Diese Arbeit wurde vorgelegt am
Lehr- und Forschungsgebiet Theorie der hybriden Systeme

Optimale Klimasteuerung in Gebäuden Unter Einsatz von Künstlicher Intelligenz

Optimal Climate Control in Buildings Using Artificial Intelligence

Bachelorarbeit
Informatik

September 2019

Vorgelegt von Presented by	Kristina Yaneva Am Weißenberg 18 52074 Aachen Matrikelnummer: 371610 kristina.yaneva@rwth-aachen.de
Erstprüfer First examiner	Prof. Dr. rer. nat. Erika Ábrahám Lehr- und Forschungsgebiet: Theorie der hybriden Systeme RWTH Aachen University
Zweitprüfer Second examiner	Prof. Gerhard Lakemeyer, Ph.D. Lehr- und Forschungsgebiet: Wissensbasierte Systeme RWTH Aachen University
Externer Betreuer External supervisor	Dr. rer. nat. Pascal Richter Steinbuch Centre for Computing Karlsruhe Institute of Technology

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen und Abbildungen. Diese Arbeit hat in dieser oder einer ähnlichen Form noch nicht im Rahmen einer anderen Prüfung vorgelegen.

Aachen, im September 2019

Kristina Yaneva

Contents

1 Introduction

1.1 Background

The building construction sector is responsible for the largest energy consumption in the world (35%) and is an important source of carbon dioxide (CO_2) emissions (nearly 40%) [9]. The clear upward trend in energy demand is expected to continue in the future since phenomena like population growth, increasing demand for building services and comfort levels, together with the rise in time spent inside buildings, are from vital importance for humanity and must not be neglected. Buildings' final energy consumption grows steadily as a result of increasing floor area growth, outpacing energy intensity reduction [11]. Therefore, it is of fundamental significance to control the existing Heating Ventilation and Air Conditioning (HVAC) systems efficiently, since they are the primary energy consumption factor (50% of building consumption) and in most cases, these are poorly handled [47]. As an important part of residential structures, medium to large industrial and office buildings, their two main requirements are to provide safe and healthy building conditions, which are regulated with respect to temperature and humidity, using fresh air from outdoors, and to minimize overall energy consumption [41]. The digitalization and the ability to capture and analyze large data sets are enabling these tasks. Smart buildings improve efficiency, sustainability, and comfort, leveraging IoT technology, sensors, and wireless connectivity, which provide the required energy use data. While increasing the efficiency of the heating system, they also deliver cost savings - goals of property owners, managers, and tenants. Reducing building energy costs has become an urgent task due to increasing environmental concerns and energy prices.

1.2 Related Work

In recent years, predictive control techniques for HVAC systems have been paid increasing attention. Researchers have shown that they can significantly reduce the energy costs associated with HVAC systems, i.e., up to 65% of energy when compared with the current strategy used in the building [2, 15, 26, 35, 40, 55]. The knowledge of future energy consumption can bring significant value to building energy management. Predictive control allows one to take advantage of weather forecast and occupancy prediction to reduce energy costs and improve thermal comfort. Moreover, the prediction of temporal energy consumption enables building managers to plan out the energy usage over time, shift energy usage to off-peak periods, exploit the potential in thermal storage, and make more effective energy purchase plans [35].

1.2.1 Thermodynamic models

To keep occupants comfortable, HVAC systems in buildings aim to manage the indoor climate by controlling the temperature and airflow. Hence, it is necessary to have adaptable control systems that could deal with the required parameters regarding the

indoor climatic conditions. There are for this purpose computer models that could simulate indoor climatic processes and control their parameters [39]. Predicting the dynamic air conditioning load in a building accurately is a key for HVAC system design, as well as for adjusting the starting time of cooling to meet start up loads, minimizing the electric on peak demand, optimizing the costs and energy use in cool storage systems [45, 65]. The cooling load in the building is affected by many parameters, which can be grouped into two main categories: the optical and thermal properties of the building and the meteorological data. Because of the complexity of these affecting parameters, it is challenging to consider all of them precisely in the whole building cooling load prediction process. This makes the prediction of the building cooling load and especially its hourly forecast, become a challenging task [36].

The most crucial step of implementing an effective HVAC system control strategy is to create a thermodynamic model, able to predict changes in the building temperature accurately. For control purpose, the model should have a simple structure and be suitable for a wide operational range. However, Huang et al. [25] state that obtaining building models is challenging, as buildings' thermodynamics are non-linear, contain uncertainties, have long time delays and coupled control processes that cannot be treated independently. The success of the modeling process depends on the ability to deal with changing conditions and respond effectively in order to maintain the necessary conditions for the comfort of the occupants, while taking into account the cost [39].

Modeling is typically an iterative process that involves model formulation, parameter estimation, and model validation. Heat dynamic models are usually composed of sub models for the mechanisms behind heat transfer [63]. The most general of these are:

- **Conduction:** Heat transfer through a medium
- **Convection:** Heat transfer between two different media
- **Ventilation:** Heat transfer due to mass transfer, e.g. heated air through an open window
- **Radiation:** Heat transfer between objects that are in optical contact, e.g. solar radiation

By controlling the electrical space heating in buildings intelligently, power consumption can usually be increased or postponed within the hourly timescale. However, commercial load estimation programs are generally time consuming, especially when it comes to identifying the proper thermophysical properties of the construction materials. [28] The problem of identifying a suitable model is both finding a model that is in agreement with the physical reality and finding a model, which has a complexity that is in agreement with the level of information embedded in data, which means that the model should neither be underfitted nor overfitted [4].

