelated time-series [16], [17]. To further examine autocorrelation within the dataset buildings, three example buildings were selected. Ilmalanrinne D-E, Tiedepuisto school & Vilppulantie 26 all represent different building type categories – office, school & apartment buildings, respectively. Representatives from different building classes were selected so that not only the autocorrelation could be examined, but also differences in between the categories could be examined. The autocorrelation function plots (AFC) of three example buildings can be found in Figure 6 and Figure 7 for longer and shorter time scales, respectively.

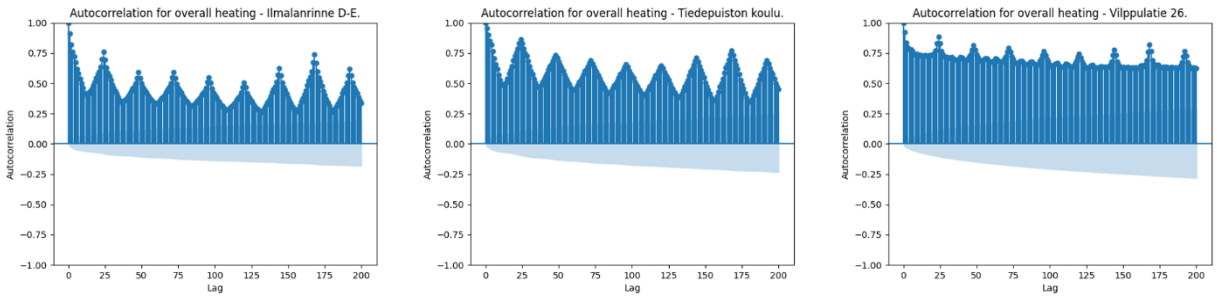

Figure 6. Autocorrelation for example buildings at long time scale.

The overall consumption is overall positively autocorrelated for all three building categories. High autocorrelation is observed for all three example buildings at lag 24 and all its future composites with a declining trend. This declining trend for correlation peaks for the composites of 24 is disrupted at 168 lags (one week), where correlation peak attains a notably higher value. Findings suggest a strong dependency between current consumption and the consumption same time last day. The dependency decreases for days further away. However, a stronger dependency can be found again between current consumption and consumption at the same time and the same day of the previous week, demonstrating the existence of weekly patterns. Between the discussed daily and weekly peaks, the apartment building exhibits strong dependency also with respect to consumption at hours not included in the daily or weekly cycles. Office and school building examples show a significantly lower, albeit still significant, dependency for these hours.

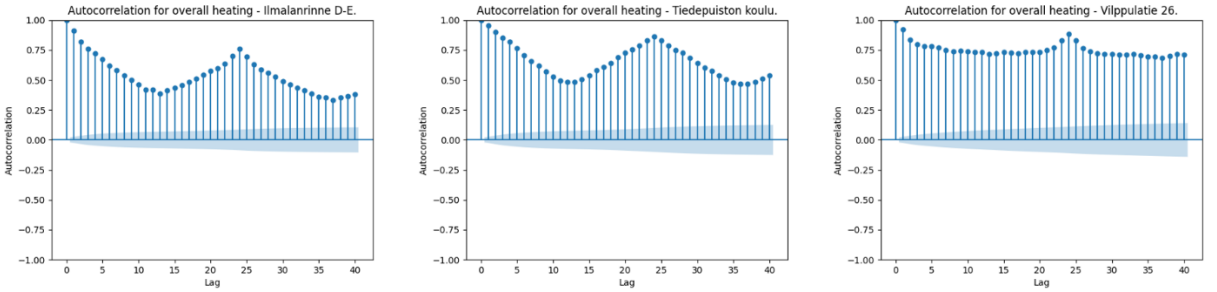

Figure 7. Autocorrelation for example buildings at short time scale.

The autocorrelation function plotted for a shorter time scale exhibits the same phenomena of clear daily cycles. Autocorrelation steadily declines for both office and school building examples and reaches the lowest correlation at 12 lags demonstrating the existence of two occupation patterns within one day, usually between day and night. The autocorrelation then rises steadily, exhibiting largest correlation at 24 lags. The apartment building example shows a steady decline in during the

first lags but remains at a high level of correlation. The autocorrelation rises again exhibiting highest correlation at 24 lags. Findings show that apartment building example exhibit strongest autocorrelation for the daily cycle. The 24 lags peak is clearly present also in the short time scale autocorrelation function plots for all building type examples. Dependencies for office and school building examples is not as high between the 24 lags peaks compared to the apartment building example, but still present strong dependencies through the cycle.

The example buildings exhibit high autocorrelations which implies a presence of time dependent patterns. In addition to the exhibited short and long-time scale autocorrelation analysis, various plots were additionally examined, and the results were in line with the presented findings. From the perspective of data-driven model development, potentially poor performance of models with no temporal feedback connections might be attributed to the strong temporal patterns found in this autocorrelation analysis. It is important to note however, that these findings were made on example buildings and thus only approximate the qualities of the entire dataset. Also, the large amount of features covered by the dataset could provide sufficient representations for a model with no temporal feedback connections to capture accurate representations of the phenomena. Even though the presence of temporal patterns is evident from the autocorrelation analysis, the suitability of different machine learning algorithms needs to be validated by training the models and assessing their performance.

The overall dataset contains 210240 data instances with overall 28 features as explanatory features and 4 labels. The data pre-processing was implemented with a Python program utilizing the Pandas and the Sci-kit learn libraries. Data analysis was implemented with Pandas, Sci-kit learn and Statsmodels libraries with Matplotlib library used for visualizations. Additionally, Microsoft Excel was used in the data collection, quality control and visualizations.

3.2 Proposed model and results

Proposed models for further examination for the SWECO building heat disaggregation problem are based on the findings in the literary review. The design considerations need to be adapted to the specific requirements and potential of the available data. The problem was formulated as a multinomial supervised regression task. In this chapter, the models with potentially best performance will be presented. It is important to note that the model's performance needs to evaluated in the specific task and preliminary findings are to be treated as only guidelines that require further validation.

3.2.1 Literary review findings with respect to neural network design considerations

Literature review shows that machine learning models achieve good performance in energy disaggregation tasks as well as energy forecasting and prediction tasks. Various neural network models achieved generally good performance compared to traditional machine learning models on both task categories.

In energy disaggregation tasks, both classification tasks and regression tasks are represented. Classification tasks are more common in the cases of non-intrusive load monitoring of appliances where foremost interest is classifying the on/off -instances of specific household electrical appliances. Regression tasks give more insight into the load profiles of specific appliances. The SWECO building heat disaggregation case was defined as a regression energy disaggregation task. The available training data and the associated labels are not in binary but continuous form. Also, the ability for a model to not only recognize the instances when any specific heat consumption category is active but approximate the consumption share of overall consumption at any given time [6].

Neural network models with differing architectures achieved good performance in energy disaggregation tasks. The complexity of especially energy disaggregation tasks requires complex models to achieve a good representation of the phenomena. In turn, the computational requirements for training neural networks might prove problematic in some cases. Appliance specific energy disaggregation tasks suffer from the computationally heavy complex neural networks due to every distinctive appliance requiring a dedicated class in the classification task. Similarly, if appliances are switched or new appliances are added, the model will need to be retrained to accommodate changes. In the case of SWECO building heat disaggregation, these factors are not a problem. The classes are predefined and established in the engineering consultancies mode of operations. Due to this fundamental difference in the problem definition, the computational costs do not present a similar problem that is often found restricting in the appliance specific non-intrusive load monitoring tasks.

The performance is further enhanced by recurrent neural network architectures. The feedback connections improve the model's ability to capture temporal dependencies in the data. Of various recurrent neural network algorithms, LSTM networks were found to perform well in most cases compared to feedforward neural networks. Energy consumption data is inherently temporal. Leveraging the temporal patterns by utilizing feedback connections was found useful in several studies. However, LSTM layers are computationally intensive to train.