Thermal load is mainly associated with external weather variables and internal gains. The weather variables include solar irradiation, ambient temperature, relative humidity, wind speed, etc. The internal gains are associated with people's behavior and

equipment usage inside of buildings and include human occupancy, light schedule, plug load schedule, and infiltration. The literature on building thermal load prediction methodologies can be broadly arranged in three categories: regression analysis, energy simulation, and intelligent computer systems [46].

1.2.2 Optimization

Forecasting of the indoor temperature is necessary for the regulation of energy devices to ensure occupant comfort, as well as for energy optimization [23]. This forecasting constitutes a complex task because it is governed by complex physical and behavioral phenomena. It is affected by a multitude of parameters, which could be classified into three groups: outdoor conditions, building characteristics, and occupants' behavior [20, 44].

To achieve flexibility, different methods are proposed. One approach pursued by Thavlov and Bindner [63] is to use the heat capacity of the thermal mass in buildings to temporarily store excess power production by increasing the electrical heating. The electrical heating is postponed in periods with a lack of production. The researchers presented a model for the prediction of indoor temperature and power consumption from electrical space heating in an office building. The heat dynamic model was build using a grey box approach, i.e., by formulating the model using physical knowledge about heat flow, while the parameters in the model are estimated using collected data and statistics. The physical parameters in the model, e.g., heat capacities and resistances to transfer heat, have been estimated for an actual office building using a maximum likelihood technique. The model has successfully been used in applications for providing power system services in the small distributed power system, SYSLAB.

Indoor temperature forecasting could be carried out using physical or data-driven approaches [50] since it does not have uniform distribution [57]. The physical approach is based on the use of numerical modeling [13], which requires detailed information about a building's characteristics, appliances, and occupant behavior. The data-driven approach is based on the use of collected data for developing relationships (models) between input and output parameters. These relationships could be established by learning from collected data [1]. For the estimation of the flow of energy and the performance of energy systems in buildings, analytic computer codes are often used. The algorithms employed are usually complicated, involving the solution of complex differential equations. These programs typically require large computer power and need a considerable amount of time to give accurate predictions. Therefore, data from building energy systems, being inherently noisy, can be handled using an artificial intelligence approach, i.e., artificial neural networks (ANNs) [63].

1.2.3 Artificial Neural Networks

Many studies have been published on the incorporation of artificial intelligence into the design and operation of HVAC control systems. The techniques employed include fuzzy logic, expert systems, and ANNs. The ANN approach was used to build data-driven

models [14, 24] and predict a number of quality characteristics of buildings, such as internal illuminance, thermal comfort conditions, air quality, and energy consumption, based on different environmental conditions, along with a number of design-relevant building attributes – such as window size and room layout [32, 43, 61]. To predict the behavior of building energy systems, it is required to consider non-linear multivariate inter-relationships. The performance of a building energy system depends on the environmental conditions such as solar radiation and wind speed, the direction, strength, and duration of which are highly variable.

Kalogirou et al. [27, 28] implemented ANN at an early design stage to predict the required heating load of buildings. Input data included the areas and types of windows, walls and floors, roof classification, and room temperature. Lu and Viljanen [38] used the ANN approach to predict air temperature and relative humidity in a test house using indoor and outdoor temperature and humidity. Soleimani-Mohsenishown et al. [59] showed that the operative temperature can be estimated fairly well by using variables, such as the indoor and outdoor temperature, the electrical power use in the room, the wall temperatures, the ventilation flow rates and the time of day. Ekici and Aksoy [3] used a backpropagation three-layered ANN for the prediction of the heating energy requirements of different building samples, benefitting from orientation, insulation thickness, and transparency ratio. Using heating loads calculated with a finite difference approach as ground truth, they reported accuracies of 94.8% to 98.5%. Nassif [42] proposed a model-based optimization process for HVAC systems, capturing very well the system performance and reducing cooling energy consumption by about 11% when compared to the traditional operating strategies applied.

Gonzalez and Zamarreno [17] showed an ANN approach to predict the hourly energy consumption in buildings. The inputs of the network were current and forecasted values of temperature, the current load, the hour, and the day. Saboksayr et al. [53] designed a neural network based decentralized controller to improve the operation of a multizone space heating (MZSH) system. Morel et al. [40] developed a predictive heating controller-algorithm, using ANNs to allow the adaptation of the control model to real conditions and accommodate the non-linearities of buildings.

Egilegor et al. [12] presented a neuro-fuzzy control system. By tuning zone temperature according to the humidity level, they optimized the value of the thermal comfort index PMV (predicted mean vote), which was used as a comfort variable. Ben-Nakhi and Mahmoud [5] demonstrated an HVAC setback scheduling optimization, focusing on energy conservation in air conditioning of public buildings, which are used for only part of every workday. Allowing the temperature to rise inside the building when it is not in use, leads to energy savings, and is known as off-hours thermostat setback. This is achieved by setting back the thermostat temperature after work hours, then resetting it early enough before the start of the workday, such that the desired temperature in the building is restored in time for actual work start. Additionally, there are other examples of ANN design, including PMV-based thermal comfort controller for zone thermal environment control [37], as well as optimization of air conditioning setback scheduling using the outdoor temperature [5]. Garnier et al. [15] also proposed a strategy based on predictive control and managed thermal comfort in non-residential

buildings equipped with HVAC sub-systems without online optimization, saving up to 65% of energy when compared with the current strategy used in the real building. They modeled the non-linear behavior of the PMV index from solar radiation, outdoor temperature, and internal gains and considered the PMV index as a control set-point. Hence, thermal comfort could be maintained in a desired interval which can be adjusted by people working in the given building.

In addition to these specific cases, Attoue et al. [1] developed a simplified model for indoor temperature forecasting, based on the selection of input parameters, analyzing a large set of combinations, including solar radiation outdoor temperature history, outdoor humidity, indoor facade temperature, and humidity. They proposed a methodology, which could be followed for the use of the ANN approach for the indoor temperature forecasting in any type of building.

Karatasou et al. [31] discussed how ANNs, applied to predict energy consumption in buildings, can advantageously be improved, guided by statistical procedures, such as hypothesis testing, information criteria, and cross validation. Their approach consists of identifying all potential relevant input, selecting hidden units for this preliminary set of inputs, through an additive phase, and removing irrelevant inputs and useless hidden units through a subtractive phase. Li et al. [34] compared different network architectures and Jovanovic et al. analyzed the possible application of various network topologies on the same case study. The prediction results achieved with feedforward, radial basis, and adaptive neuro-fuzzy inference system were compared and then combined into an ensemble. The ensemble of ANNs is a very successful technique where the outputs of a set of separately trained neural networks are combined to form one unified prediction [68]. The authors created an ensemble of ANNs for the prediction of heating energy consumption.

1.3 Outline

The thesis is divided into eight sections. After this introduction, Section 2 describes a general thermodynamic model. Later in Section 3 the optimization approach is introduced to the reader. In Section 4 the structure and the problem-solving approach of ANNs are depicted. Section 5 introduces to the reader the real-world data on which the ANN approach was tested. In Section 6 our case study is presented. The next section states our recommendations and proposes future work to be carried out. In the final section, we make our acknowledgments.

2 Modeling thermal load in buildings

The first goal of our study is to identify a proper model, which serves as a basis for the structure of the prediction process. A thermal model usually describes the process of displacement and heat propagation between the temperature of the room and the adjacent room in a building [56]. In order to obtain accurate data with minimal error, we need to analyze the thermodynamic behavior of multi-zoned buildings.

Liang and Du [37] describe the design of a thermal comfort controller for indoor thermal environment regulation and develop a thermal space model. They use computer simulation to show that the controller can maintain the indoor comfort level within the desired range under both heating/cooling modes. The HVAC and thermal space model is shown in Figure 1 and the nomenclature of the used control parameters can be found in Table 1.

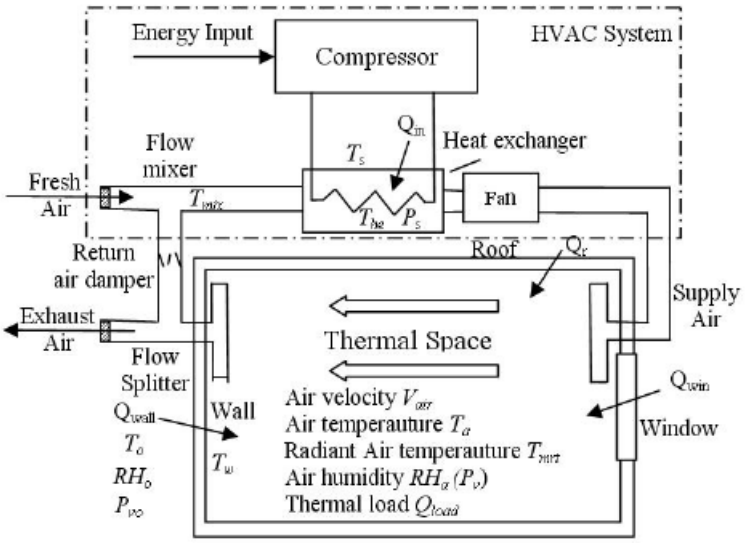

Figure 1: HVAC and thermal space model. Copied from [37].

Then, the following assumptions are made:

1. The wall temperature T_{w} is equal to the mean radiant temperature T_{mrt}.
   $T_{\mathrm{w}} = T_{\mathrm{mrt}}$

2. The indoor air velocity V_{air} is proportional to the supply air flow rate f_{mix}
   $V_{\mathrm{air}} = k \cdot f_{\mathrm{mix}}$

3. The humidity mass ratio W_{a} is proportional to the vapor pressure K_{wv}
   $W_{\mathrm{a}} = l \cdot K_{\mathrm{wv}}$

4. The heat transfer coefficients h are the sum of a natural convective heat transfer coefficient h_{c} and a forced convective $h_{\mathrm{v}} V^{\frac{2}{3}}$
   $h = h_{\mathrm{c}} + h_{\mathrm{v}} V^{\frac{2}{3}}$

5. Time delay is negligible.

Considering both the sensible and latent heat exchange, the mathematical model is derived from the energy conservation and mass balance in different system components:
In the airflow mixer, the return air and the fresh air are mixed perfectly:

$$T_{\mathrm{mix}} = \frac{1}{r} T_{\mathrm{o}} + \frac{r-1}{r} T_{\mathrm{a}} \tag{1}$$

where T_{o} is the outdoor ambient temperature and T_{a} is the temperature in the thermal space, and r is the system-to-fresh-air volumetric flow-rate ratio.

$$W_{\mathrm{mix}} = \frac{1}{r}W_{\mathrm{o}} + \frac{r-1}{r}W_{\mathrm{a}} \tag{2}$$

where W_{mix} is the humidity mass ratio of mixed air, W_{o} is the outdoor humidity mass ratio, W_{a} is the humidity mass ratio in thermal space.

$$p_{\mathrm{mix}} = \frac{1}{r}p_{\mathrm{o}} + \frac{r-1}{r}p_{\mathrm{a}} \tag{3}$$

where p_{mix} is the vapor pressure of water in mixed air, p_{o} is the outdoor vapor pressure of water, and p_{a} is the vapor pressure of water in the thermal space.