Neural networks were found to perform well in energy prediction and forecasting tasks. Generally, the number of hidden layers in the proposed neural nets was relatively small. Recurrent architectures improved performance similar to energy disaggregation tasks. Forecasting and prediction models studied were regression tasks and therefore differ from the SWECO building heat disaggregation task.

Summary of neural network hyperparameter design in energy disaggregation as well as energy prediction and forecasting models can be found in Table 8.

Table 8. Summary of hyperparameter design considerations for deep learning models.

Hyperparameter	Findings
Activation function	
ReLU	ReLU was used most commonly with deep networks that had several hidden layers.
Leaky ReLU	Leaky Relu, a variation of the ReLU function was also used. Leaky ReLU activated neurons gradient do not become zero during training.
Sigmoid	Sigmoid activation was most common, but is generally not as effective as ReLU [46]
Tanh	Tanh activation was not widely used and is generally not as effective as ReLU [46]
Network architecture	
Number of hidden layers	Some forecasting & prediction models proposed utilized shallow (1 – 2 hidden layers) network architectures. More complex models in energy disaggregation tasks utilized deeper networks.
Feedback connections in networks	Recurrent neural networks presented good performance compared to feedforward neural networks. Several LSTM networks achieved performance gains over other architectures.

Feedforward networks and LSTM-networks are proposed for further study due to their good performance. Even though shallow networks were present in the studies examined, deeper networks with several layers were able to perform well, especially in the energy disaggregation tasks. The feedforward networks as well as LSTM-networks should be tested with several hidden layers. It is important to note that the number of hidden layers as well as the number of neurons in each layer can and should be considered a tunable hyperparameter in further research due to the relatively small size of the dataset and thus manageable training times. If the test metrics don't imply otherwise, the number of hidden layers can be limited to 10 since no examples of deeper networks

were found. ReLU and leaky ReLU can be considered as activation functions. Including self-attention to the LSTM-network should be tested, as the inclusion of attention was seen to improve LSTM-network performance.

3.2.2 Traditional machine learning model implementation as baseline

Traditional machine learning algorithms were implemented as a first stage of statistical model development for the problem. Multivariate linear regression as well as 2-degree and 3-degree polynomial regression models were implemented. Linear regression was chosen as it is one of the simplest yet widely used machine learning algorithms for regression tasks. Polynomial, both 2-degree and 3-degree, were chosen since the relative performance with respect to each other as well as the linear regression give insight to the complexity of the problem. If no accuracy gains would be gained on a task with the more complex polynomial models, the risks of overfitting increases. The traditional machine learning models performance results serve as a baseline for further model implementations.

Data was standardized using the Scikit-learn library's StandardScaler -function. The function standardizes the data by removing the mean and scaling to unit variance. The dataset was split for a training and test set with a ratio of 0.8. 2-degree and 3-degree polynomial regression models were trained utilizing the same standardization method and train-test split. The results are presented in Table 9. The model outputs shares of overall consumption for the four specific categories. The units for outputs and the respective error metrics are not in energy quantity or power units.

Table 9. Test results for linear & polynomial regression models.

Algorithm	MAE (%)	RMSE	r2 -score
Linear regression	0.148	0.190	0.46
2-degree polynomial	0.107	0.143	0.69
3-degree polynomial	0.078	0.109	0.82

For the preliminary baseline tradition machine learning algorithm implementation, only test metrics were calculated. The 3-degree polynomial regression performed best in all the metrics used achieving an MAE of 0.078, RMSE of 0.109 & r2 -score of 0.82. Linear regression performed poorly, achieving nearly 0.15 MAE, RMSE of 0.19 and r2 -score of 0.46. The 2-degree polynomial regression models' performance was an improvement on the linear model, but significantly worse compared to the 3-degree polynomial model. In addition to the metrics used in Table 9 that describe model

performance on the overall test set, the maximum errors in the test set were recorded and can be found in Table 10.

Table 10. Maximum errors on test set for linear & polynomial regression models.

Algorithm	**MAE** Space heating (%)	**MAE** Ventilation intake heating (%)	**MAE** Domestic hot water heating (%)	**MAE** Domestic hot water circulation losses (%)
Linear regression	**16.98**	**8.60**	**17.14**	**16.34**
2-degree polynomial	**12.78**	**5.84**	**12.52**	**11.58**
3-degree polynomial	**9.96**	**4.40**	**9.43**	**7.30**

The performance on category specific MAE for the assessed models follows similar trends to the overall metrics. The 3-degree polynomial model achieved lowest category specific MAE over the test set with highest error made in the Space heating category. The 2-degree polynomial achieved lower category specific MAE over the test set in all categories compared to the linear regression model. category specific MAE for each category is used as a performance metric to give further insight into the performance of the models at specific output category level.

It is important to note that the performance metrics for the traditional machine learning algorithms were calculated on the test set. Thus, they only serve as a baseline and the generalization performance of these models is not validated.

3.2.3 Neural network model implementation and accuracy

Various neural network models were implemented on the problem utilizing the full training dataset. The model output Different network architectures were trained and evaluated in order to obtain the optimal model architecture. As well as architectural design considerations, hyperparameters values were optimized by trial-and-error model training and evaluation. Results of hyperparameter and network architecture tests can be found in Appendix A Table 1. Feedforward, simple RNN, GRU & LSTM architectures models were implemented.

The dataset was expanded slightly for the training of the neural network architectures as the number of available projects increases continuously. The initial dataset containing 210 240 data entries was expanded with three additional building projects expanding the dataset to 236 520 entries in total with 28 explanatory features. The data analysis conducted on the previous iteration of the dataset can be assumed to be accurate due to the relatively marginal amount of additional data. Due to the

high autocorrelation indicating a presence of temporal patterns in the consumption time-series, a series of lagged values with shifted overall consumption values were feature engineered into the training dataset. Additional features consisting of lagged overall consumption for a given data-instance were added as input features thus enabling network architectures without feedback connections to learn and leverage temporal patterns. The overall consumption time series feature was used to construct the lagged values. The scope of the lagged values was treated as a tunable hyperparameter value to achieve optimal results. Models were trained with both the original dataset as well as the dataset including the lagged values.

The accuracy was evaluated in manner similar to the traditional machine learning baseline implementation. MAE, RMSE, R2-score as well as category specific MAE for the output categories were calculated as part of the evaluation of the models. The generalization performance of the models was evaluated by utilizing a distinctive validation dataset. The validation dataset is constructed of three standardized buildings and their respective time-series. The three buildings represent three different building categories, apartment building, office, and school, and are used in SWECO internal tool performance evaluations. The validation dataset or any parts of it is not included in the training phase of the models in order to attain a sufficient understanding of model generalization in previously unseen cases.

Initial testing was used to determine network architectures and hyperparameter values with high probability of achieving optimal performance. Hyperparameter tuning was not conducted for model architectures that presented initially drastically weaker performance on the task. Only model architectures that showed best potential were subjected to a comprehensive hyperparameter tuning. This selective hyperparameter tuning was used in order to decrease the overall project computing time to a feasible level. The train-test split was initiated at 0.8. For further hyperparameter testing the split was increased to 0.95 to allow the models to leverage an adequate amount of data from the relatively small overall dataset. Adaptive Adam optimizer was used and allowed the models to converge efficiently.

After initial tests, it was found that shallow networks (<2 hidden layers) with fairly low number of free parameters were not suitable for the tasks. Deeper networks (4 – 5 layers) exhibited significantly better initial performance. Increasing the depth of the network further was found initially not to increase validation performance. The inclusion of lagged values enabled networks to achieve better test and validation performance. Performance was enhanced most significantly with 24-30 time steps of lagged values. This finding is in line with the high 24-hour cycle correlation between overall

consumption found in the autocorrelation analysis. Optimal lagged value time steps were examined by testing different time step values. 30 time steps of lagged values was found to provide best performance.