Supply air heat exchanger:

$$\rho C_{\mathrm{p}} V_{\mathrm{he}} \dot{T}_{\mathrm{s}} = f_{\mathrm{mix}}\rho C_{\mathrm{p}}(T_{\mathrm{mix}} - T_{\mathrm{s}}) + f_{\mathrm{mix}}\rho H_{\mathrm{fg}} K_{\mathrm{wv}}(p_{\mathrm{mix}} - p_{\mathrm{s}}) + Q_{\mathrm{he}} + lh'_{\mathrm{he}} \min\left[p(T_{\mathrm{he}}) - p_{\mathrm{s}}, 0\right] \tag{4}$$

where ρ is the air density, C_{p} is the constant pressure specific heat of air, V_{he} is the effective heat exchanger volume, T_{s} is the supply air from the heat exchanger, f_{mix} is the mixed air volumetric flow rate, T_{s} is the supply air from the heat exchanger, H_{fg} is the enthalpy of water vapor, p_{s} is the vapor pressure near heat exchanger, Q_{he} is the thermal power from the heat exchanger, l is the Lewis relation, derived as $l = \frac{H_{\mathrm{fg}}K_{\mathrm{wv}}}{C_{\mathrm{p}}}$, h'_{he} is heat transfer coefficient on the surface of heat exchanger, which is derived as $h'_{\mathrm{he}} = h_{\mathrm{he}} V'^{\frac{2}{3}}_{\mathrm{air}} A_{\mathrm{he}}$.

$$K_{\mathrm{wv}}\dot{p_{\mathrm{s}}} = \frac{lh'_{\mathrm{he}}}{H_{\mathrm{fg}}V_{\mathrm{he}}} \min\left[p(T_{\mathrm{he}}) - p_{\mathrm{s}}, 0\right] + K_{\mathrm{wv}}\frac{f_{\mathrm{mix}}}{V_{\mathrm{he}}}(p_{\mathrm{mix}} - p_{\mathrm{s}}) \tag{5}$$

$$h'_{\mathrm{he}} = h_{\mathrm{he}} V'^{\frac{2}{3}}_{\mathrm{air}} A_{\mathrm{he}} \tag{6}$$

$$Q_{\mathrm{he}} = h_{\mathrm{he}} V'^{\frac{2}{3}}_{\mathrm{air}} A_{\mathrm{he}}(T_{\mathrm{he}} - T_{\mathrm{s}}) \tag{7}$$

$$C_{\mathrm{he}}\dot{T_{\mathrm{he}}} = -h_{\mathrm{he}} V'^{\frac{2}{3}}_{\mathrm{air}} A_{\mathrm{he}}(T_{\mathrm{he}} - T_{\mathrm{s}}) - lh'_{\mathrm{he}} \min\left[p(T_{\mathrm{he}}) - p_{\mathrm{s}}, 0\right] + Q_{\mathrm{in}} \tag{8}$$

Governing equations:

$$\dot{W}_{\mathrm{a}} = \frac{f_{\mathrm{mix}}}{V_{\mathrm{a}}}(W_{\mathrm{s}} - W_{\mathrm{a}}) \tag{9}$$

$$\dot{p_{\mathrm{a}}} = \frac{f_{\mathrm{mix}}}{V_{\mathrm{a}}}(p_{\mathrm{s}} - p_{\mathrm{a}}) \tag{10}$$

$$\rho C_{\mathrm{p}} V_{\mathrm{a}} \dot{T}_{\mathrm{a}} = f_{\mathrm{mix}}\rho C_{\mathrm{p}}(T_{\mathrm{s}} - T_{\mathrm{a}}) + \rho f_{\mathrm{mix}} H_{\mathrm{fg}} K_{\mathrm{wv}}(p_{\mathrm{s}} - p_{\mathrm{a}}) + Q_{\mathrm{load}} + Q_{\mathrm{w}} \tag{11}$$

$$Q_{\mathrm{w}} = h_{\mathrm{w}} A_{\mathrm{w}}(T_{\mathrm{w}} - T_{\mathrm{a}}) \tag{12}$$

Equation describing the heat transfer process:

$$C_{\mathrm{w}}\dot{T}_{\mathrm{w}} = -h_{\mathrm{w}}A_{\mathrm{w}}(T_{\mathrm{w}} - T_{\mathrm{a}}) - h_{\mathrm{o}}A_{\mathrm{w}}(T_{\mathrm{w}} - T_{\mathrm{o}}) \tag{13}$$