Tanh, sigmoid and ReLU activation functions were tested and ReLU was found to achieve the best performance. Sigmoid, softmax & linear activations were tested for the output layer and sigmoid was found to achieve best performance. Softmax activation achieved only marginally worse performance.

Data was scaled to increase model performance and different scaling algorithms were tested. In addition to the Standard scaler, Min-Max scaling, Max-Absolute scaler, Quantile Transformer scaler as well as Power Transformer scaling. While all other features were scaled, the DHW circulation value was not scaled. This is due to the linear relationship between the feature and the circulation losses. If the DHW circulation value is obtainable in a real-life example, the value of the building circulation losses can be obtained with a linear function and thus errors in this output category decreased when no scaling transformation was applied to the feature. Power transformer scaling was found to allow models to achieve best performance. Power transforms create monotonic transformations of the scalable data using the Yeo-Johnson transformation. Monotonic transformation stabilizes variance and transforms the data to a more Gaussian-like distribution. Dropout regularization was implemented to the networks to enhance the generalization performance of the models. Various dropout probabilities were tested for various network architectures to survey the optimal dropout probability. A probability of 0.4 produced best results in terms of validation performance.

Table 11. Validation performance of best performing RNN, feedforward network with lagged values and feedforward network without lagged values.

Performance metric	**RNN**	**FFN Lagged values**	**FFN**
Model description	*200-unit LSTM, 4 x 200 unit feedforward layers*	*4 x 200-unit feedforward layers. Lagged values for overall consumption.*	*4 x 200-unit feedforward layers. No aggregated lagged values.*
MAE	0.191	0.0858	0.0814
RMSE	0.265	0.1482	0.1388
r2-score	-	0.65	0.704
Category specific MAE			
Space heating	36.0 %	14.55 %	12.49 %
Ventilation intake air heating	36.1 %	11.33 %	10.54 %
Domestic hot water heating	36.1 %	5.79 %	6.49 %
Domestic hot water circulation losses	36.4 %	2.65 %	3.05 %

Recurrent neural network architectures failed to achieve comparable performance when compared to the feedforward networks with and without lagged value vision. The validation performance of best performing RNN and feedforward networks is presented in Table 11. GRU, SimpleRNN and LSTM layers were tested with a combination of feedforward layer architectures as well as recurrent layers only architectures. LSTM network with one LSTM hidden layer and three feedforward hidden layers achieved the best performance of the recurrent neural network architectures. The LSTM layer had 200 units and feedforward layers had 200 neurons. Dropout was applied after each layer with a dropout probability of 0.4. This Network achieved a test MAE of 0.149 and RMSE of 0.200 with category specific mean absolute errors of 25.7 % for space heating, 24.6 % for ventilation intake air heating, 24.7 % for domestic hot water heating and 24.7 % for domestic hot water circulation losses. The model had a validation MAE of 0.191 and RMSE of 0.265 with category specific mean absolute errors of 36.0 % for space heating, 36.1 % for ventilation intake air heating, 36.1 % for domestic hot water heating and 36.4 % for domestic hot water circulation losses. Even though an architecture with simple RNN hidden layer combined with feedforward layers achieved better validation MAE of 0.184 and RMSE of 0.26, the RNN architecture exhibited significant maximum errors in the output classes, indicating a tendency for high variance of errors making it not suitable solution for the initial problem. The poor performance of recurrent neural networks despite the autoregressive nature of the data can be attributed to the insufficient amount of data compared to the significantly large complexity of the recurrent neural architectures. The number of transformations in recurrent neural networks exceeds the amount of similar sized feedforward network due to the presence of hidden states and the respective additional connections. The relatively small dataset available for the problem examined in this work hinders the ability of complex recurrent neural networks to learn accurate representations of well generalizing patterns. With future expansion of the available dataset, recurrent architectures should be revisited. [47] It is also important to note that the computational requirements for training recurrent neural networks, especially gated layers such as LSTM and GRU, is extremely heavy compared to the feedforward networks, which limits the effectiveness of try-and-test methodology to examine all possible recurrent neural network architectures and hyperparameter sets.

Best performing feedforward network architecture with dataset not containing the lagged values was a four layer deep network with 200 neurons in each layer. Dropout layer with a dropout probability of 0.4 was implemented after each feedforward layer. Batch size of 64 was used, MAE was used as the loss algorithm and the model was trained for 100 epochs. Data was scaled with power transformation Yeo-Johnson transformation. The model achieved a test MAE of 0.0230, RMSE of

0.0454 and r2-score of 0.96. Category specific mean absolute errors for the model on the test set predictions were 3.18 % for space heating, 1.85 % for ventilation intake air heating, 2.61 % for domestic water heating and 1.52 % for domestic hot water circulation losses. The model achieved a validation MAE of 0.08645, RMSE of 0.1472 and r2-score of 0.67 with category specific mean absolute errors of 13.9 % for space heating, 11.64 % for ventilation and intake air heating, 6.35 % for domestic hot water heating and 3.16 % for domestic hot water circulation losses.

The hyperparameter and architecture search was done individually for both lagged dataset and the non-lagged dataset. However, the best performing feedforward network trained on dataset containing the lagged values had the same architecture and hyperparameter layout as the one without lagged values. In addition to the hyperparameters described in the case of the non-lagged dataset, the number of shifts included in the lagged values was 30. The model achieved a test MAE of 0.0210, RMSE of 0.0420 and r2-score of 0.97. Category specific mean absolute errors for the model on the test set predictions were 2.90 % for space heating, 1.87 % for ventilation intake air heating, 2.02 % for domestic water heating and 1.64 % for domestic hot water circulation losses. The model achieved a validation MAE of 0.08611, RMSE of 0.1485 and r2-score of 0.66 with category specific mean absolute errors of 14.61 % for space heating, 11.01 % for ventilation and intake air heating, 5.77 % for domestic hot water heating and 3.05 % for domestic hot water circulation losses.

After determining the best performing model and associated hyperparameter values, the final model was trained on 0 % test set allocation leveraging the entire training set for model training. The best performing network architecture is summarized in Table 12. The feedforward model with and without the lagged values added to the dataset obtained best results. The model trained on the entirety of the dataset with added lagged values achieved a validation MAE of 0.0858, RMSE of 0.1482 and r2 -score of 0.65. Category specific mean absolute validation errors for the model were 14.55 % for space heating, 11.33 % for ventilation intake air heating, 5.79 % for domestic hot water heating and 2.65 % for domestic hot water circulation losses. Model trained on the entire dataset without the added lagged values achieved a validation MAE of 0.0814, RMSE of 0.1388 and r2-score of 0.704. Category specific mean absolute validation errors for the model were 12.49 % for space heating, 10.54 % for ventilation intake air heating, 6.49 % for domestic hot water heating and 3.05 % for domestic hot water circulation losses. Model trained on dataset not containing lagged values has the best overall performance with lowest MAE and RMSE as well as highest r2-score. The NMBE was also calculated for both models and the models achieve a neutral bias in the errors made. For both lagged and non-lagged the NMBE was in the magnitude of $\ll n \times 10^{-10}$ The model trained on lagged values achieves slightly lower max errors throughout the output categories. The differences

can be considered marginal and thus the model trained without lagged values exhibits overall best performance. The best performing final model had 124 004 trainable parameters and its architecture is summarized in Table 12.

Table 12. Summary of the final model architecture.

Layer (type)	Output Shape	Param #
input_data (InputLayer)	[(None, 27)]	0
dense (Dense)	(None, 200)	5600
dropout (Dropout)	(None, 200)	0
dense_1 (Dense)	(None, 200)	40200
dropout_1 (Dropout)	(None, 200)	0
dense_2 (Dense)	(None, 200)	40200
dropout_2 (Dropout)	(None, 200)	0
dense_3 (Dense)	(None, 200)	40200
dropout_3 (Dropout)	(None, 200)	0
output (Dense)	(None, 4)	804

Total params: 127,004
Trainable params: 127,004
Non-trainable params: 0

The model is available for download and use and the instructions for utilizing the model as well as the respective data scaler are presented in Appendix B.