Based on the above equations, the state-space model for the HVAC and thermal space can be derived as follows:

$$\begin{aligned}
\dot{T}_{\mathrm{s}} =& \frac{f_{\mathrm{mix}}}{V_{\mathrm{he}}}[(\frac{1}{r}T_{\mathrm{o}} + \frac{r-1}{r}T_{\mathrm{a}}) - T_{\mathrm{s}}] + \frac{f_{\mathrm{mix}H_{\mathrm{fg}}K_{\mathrm{wv}}}}{C_{|}pV_{\mathrm{he}}}[(\frac{1}{r}p_{\mathrm{o}} + \frac{r-1}{r}p_{\mathrm{a}}) - p_{\mathrm{s}}] \\
&+ \frac{h_{\mathrm{he}}V_{\mathrm{air}}'^{\frac{2}{3}}A_{\mathrm{he}}}{\rho C_{\mathrm{p}}V_{\mathrm{he}}}l(T_{\mathrm{he}} - T_{\mathrm{s}}) \min\Big[p(T_{\mathrm{he}}) - p_{\mathrm{s}}, 0\Big] \\
\dot{T}_{\mathrm{a}} =& \frac{f_{\mathrm{mix}}}{V_{\mathrm{a}}}(T_{\mathrm{s}} - T_{\mathrm{a}}) + \frac{f_{\mathrm{mix}}H_{\mathrm{fg}}K_{\mathrm{wv}}}{C_{\mathrm{p}}V_{\mathrm{a}}}(p_{\mathrm{s}} - p_{\mathrm{a}}) + \frac{Q_{\mathrm{load}} + h_{\mathrm{c}} + h_{\mathrm{v}}V_{\mathrm{air}}'^{\frac{2}{3}}}{\rho C_{\mathrm{p}}V_{\mathrm{a}}}[A_{\mathrm{w}}(T_{\mathrm{w}} - T_{\mathrm{a}})] \\
\dot{T_{\mathrm{he}}} =& \frac{-h_{\mathrm{he}}V_{\mathrm{air}}'^{\frac{2}{3}}A_{\mathrm{he}}}{C_{\mathrm{he}}}(T_{\mathrm{he}} - T_{\mathrm{s}}) - \frac{lh_{\mathrm{he}}V_{\mathrm{air}}'^{\frac{2}{3}}A_{\mathrm{he}}}{C_{\mathrm{he}}} \min\Big[p(T_{\mathrm{he}}) - p_{\mathrm{s}}, 0\Big] + \frac{Q_{\mathrm{in}}}{C_{\mathrm{he}}} \\
\dot{T}_{\mathrm{w}} =& \frac{-(h_{\mathrm{c}} + h_{\mathrm{v}}V_{\mathrm{air}}^{\frac{2}{3}})A_{\mathrm{w}}}{C_{\mathrm{w}}}(T_{\mathrm{w}} - T_{\mathrm{a}}) - \frac{h_{\mathrm{o}}A_{\mathrm{w}}}{C_{\mathrm{w}}}(T_{\mathrm{w}} - T_{\mathrm{o}}) \\
\dot{p_{\mathrm{s}}} =& \frac{lh_{\mathrm{he}}V_{\mathrm{air}}'^{\frac{2}{3}}A_{\mathrm{he}}}{H_{\mathrm{fg}}V_{\mathrm{he}}K_{\mathrm{wv}}} \min\Big[p(T_{\mathrm{he}}) - p_{\mathrm{s}}, 0\Big] + \frac{f_{\mathrm{mix}}}{V_{\mathrm{he}}}[(\frac{1}{r}p_{\mathrm{o}} + \frac{r-1}{r}p_{\mathrm{a}}) - p_{\mathrm{s}}] \\
\dot{p_{\mathrm{a}}} =& \frac{f_{\mathrm{mix}}}{V_{\mathrm{a}}}(p_{\mathrm{s}} - p_{\mathrm{a}})
\end{aligned}$$

Bacher and Madsen [4] suggest a procedure for the identification of suitable models for the heat dynamics of a building. The models can be used for different purposes, e.g., control of the indoor climate, forecasting of energy consumption, and for the accurate description of the energy performance of the building. All of the above are from significance for the following work. Based on prior physical knowledge, grey-box and data-driven models can be applied. Furthermore, taking into account time series of weather forecast data and measured indoor data, a suitable model with increased complexity is formulated.

In order to obtain an accurate prognosis of the building heating load, a thermodynamic zone model has to be designed concerning specific input and output variables. The selection of the former is of great significance for the complexity of the predictive model. Additional redundant input variables could unnecessarily increase the obscurity during the development and execution of the model. However, the lack of information for a building's construction and materials could also restrict the amount of the selected input variables. Hence, a reasonable number of inputs must be selected, choosing the most relevant and giving up the uninformative ones to retain the accuracy and efficiency of the model.

Taking into consideration these observations, the input variables introduced to the model should be based on the control parameters, the weather forecast, the measured indoor conditions, the building geometry and construction materials, the internal loads, and the HVAC configurations.

2.1 Control data

There are three possible states of the HVAC system, i.e., heating, cooling, and ventilation. Every state is measured in continuous values (%). Both heating and cooling are mutually exclusive. This is ensured by the climate profile, which will be explained later in the next section.

The ratio of the HVAC states depends on the temperature of the water, which flows from a heating source into a pump. Exactly the inlet temperature is the parameter we have control on. Adjusting its value has a direct impact on the HVAC mode, consequently on the measured indoor temperature and therefore, on the thermal comfort of the occupants. To enhance the reader's understanding of the connection between the inlet temperature and the rate of the HVAC mode, we use the HVAC heating mode. For example, the smallest possible value of the inlet temperature when in heating mode is 20°C, which corresponds to 0% heating. Respectively, when the largest value is 80°C, the corresponding ratio is 100% heating.

The control parameters, i.e., the inlet temperature, and the possible HVAC states are shown in Table 1.

Variable	Unit
Inlet temperature (Heating mode)	°C
Inlet temperature (Cooling mode)	°C
Inlet temperature (Ventilation mode)	°C

Table 1: Input variables describing the control data

2.2 Future weather data and measured indoor data

The second and the third group of input variables, i.e., the future weather data and the measured indoor data are represented in Table 2. Only the former must be forecasted as the latter is directly related to the building's construction and internal heating control. Consequently, it can be easily derived from the given input. In some papers, it is stated that since the model is indexed with respect to the ambient temperature, there is no need to refine the model further to include the type of day or the season of the year [66]. This could be investigated through reasonable examples and testings of various combinations of input data. In further research, the influence of the use of different sets of input parameters on the mean square error (MSE) and on the coefficient of correlation (R) between the input and the output variables has been examined [1]. The indoor temperature can be forecasted with good precision even if

only outdoor temperature and outdoor temperature history are taken into account. Predictions of the facade temperature are also sufficient in their accuracy within the time period of two hours. However, indoor activities are not included in the input variables, i.e., meetings, use of energy-consuming devices, opening doors and windows. If these activities affect the temperature in the building, they should be monitored and included in the input data of the heating model.

Future weather data		**Measured indoor data**	
Variable	**Unit**	**Variable**	**Unit**
Outdoor temperature	°C	Indoor zone temperature	°C
Relative humidity	%	Relative humidity	%
Global radiation	W/m^2		
Wind speed	m/s		
Sky clearness index	K_T		

Table 2: Input variables describing the future weather and the measured indoor data

2.3 Construction material data

The next group of input variables is the construction material data and can be seen in Table 3. Some parameters such as window, wall and roof type are described through U-values. In these cases, exact U-values were not used. Instead, class numbers n $\in$ [1,4] corresponding to each type of construction have been assigned. This methodology is applied for simplification of the calculations. The wall, window, and roof type are additionally described in Table 4, Table 5 and Table 6 accordingly. The construction material data can be held constant in our future computation since we only consider one prototype zone. In a later phase, the same computation can be made for an entire multi-zoned building with only a few modifications.

Variable	**Unit**
Volume	m^3
Floor area	m^2
Wall area	m^2
Glazing area	m^2
Window Type	-
Wall Type	-
Roof Type	-

Table 3: Input variables describing the construction material data

Class	Description	U-value (W/m^2K)
1	Single glazing	6.4
2	Double glazing	3.2

Table 4: Window types under examination

Class	Description	U-value (W/m^2K)
1	Single wall without insulation	2.0
2	Double wall without insulation	1.5
3	Double wall with 25mm polystyrene	0.83
4	Double wall with 50mm polystyrene	0.53

Table 5: Wall types under examination

Class	Description	U-value (W/m^2K)
1	Non insulated roof	2.3
2	Roof with 25mm polystyrene	0.88
4	Roof with 50mm polystyrene	0.55

Table 6: Roof types under examination

2.4 Internal loads

The internal load of a building can be calculated correctly if all of the sources of internal heat gains are taken into account. The main sources of internal loads are occupants, lighting devices, and electrical equipment.

The determination of the heating/cooling load of a building depends on heat gains in the building energy model [8]. The dynamic relationship between building occupancy and energy consumption is of significant importance. Therefore, since we observe office buildings, the working hours of the building should also be included in the input parameters. On the other hand, the produced CO_2 levels from the occupants are a further factor, which should be taken into account as they have an impact on the humidity of the zone.

Another aspect that should also be considered is the clothing of the occupants, which influences the perception of the temperature in the building and it depends on the season of the year. For example, in winter, people are wearing more insulating clothing than in summer and consequently, the indoor temperature would be felt as higher in the winter although it is held constant, no matter the season. An additional detail is the metabolic rate - the total energy produced by the occupants. Feasibly, it is assumed that the higher the activity level is, the higher the metabolic rate and thus, the produced heat inside the building.

Lighting devices are also a source of convectional and radiant heat gains in the building. There are some possible ways to decrease the cooling load of the building by changing the lighting type, e.g., the LED lighting has the potential to provide energy savings.

Sample input variables are presented in Table 7.

Variable	Unit
Heat input	W
Thermal power from the wall	W
Thermal power from the heat exchanger	W
Indoor wall temperature	°C
Occupancy	people
Clothing insulation	m^2K/W
CO_2 level	%
Working hours	h
Metabolic rate	W/m^2
Lightning	W/m^2

Table 7: Input variables describing the internal gains

2.5 HVAC data

Finally, the parameters of the HVAC equipment are also fed into the model, as they represent a fundamental part of the calculation process. They are considered in the thermodynamic zone model as well and are those mentioned in the equations describing the thermal comfort model at the beginning of this section. The variables describing the HVAC characteristics are listed in Table 8.

Variable	Unit
Air velocity	m/s
Air flow rate	m^3/h
Water mass flow	kg/s
Partial water vapor pressure	Pa
Mixed air volumetric flow rate	m^3
Vapor pressure	Pa
Radiator dimensions	m
Floor heating	W
Humidity mass ratio	%

Table 8: Input variables describing the HVAC data under examination

2.6 Future indoor data

Output variables are the thermal loads, which will be forecasted from the model for the next hours or day, i.e., depending on the desired prediction. This future indoor data can be used for further analysis considering the goal of the optimization task, e.g., thermal comfort and cost minimization.