3.2.4 Model performance with respect to end-use case

Reflecting model performance with respect to the end-use case of the model is central. The SWECO building heat disaggregation model is intended as a preliminary assessment tool. The use cases of the model include preliminary evaluation of previously unseen buildings heat use. Resulting model is not intended to be used as the most comprehensive energy calculation tool since the more interpretable and accurate simulation software is more suitable for such tasks. It is also important to note that the inaccuracies of the simulation software energy modelling are inherently built into the trained model, since the training data for the model is constructed using the synthetic dataset originated from the simulation models. Therefore, all inaccuracies and errors of the simulation software are a part of the dataset and the resulting patterns that a data-driven model aims to learn.

The overall validation performance of the model is sufficient for the defined end-use. MAE of 0.0814, RMSE of 0.1388 and r2-score of 0.704 indicate that the model is suitable for preliminary assessment use. Of the output categories, space heating and ventilation intake air heating have the highest mean absolute error. The validation performance metrics of the final model can be found in Table 13

Table 13.

Table 13. Final model validation performance.

Performance metric	Model validation performance
MAE	0.0814
RMSE	0.1388
r2-score	0.704
Category specific MAE	
Space heating	12.49 %
Ventilation intake air heating	10.54 %
Domestic hot water heating	6.49 %
Domestic hot water circulation losses	3.05 %

The model predictions on an example building from the validation dataset are plotted in Figure 8. The model outputs the relative shares of the four categories of the overall heat consumption and thus the predictions are not in energy quantity or power units. This is important to note when examining the predictions. The predictions are plotted for a two-week time interval. As can be seen in Figure 8, the model captures the consumption trends in all four output categories and the scale of predicted values is accurate. However, the maximum errors between validation values are high in all four output categories. Through the entire validation dataset, the maximum errors were 87 % for space heating, 86 % for ventilation intake air heating, 80 % for domestic hot water use and 41 % for domestic hot water circulation losses. Clear borderline cases, where consumption share shifts are rapid, are vulnerable to proportionally high errors for a short period of time instances. As a preliminary assessment tool, this might not be problematic but should be considered thoroughly if model is used in other end-use cases.

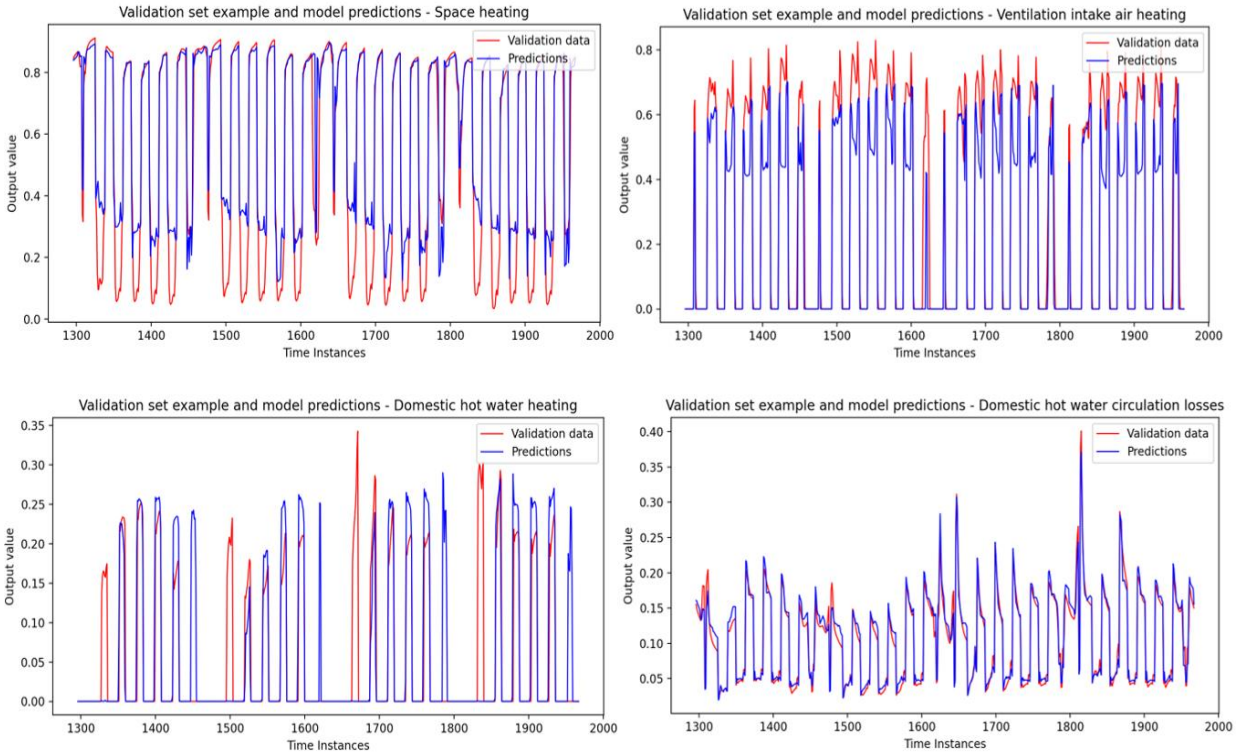

Figure 8. Model predictions and validation data on an example building.

The model trained in this work performs the heat disaggregation task sufficiently. The model accuracy is sufficient with respect to the preliminary end-use case. The prediction errors of the model are of similar or smaller magnitude than the expected quality of data in real-life cases, with deviation accuracies of 10 % expected to be common [6]. However, the model is not suitable for comprehensive and final analysis of a buildings heat use assessment due to the inherit inaccuracies in the synthetic dataset and high maximum error. While this is not seen as an obstacle for model deployment in the defined use case, it is important to note for potential future applications. However, it is crucial to note that a more comprehensive evaluation of the model's performance should take place if the model is considered to be used outside the specified task.

The relatively small dataset is considered as the preliminary obstacle for further model performance increases. If further increases in performance in this task as well as potential future tasks in this domain are to be sought, expanding the dataset is a key factor. Expanding the dataset to also include real-life data, would greatly increase data-driven model generalization performance. The computational costs of training the models in this work can be stated to be marginal. Re-evaluating architectures, hyperparameter values and re-training proven architectures is recommended, if the dataset is expanded to be more comprehensive.

3.2.5 Performance with respect to industry standards

Performance of a model can be evaluated by various metrics. Performance metrics should be chosen so that they reflect the performance with respect to the end-use case generalization in order to obtain a sufficient understanding of the model performance in the potential deployment environment. Additionally, the performance of energy disaggregation and forecasting models should ideally be comparable with other models in the domain. Several methodologies for assessing model performance and accuracy have been developed in order to retain the ability for intersectional and comparable evaluation of distinct models and their applications. ASHRAE (American Society of Heating Refrigerating and Air-Conditioning Engineers) guideline 14, the International Performance Measurement and Verification protocol (IPMVP) and Federal Energy Management Program (FEMP) are generally accepted methodologies for assessing energy model performance in a uniform manner [18].

FEMP criteria, IPMVP and ASHRAE guideline 14 recommend the use of normalized mean bias error (NMBE) and root-mean-square deviation (RMSE) for model calibration criteria. Hourly and monthly criteria differ with hourly prediction criteria recommended limit values being generally higher compared to monthly prediction criteria. FEMP and ASHRAE guideline 14 state a 30 % RMSE and +/- 10 % NMBE as criteria for hourly predictions. IPMVP recommends 20 % RMSE and +/- 5 % NMBE, respectively. ASHRAE guideline 14 and IPMVP additionally set a recommended r2 -score of >0.75 for evaluating model goodness of fit. The values can be found in Table 14.

Table 14. Calibration criteria of FEMP, ASHRAE & IPMVP.