All of the represented input and output data and the concept of the thermodynamic model are shown in Figure 2. However, not all models make use of all of the input variables, since an accurate calculation can be made using only a subset of the initial set.

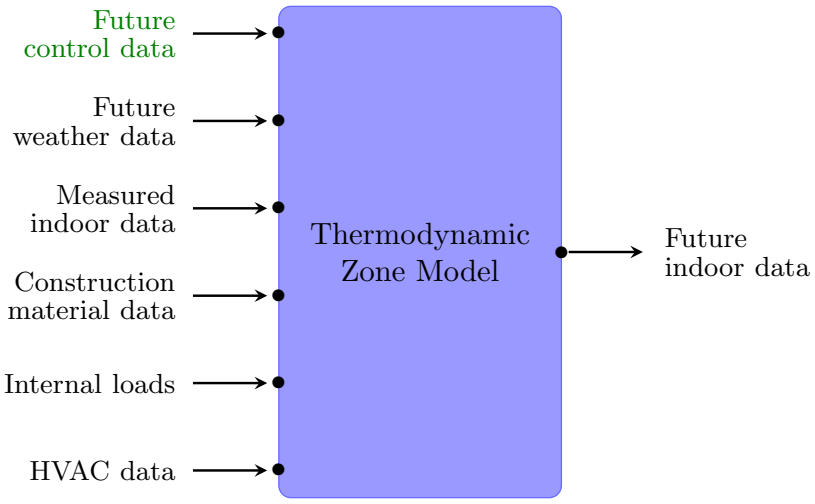

Figure 2: Construction of the thermodynamic zone model - the input consists of future control data, future weather data, measured indoor data, construction material data, internal loads, and HVAC data. The expected output is the future indoor data which is needed for the optimization task.

3 Optimization of climate in buildings

Since one time feeding the thermodynamic model with the input variables from the categories, described in the previous section, is not sufficient to compute the minimum costs of the HVAC system to maintain the temperature in the desired boundaries, we upgrade the computational process with an optimizer. The optimizer performs this task several times in order to estimate the optimal settings of the HVAC system and consequently, the required control parameters, which define the indoor climate. The process of optimizing the control data consists of several computation rounds, i.e., iterations, performed by a smart iterator. Initially, the described input data is fed into the thermodynamic model and a first prediction for the future indoor data is made. Then the future indoor data obtained from the thermodynamic zone model is compared to a climate profile given as an input into the optimizer.

3.1 Climate profile

The climate profile sets an upper and a lower boundary for the indoor temperature. The profile is updated every 15 minutes with a prediction for the next two days. The

indoor temperature must be preserved between these boundaries in order to meet the required environment expectations. If the predicted temperature does not fit into the boundaries, then the procedure of optimizing the control data is repeated until a proper configuration is found. An example of climate profile can be seen in Figure 3.

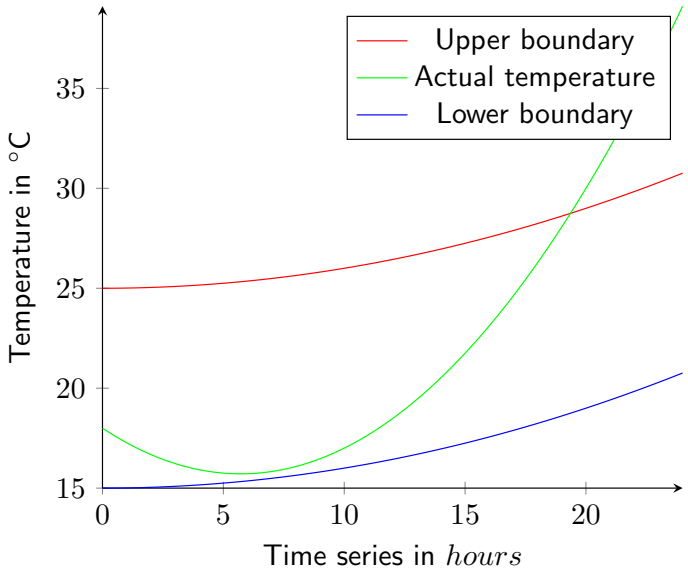

Figure 3: Climate profile example: the measured temperature in the room must not fall below the lower boundary or exceed the upper boundary

The input variables regarding the climate profile are outlined in Table 9.

Variable	Unit
Climate profile upper limit	°C
Climate profile lower limit	°C

Table 9: Input variables describing the climate profile

3.2 Economic energy model

If the predicted future indoor data lies in the climate profile boundaries and hence the temperature is maintained in compliance with the climate profile, the optimizer proceeds with the estimation of the energy costs and then obtains the minimum costs, according to an economic energy model, shown in Figure 4. The economic energy model takes a vector of future control data and computes the minimum energy costs, needed to reach the values of the control parameters which determine the indoor temperature. The minimum cost is estimated considering the future energy profile, which is known in advance and is also fed into the model.