Data type	Index	FEMP	ASHRAE	IPMVP
Calibration criteria				
Monthly %	NMBE	+/- 5	+/- 5	+/- 20
	RMSE	15	15	-
Hourly %	NMBE	+/- 10	+/- 10	+/- 5
	RMSE	30	30	20
Model recommendation				
	r2-score	-	> 0.75	> 0.75

The best performing model achieves a NMBE and RSME lower than the FEMP, ASHRAE & IPMVP recommendations. The models r2-score of 0.67 is below the threshold of ASHRAE and IPMVP. It is important to note that r2-score is sensitive to outliers and linearity assumption between dependent and independent variables. The overall low MAE, RMSE and max errors indicate that the model is performing well in this specific task. It is important to note that the model is trained and validated on

synthetic data and thus caution should be taken in the comparison of FEMP, ASHRAE and IPMVP guidelines with respect to model performance.

4 Conclusions

The aim of this work was to develop a data-driven model for a building heat disaggregation task. The foundations of building heating systems and machine learning were reviewed. Literary review of data-driven models examined in the energy prediction and forecasting as well as heat disaggregation tasks was conducted. The findings were assessed with respect to the SWECO building heat disaggregation problem and key design factors for the model development were identified. A dataset was constructed from IDA ICE simulations from the SWECO project archive. The dataset was analyzed and engineered to serve as a training dataset for data-driven models. Traditional and deep learning algorithms were trained and evaluated, and the final model performance was assessed with respect to the end-use case.

The literary review showed that a variety of machine learning algorithms have been deployed successfully in the energy domain. Several promising data-driven algorithms were identified, and neural networks showed promising results in both energy disaggregation as well as energy prediction and forecasting tasks. Recurrent neural networks with feedback connections were found to outperform feedforward networks in most cases with especially LSTM-networks have showing promising results.

The training data of the model was constructed from IDA ICE building simulations sourced from SWECO project archives. The dataset contained 210 240 instances with 27 explanatory features and four labels. The data analysis showed a high variability of scale between variables, suggesting a need for training data scaling. Autoregression analysis confirmed a presence of strong temporal correlations for the energy consumption, which is in line with energy time-series data in general. Dataset was aggregated to contain lagged values of the overall consumption to allow models without feedback connections to leverage the temporal correlations. The autoregressive nature of the data also further motivated the testing of recurrent neural network architectures. Traditional machine learning algorithms were trained and evaluated as baseline models to assess the suitability of data-driven models for the task.

A variety of neural network architectures were trained and evaluated. Network architectures and hyperparameter values were tried and tested to obtain best performing model structure. Shallow networks (<2 hidden layers) failed to learn required representations from the dataset. Deeper networks with four hidden layers showed best performance. The dataset containing lagged values initially showed promising results and the number of lags was treated as a tunable hyperparameter. For majority of feedforward networks, the optimal value of lags was determined to be 30 time-

intervals. Various scaling algorithms were tested and power transformer utilizing Yeo-Johnson transformation was found to allow models to achieve best validation performance. Inclusion of dropout layers was found to increase model validation performance and the optimal dropout probability was found to be 0.4. Recurrent neural networks did not achieve a performance comparable to less-complex feedforward neural networks. The relatively small training dataset was identified as one of possible explanations to the poor performance.

The final model had four hidden layers and an output layer. The hidden layers had 200 neurons each and the output layer had 4. Dropout with a dropout probability of 0.4 was included after each hidden layer. ReLU activation was used for the hidden layers and softmax for the output layer. The model had 127 004 trainable parameters. Model achieved a validation MAE of 0.0814, RMSE of 0.1388 and r2-score of 0.704. Space heating and ventilation intake air heating had highest output category specific MAE's. The maximum errors made by the model over the entire validation dataset were high. The high maximum errors and relatively low r2-score need to be further considered if alternative end-use cases are considered for the model.

The model was assessed to perform the SWECO heat disaggregation task sufficiently with respect to the preliminary assessment nature of the end-use case of the model. Expanding the relatively small dataset is identified as key factor in increasing data-driven model performance in this and other related tasks. The inclusion of real-life data into the currently entirely synthetic dataset is also identified as a key factor for increasing model generalization performance. Computational costs for models trained in the work are stated to be marginal and re-evaluating model architectures and hyperparameters values is recommended if the dataset is expanded.

References

[1] Motiva, “Rakentaminen ja rakennukset,” Nov. 01, 2022.

[2] Statistics Finland, “Statistics Finland’s statistical database - Energy,” 2022.

[3] T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang, and H. Zareipour, “Energy Forecasting: A Review and Outlook,” *IEEE Open Access Journal of Power and Energy*, vol. 7. Institute of Electrical and Electronics Engineers Inc., pp. 376–388, 2020. doi: 10.1109/OAJPE.2020.3029979.

[4] J. Runge and R. Zmeureanu, “Forecasting energy use in buildings using artificial neural networks: A review,” *Energies*, vol. 12, no. 17. MDPI AG, Aug. 23, 2019. doi: 10.3390/en12173254.

[5] A. Vadiee and L. Gustavsson, “A comparison between four dynamic energy modeling tools for simulation of space heating demand of buildings.”

[6] SWECO, “Joni Hilpinen, kehityspäällikkö,” *Discussions about energy modelling in engineering consulting.* 2023.

[7] J. D. Racusin, *Essential building science : understanding energy and moisture in high performance house design*. in Sustainable Building Essentials. Gabriola, BC: New Society Publishers, 2016.

[8] Motiva, “Kaukolämpö,” 2022.

[9] Energiateollisuus Ry, “Kvalitatiivinen asiakastutkimus lämmitysmuotovalinnan päätöksentekoprosessista: Tutkimuskokonaisuus kaukolämpöalalle,” Apr. 2020.

[10] M. Lumbreras *et al.*, “Data driven model for heat load prediction in buildings connected to District Heating by using smart heat meters,” *Energy*, vol. 239, p. 122318, Jan. 2022, doi: 10.1016/J.ENERGY.2021.122318.

[11] M. Z. Pomianowski, H. Johra, A. Marszal-Pomianowska, and C. Zhang, “Sustainable and energy-efficient domestic hot water systems: A review,” *Renewable and Sustainable Energy Reviews*, vol. 128, p. 109900, Aug. 2020, doi: 10.1016/J.RSER.2020.109900.

[12] Ympäristöministeriön asetus, “Rakennuksen energiankulutuksen ja lämmitystehontarpeen laskenta,” Helsinki, 2007.

[13] I. Knight, N. Kreutzer, M. Manning, M. Swinton, and H. Ribberink, “European and Canadian non-HVAC electric and DHW load profiles for use in simulating the performance of residential cogeneration systems,” May 2007.

[14] Z. Wang, T. Hong, and M. A. Piette, “Building thermal load prediction through shallow machine learning and deep learning,” *Appl Energy*, vol. 263, Apr. 2020, doi: 10.1016/j.apenergy.2020.114683.

[15] Z.-H. Zhou, *Machine Learning*. Singapore: Springer Singapore, 2021. doi: 10.1007/978-981-15-1967-3.

[16] Goodfellow Ian, Bengio Yoshua, and Courville Aaron, *Deep Learning*. Massachutes: MIT Press, 2016.

[17] A. Glassner, *Deep Learning: a visual approach*. San Francisco: No Starch Press, 2021.

[18] G. R. Ruiz and C. F. Bandera, "Validation of calibrated energy models: Common errors," *Energies (Basel)*, vol. 10, no. 10, Oct. 2017, doi: 10.3390/en10101587.

[19] B. Macukow, "Neural networks-state of art, brief history, basic models and architecture," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2016, pp. 3–14. doi: 10.1007/978-3-319-45378-1_1.

[20] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," Jan. 2020, [Online]. Available: http://arxiv.org/abs/2001.05566

[21] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Commun ACM*, vol. 64, no. 3, pp. 107–115, Mar. 2021, doi: 10.1145/3446776.

[22] X. Dong and L. Zhou, "Understanding over-parameterized deep networks by geometrization," Feb. 2019, [Online]. Available: http://arxiv.org/abs/1902.03793

[23] N. Zaeri, A. Ashouri, H. B. Gunay, and T. Abuimara, "Disaggregation of electricity and heating consumption in commercial buildings with building automation system data," *Energy Build*, vol. 258, Mar. 2022, doi: 10.1016/j.enbuild.2021.111791.

[24] D. W. Kim, K. U. Ahn, H. Shin, and S. E. Lee, "Simplified Weather-Related Building Energy Disaggregation and Change-Point Regression: Heating and Cooling Energy Use Perspective," *Buildings*, vol. 12, no. 10, Oct. 2022, doi: 10.3390/buildings12101717.

[25] N. Batra, A. Singh, and K. Whitehouse, "Neighbourhood NILM: A Big-data Approach to Household Energy Disaggregation," Oct. 2015, [Online]. Available: http://arxiv.org/abs/1511.02900

[26] S. Paresh, N. K. Thokala, and M. G. Chandra, "Electrical load disaggregation using a two-stage deep learning approach," in *BuildSys 2019 - Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, Association for Computing Machinery, Inc, Nov. 2019, pp. 366–367. doi: 10.1145/3360322.3361003.

[27] I. Rahman, M. Kuzlu, and S. Rahman, "Power disaggregation of combined HVAC loads using supervised machine learning algorithms," *Energy Build*, vol. 172, pp. 57–66, Aug. 2018, doi: 10.1016/j.enbuild.2018.03.074.

[28] A. Tongta and K. Chooruang, “Long Short-Term Memory (LSTM) Neural Networks Applied to Energy Disaggregation,” in *2020 8th International Electrical Engineering Congress, iEECON 2020*, Institute of Electrical and Electronics Engineers Inc., Mar. 2020. doi: 10.1109/iEECON48109.2020.229559.

[29] M. Kaselimi, N. Doulamis, A. Voulodimos, E. Protopapadakis, and A. Doulamis, “Context Aware Energy Disaggregation Using Adaptive Bidirectional LSTM Models,” *IEEE Trans Smart Grid*, vol. 11, no. 4, pp. 3054–3067, Jul. 2020, doi: 10.1109/TSG.2020.2974347.

[30] W. He and Y. Chai, “An Empirical Study on Energy Disaggregation via Deep Learning,” 2016.

[31] V. Piccialli and A. M. Sudoso, “Improving non-intrusive load disaggregation through an attention-based deep neural network,” *Energies (Basel)*, vol. 14, no. 4, Feb. 2021, doi: 10.3390/en14040847.

[32] A. Vaswani *et al.*, “Attention Is All You Need.”

[33] H. Y. Park, B. H. Lee, J. H. Son, and H. S. Ahn, “A comparison of neural network-based methods for load forecasting with selected input candidates,” in *Proceedings of the IEEE International Conference on Industrial Technology*, Institute of Electrical and Electronics Engineers Inc., Apr. 2017, pp. 1100–1105. doi: 10.1109/ICIT.2017.7915516.

[34] M. Mordjaoui, S. Haddad, A. Medoued, and A. Laouafi, “Electric load forecasting by using dynamic neural network,” *Int J Hydrogen Energy*, vol. 42, no. 28, pp. 17655–17663, Jul. 2017, doi: 10.1016/j.ijhydene.2017.03.101.

[35] J. Moon, S. Park, S. Rho, and E. Hwang, “A comparative analysis of artificial neural network architectures for building energy consumption forecasting,” *Int J Distrib Sens Netw*, vol. 15, no. 9, Sep. 2019, doi: 10.1177/1550147719877616.

[36] C. Fan, F. Xiao, and Y. Zhao, “A short-term building cooling load prediction method using deep learning algorithms,” *Appl Energy*, vol. 195, pp. 222–233, 2017, doi: 10.1016/j.apenergy.2017.03.064.

[37] Ö. F. Ertugrul, “Forecasting electricity load by a novel recurrent extreme learning machines approach,” *International Journal of Electrical Power and Energy Systems*, vol. 78, pp. 429–435, Jun. 2016, doi: 10.1016/j.ijepes.2015.12.006.

[38] N. Zeng, H. Zhang, W. Liu, J. Liang, and F. E. Alsaadi, “A switching delayed PSO optimized extreme learning machine for short-term load forecasting,” *Neurocomputing*, vol. 240, pp. 175–182, May 2017, doi: 10.1016/j.neucom.2017.01.090.

[39] S. Li, L. Goel, and P. Wang, “An ensemble approach for short-term load forecasting by extreme learning machine,” *Appl Energy*, vol. 170, pp. 22–29, May 2016, doi: 10.1016/j.apenergy.2016.02.114.

[40] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, “Deep learning for estimating building energy consumption,” *Sustainable Energy, Grids and Networks*, vol. 6, pp. 91–99, Jun. 2016, doi: 10.1016/j.segan.2016.02.005.

[41] G. Mohi, U. Din, and A. K. Marnerides, “Short Term Power Load Forecasting Using Deep Neural Networks.” [Online]. Available: http://www.iso-

[42] A. Dedinec, S. Filiposka, A. Dedinec, and L. Kocarev, “Deep belief network based electricity load forecasting: An analysis of Macedonian case,” *Energy*, vol. 115, pp. 1688–1700, Nov. 2016, doi: 10.1016/j.energy.2016.07.090.

[43] S. Ryu, J. Noh, and H. Kim, “Deep neural network based demand side short term load forecasting,” *Energies (Basel)*, vol. 10, no. 1, 2017, doi: 10.3390/en10010003.

[44] K. Amarasinghe, D. L. Marino, and M. Manic, “Deep Neural Networks for Energy Load Forecasting.”

[45] L. Saad Saoud, H. Al-Marzouqi, and R. Hussein, “Household Energy Consumption Prediction Using the Stationary Wavelet Transform and Transformers,” *IEEE Access*, vol. 10, pp. 5171–5183, 2022, doi: 10.1109/ACCESS.2022.3140818.

[46] S. Sharma, S. Sharma, and A. Athaiya, “ ACTIVATION FUNCTIONS IN NEURAL NETWORKS,” 2020. [Online]. Available: http://www.ijeast.com

[47] S. Zhang *et al.*, “Architectural Complexity Measures of Recurrent Neural Networks.”

APPENDICES

Appendix A

Test description	Model description	MEA	RMSE	MAE (%) Space heating	MAE (%) Ventilation intake heating	MAE (%) Domestic hot water heating	MAE (%) Domestic hot water circulation losses	Architecture & hyperparameters
Dropout tests	Feedforward no memory	0.073	0.107	13.78	1.01	11.44	3.101	(200, 200, 200, 200, 4), ReLU, Dropout 0.2
	Validation	0.1315	0.2083	15.25	15.05	8.85	13.44	(200, 200, 200, 200, 4), ReLU, Dropout 0.2
	Feedforward no memory	0.076	0.117	12.6	7.18	8.56	14.31	(200, 200, 200, 200, 4), ReLU, Dropout 0.3
	Validation	0.1315	0.212	15.1	13.9	8.6	15.6	(200, 200, 200, 200, 4), ReLU, Dropout 0.3
	Feedforward no memory	0.069	0.961	12.1	0.5	11.4	3.39	(200, 200, 200, 200, 4) ReLU, Dropout 0.4, mae
	Validation	0.1315	0.21	16.21	13.65	9.04	13.72	(200, 200, 200, 200, 4) ReLU, Dropout 0.4, mae
	Feedforward no memory	0.0858	0.127	17.3	1.15	11.7	3.7	(200, 200, 200, 200, 4), ReLU, Dropout 0.5
	Validation	0.136	0.21	15.5	13.8	9.39	14.2	(200, 200, 200, 200, 4), ReLU, Dropout 0.5
	Feedforward no memory	0.104	0.139	18.77	5.77	12.37	4.68	(200, 200, 200, 200, 4), ReLU, Dropout 0.6
	Validation	0.138	0.217	17.2	15.2	10.2	12.7	(200, 200, 200, 200, 4), ReLU, Dropout 0.6
	Feedforward no memory	9.91	12.71	16.84	0.67	14.13	4.76	(200, 200, 200, 200, 4), ReLU, Dropout 0.7
	Validation	0.1423	0.22696	15.73	15.02	11.28	14.9	(200, 200, 200, 200, 4), ReLU, Dropout 0.7
	Feedforward lagged vision	0.698	0.1081	9.77	6.61	7.81	3.72	(200, 200, 200, 200, 4), ReLU, Dropout 0.2
	Validation	11.99	0.2052	14.07	12.08	8.87	12.95	(200, 200, 200, 200, 4), ReLU, Dropout 0.2
	Feedforward lagged vision	0.0667	0.0975	10.1	6.1	7.62	2.84	(200, 200, 200, 200, 4), ReLU, Dropout 0.3
	Validation	0.1268	0.2093	14.6	12.9	9.29	13.87	(200, 200, 200, 200, 4), ReLU, Dropout 0.3
	Feedforward lagged vision	0.063	0.099	9.07	3.76	8.31	4.08	(200, 200, 200, 200, 4), ReLU, Dropout 0.4
	Validation	0.119	0.189	13.5	13.1	10.4	10.6	(200, 200, 200, 200, 4), ReLU, Dropout 0.4
	Feedforward lagged vision	0.083	0.119	14.2	4.52	9.51	4.99	(200, 200, 200, 200, 4), ReLU, Dropout 0.5
	Validation	0.124	0.204	14.7	13.8	8.93	12.2	(200, 200, 200, 200, 4), ReLU, Dropout 0.5
	Feedforward lagged vision	0.0809	0.119	14.07	4.96	8.79	4.55	(200, 200, 200, 200, 4), ReLU, Dropout 0.6
	Validation	0.117	0.1964	14.11	13.12	8.22	11.66	(200, 200, 200, 200, 4), ReLU, Dropout 0.6
	Feedforward lagged vision	0.0777	0.1148	12.25	1.706	11.5	5.64	(200, 200, 200, 200, 4), ReLU, Dropout 0.7
	Validation	0.1283	0.2168	14.65	13.44	9.39	13.84	(200, 200, 200, 200, 4), ReLU, Dropout 0.7
Layer & neuron configuration tests	Feedforward no memory	0.0777	0.117	13.38	1.65	11.24	4.79	(100, 100, 100, 100, 100, 100, 100, 100), ReLU, dropout 0.4
	Validation	0.124	0.2016	14.62	13.64	9.43	11.84	(100, 100, 100, 100, 100, 100, 100, 100), ReLU, dropout 0.4
	Feedforward lagged vision	0.0612	0.0939	9.15	1.86	8.61	4.85	(100, 100, 100, 100, 100, 100, 100, 100), ReLU, dropout 0.4
	Validation	0.1217	0.202	13.55	14.16	8.9	12.05	(100, 100, 100, 100, 100, 100, 100, 100), ReLU, dropout 0.4
	Feedforward no memory	0.08233	0.12796	14.18	3.048	9.982	5.723	(200, 200, 200, 200, 200, 200, 4), ReLU, dropout 0.4
	Validation	0.1325	0.209	15.05	13.63	10.33	14	(200, 200, 200, 200, 200, 200, 4), ReLU, dropout 0.4
	Feedforward lagged vision	0.0683	0.106	10.17	1.64	10.01	5.52	(200, 200, 200, 200, 200, 200, 4), ReLU, dropout 0.4
	Validation	11.35	19.29	13.12	11.92	8.38	11.99	(200, 200, 200, 200, 200, 200, 4), ReLU, dropout 0.4
Activation functions	Feedforward no memory	0.0758	0.1052	13.21	0.97	11.26	19.43	(200, 200, 200, 200, 4), tanh, dropout 0.4
	Validation	0.167	0.27	21.5	14.7	11.25	19.43	(200, 200, 200, 200, 4), tanh, dropout 0.4
	Feedforward lagged vision	0.0555	0.08045	8.85	0.8	7.38	5.13	(200, 200, 200, 200, 4), tanh, dropout 0.4
	Validation	0.136	0.2254	16.48	13.43	9.16	15.31	(200, 200, 200, 200, 4), tanh, dropout 0.4
	Feedforward no memory	0.0898	0.1189	16.9	0.61	14.43	4	(200, 200, 200, 200, 4), sigmoid, dropout 0.4
	Validation	0.128	0.2126	15.48	14.78	8.68	12.29	(200, 200, 200, 200, 4), sigmoid, dropout 0.4
	Feedforward lagged vision	0.77	0.106	12.77	0.35	12.06	5.63	(200, 200, 200, 200, 4), sigmoid, dropout 0.4
	Validation	0.1261	0.2085	13.89	15.61	9.49	11.45	(200, 200, 200, 200, 4), sigmoid, dropout 0.4
Ouput layer activations	Feedforward no memory	0.081	0.1279	13.63	3.88	10.8	4	(200, 200, 200, 200, 4), sofmax, dropout 0.4
	Validation	0.1438	0.2238	17.68	15.05	10.49	14.32	(200, 200, 200, 200, 4), sofmax, dropout 0.4
	Feedforward lagged vision	0.0637	0.10037	9.22	2.3	7.71	6.26	(200, 200, 200, 200, 4), sofmax, dropout 0.4
	Validation	0.1196	0.1947	13.6	12.41	9.668	12.13	(200, 200, 200, 200, 4), sofmax, dropout 0.4
	Feedforward no memory	0.0945	0.1318	16.15	1.18	14.38	6.08	(200, 200, 200, 200, 4), linear, dropout 0.4
	Validation	0.1464	0.227	16.95	14.11	10.617	16.91	(200, 200, 200, 200, 4), linear, dropout 0.4
	Feedforward lagged vision	0.055	0.0827	8.4	0.68	6.88	6.223	(200, 200, 200, 200, 4), linear, dropout 0.4
	Validation	0.1271	0.203	15.07	13.26	9.4	13.11	(200, 200, 200, 200, 4), linear, dropout 0.4
Recurrent neural nets	LSTM	0.115	0.174	12.2	10.2	21.1	10.6	(lstm(180), 200, 200, 200, 200. 200), ReLU, Dropout 0.4
	Validation	0.2264	0.31634	29.08	16.65	18.21	27.38	(lstm(180), 200, 200, 200, 200. 200), ReLU, Dropout 0.4
	LSTM	0.1933	0.2732	29.08	24.02	37.25	29	(lstm(64), lstm(64), lstm(64))
	Validation	0.199	0.269	22.4	24.9	27.7	19.3	(lstm(64), lstm(64), lstm(64))
	LSTM	0.156	0.206	27.3	26.8	26.8	26.9	LSTM without lagged (lstm(200), dropout(0.4), 200, dropout(0.2), 200, dropout(0.4), 200, dropout(0.4), 4), epochs=15, mse, 0.95, batch_size=32
	validation	0.225	0.274	32.8	32.3	32.4	32.3	LSTM without lagged (lstm(200), dropout(0.4), 200, dropout(0.2), 200, dropout(0.4), 200, dropout(0.4), 4), epochs=15, mse, 0.95, batch_size=32
	LSTM lagged vision	0.1304	0.1897	19.02	19.67	14.38	18.35	LSTM MAE with lagged (lstm(200), dropout(0.4), 200, dropout(0.2), 200, dropout(0.4), 200, dropout(0.4), 4), epochs=15, mse, 0.95, batch_size=32
	validation	0.1996	0.277	64.07	69.14	74.15	78.75	LSTM MAE with lagged (lstm(200), dropout(0.4), 200, dropout(0.2), 200, dropout(0.4), 200, dropout(0.4), 4), epochs=15, mse, 0.95, batch_size=32
	LSTM	0.149	0.20021	25.7	24.6	24.7	24.7	LSTM MAE without lagged (lstm(200), dropout(0.4), 200, dropout(0.2), 200, dropout(0.4), 200, dropout(0.4), 4), epochs=15, mse, 0.95, batch_size=32
	validation	0.191	0.265	36	36.1	36.1	36.4	LSTM MAE without lagged (lstm(200), dropout(0.4), 200, dropout(0.2), 200,

								dropout(0.4), 200, dropout(0.4), 4), epochs=15, mse, 0.95, batch_size=32
	SimpleRNN lagged vision	0.137	0.191	21.87	21.61	17.1	18.62	RNN with lagged (rnn(200), 200, 200, 200, 4), epochs=30, mae, batch_size=32
	Validation	0.199	0.274	73.6	67.9	75.9	59.3	RNN with lagged (rnn(200), 200, 200, 200, 4), epochs=30, mae, batch_size=32
	SimpleRNN	0.1619	0.2074	27.6	27.6	27.8	27.8	RNN without lagged (rnn(200), 200, 200, 200, 4), epochs=30, mae, batch_size=32
	Validation	0.199	0.269	35.7	35.6	35.8	36.2	RNN without lagged (rnn(200), 200, 200, 200, 4), epochs=30, mae, batch_size=32
	SimpleRNN lagged vision	0.2	0.27	42.2	34.4	68.7	31.7	RNN with lagged (rnn(200), 200, 200, 200, 4), epochs=30, mae, batch_size=32
	Validation	0.184	0.26	54.3	37.2	63.4	37.1	RNN with lagged (rnn(200), 200, 200, 200, 4), epochs=30, mae, batch_size=32
	SimpleRNN	0.128	0.18	29.7	29.5	27.9	27.9	RNN without lagged (rnn(200), 200, 200, 200, 4), epochs=30, mae, batch_size=32
	Validation	0.207	0.269	36.45	36.2	35.7	35.7	RNN without lagged (rnn(200), 200, 200, 200, 4), epochs=30, mae, batch_size=32
	GRU	0.213	0.303	54.16	57.45	59.76	61.44	GRU without lagged (gru(64), gru(64), gru(64)), dropout 0.2
	Validation	0.227	0.315	45.61	49.12	51.74	53.78	GRU without lagged (gru(64), gru(64), gru(64)), dropout 0.2
	GRU lagged vision	0.2169	0.302	40.89	13.5	45.58	47.22	GRU with lagged (gru(64), gru(64), gru(64)), dropout 0.2
	Validation	0.229	0.316	44.44	47.36	49.55	51.29	GRU with lagged (gru(64), gru(64), gru(64)), dropout 0.2
	LSTM	0.206	0.297	30.86	20.9	16.67	13.4	LSTM(64), LSTM(64), LSTM(64), ReLU, dropout 0.4, PowerTransformer scaling
	Validation	0.22	0.3201	30.93	22.22	18.9	16.6	LSTM(64), LSTM(64), LSTM(64), ReLU, dropout 0.4, PowerTransformer scaling
Lagged time-steps tuning	Feedforward 12 shifts lagged vision	0.0931	0.1263	16.8	2.46	11.65	6.35	(200, 200, 200, 200, 4), ReLU, dropout 0.4
	Validation	0.1277	0.202	15.91	13.77	9.59	11.79	(200, 200, 200, 200, 4), ReLU, dropout 0.4
	Feedforward 18 shifts lagged vision	0.06915	0.098655	12.315	1.05	8.92	5.38	(200, 200, 200, 200, 4), ReLU, dropout 0.4
	Validation	0.1322	0.2155	15.67	12.86	9.69	14.69	(200, 200, 200, 200, 4), ReLU, dropout 0.4
	Feedforward 30 shifts lagged vision	0.0706	0.0984	13.03	1.13	8.93	5.13	(200, 200, 200, 200, 4), ReLU, dropout 0.4
	Validation	0.1232	0.1969	14.48	13.14	8.89	12.76	(200, 200, 200, 200, 4), ReLU, dropout 0.4
	Feedforward 36 shifts lagged vision	0.059996	0.08034	10.17	1.09	8.5	4.26	(200, 200, 200, 200, 4), ReLU, dropout 0.4
	Validation	0.12987	0.20473	15.22	14	9.39	13.34	(200, 200, 200, 200, 4), ReLU, dropout 0.4
Feedforward with relative scaling	Feedforward							(200, 200, 200, 200, 4), ReLU, dropout 0.4, StandardScaler(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae
	Validation	0.112423	0.20435	14.03	11.96	10.13	8.84	(200, 200, 200, 200, 4), ReLU, dropout 0.4, StandardScaler(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae
	Feedforward with lagged vision							(200, 200, 200, 200, 4), ReLU, dropout 0.4, StandardScaler(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae
	Validation	0.11295	0.201185	15.69	11.94	10.31	7.24	(200, 200, 200, 200, 4), ReLU, dropout 0.4, StandardScaler(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae
	Feedforward							(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae
	Validation	0.10927	0.19482	13.14	11.65	9.83	9.08	(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae
	Feedforward with lagged vision							(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae
	Validation	0.10801	0.19041	14.85	12.15	9.53	6.68	(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae
	Feedforward							(200, 200, 200, 200, 4), ReLU, dropout 0.4, StandardScaler(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Validation	0.10971	0.198449	12.55	12.02	9.82	9.49	(200, 200, 200, 200, 4), ReLU, dropout 0.4, StandardScaler(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Feedforward with lagged vision							(200, 200, 200, 200, 4), ReLU, dropout 0.4, StandardScaler(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Validation	0.106532	0.19326	12.94	12.35	8.19	9.12	(200, 200, 200, 200, 4), ReLU, dropout 0.4, StandardScaler(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Feedforward							(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Validation	0.106125	0.1929312	12.29	12.45	8.6	9.11	(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Feedforward with lagged vision							(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Validation	0.105505	0.18623506	15.71	11.98	8.85	5.66	(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30

	Feedforward							(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=28
	Validation	0.106125	0.1929312	12.29	12.45	8.6	9.11	(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=28
	Feedforward with lagged vision							(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=28
	Validation	0.1163	0.2014	17.18	12.82	10.11	6.33	(200, 200, 200, 200, 4), ReLU, dropout 0.4, MinMaxScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=28
Scaling methods	Feedforward							
	Validation	0.1265	0.23088	16.45	13.07	11.87	9.22	(200, 200, 200, 200, 4), ReLU, dropout 0.4, MaxAbsScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Feedforward with lagged vision							
	Validation	0.1215	0.2135	19.65	12.83	10.86	5.26	(200, 200, 200, 200, 4), ReLU, dropout 0.4, MaxAbsScale(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Feedforward							
	Validation	0.088993	0.154148	14.6	11.41	6.63	2.8	(200, 200, 200, 200, 4), ReLU, dropout 0.4, PowerTransformerScaling(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Feedforward with lagged vision							
	Validation	0.08858	0.15348	14.91	11.55	5.79	3.17	(200, 200, 200, 200, 4), ReLU, dropout 0.4, PowerTransformerScaling(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Feedforward							
	Validation	0.09725	0.1740385	14.64	12.54	7.11	4.62	(200, 200, 200, 200, 4), ReLU, dropout 0.4, QuantileTransformation(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30
	Feedforward with laged vision							
	Validation	0.09644	0.1738	16.03	12.25	7.05	3.25	(200, 200, 200, 200, 4), ReLU, dropout 0.4, QuantileTransformation(heating constant not scaled), 0.95 train size, batch_size=64, epochs=40, mae, shifts=30

Appendix A Table 1. Neural network architecture & hyperparameter testing results.

Appendix B

The final model is open source and free for use. The final model as well as the data scaler can be downloaded here. The model is available as an h5 -file and can be utilized with most deep learning frameworks and libraries. Additionally, the data scaler is also available for download and use in the same folder as a Pickle -file. It is important to note that the data must be formatted correctly for the model to perform expectedly. Additionally, the data scaler available must be used for the model to perform expectedly.