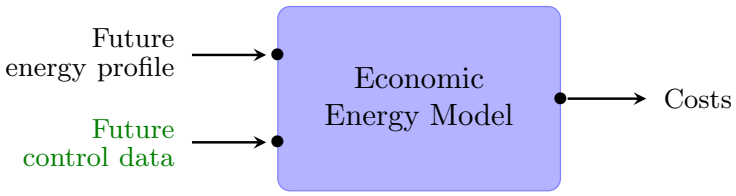

Figure 4: Construction of the economic energy model which computes the minimum energy costs according to the future indoor data and the future energy profile

3.3 Optimization

Lastly, when both the future indoor temperature and the minimum costs are estimated, the smart iterator computes the future control data and gives it as an output. If both previous conditions, i.e., the climate profile compliance and the minimality of the cots are not preserved, then the future control data is once again fed into both the thermodynamic zone model and the economic energy model, closing the circle and making another attempt to compute the optimal solution. Otherwise, if both conditions are kept true, the future control data is outputted as a result of the optimization task. The structure of the optimizer is shown in Figure 5.

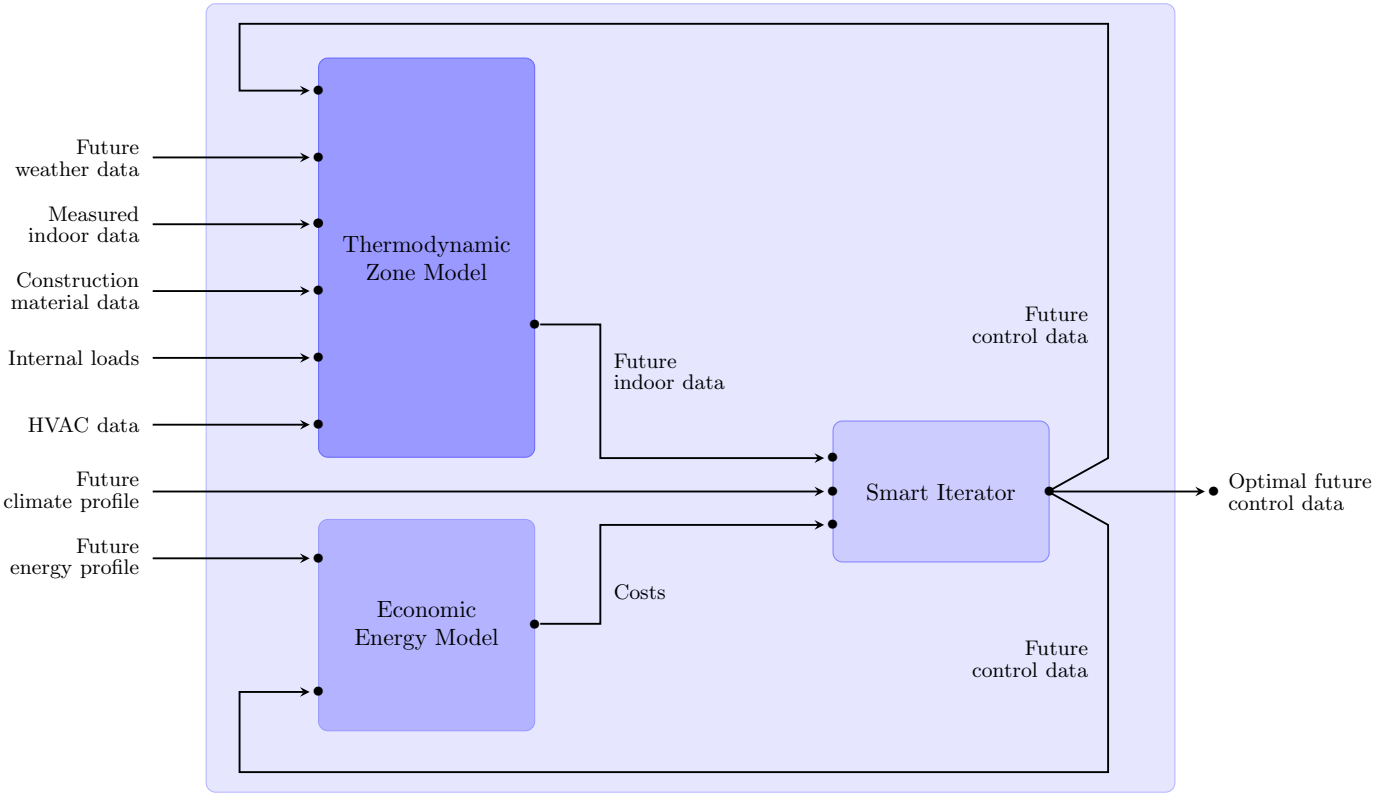

Figure 5: Construction of the optimizer including three substructures: a thermodynamic zone model, an economic energy model and a smart iterator

However, this optimization process may turn out to be very slow as the number of iterations can increase drastically in case of high energy prices and little difference between both boundaries. Therefore, we need to develop this approach further in order to accomplish our goals of minimizing the computation time of the future control data.

4 Artifical Neural Networks

Artificial neural networks (ANNs) are one of the main tools used in machine learning and have been applied successfully in various fields of mathematics, engineering, medicine, neurology, psychology, economics, meteorology, etc. ANN models may be used as an alternative method in engineering analysis and predictions [29]. Instead of complex rules and mathematical routines, ANNs are able to learn the key information patterns using multidimensional data. Based on these patterns, ANNs are capable of predicting other combinations of unseen input. Moreover, ANNs are fault-tolerant, robust, and noise immune [52].

ANNs are essentially inspired by the biological neural superstructure of the powerful cognitive and sensory functions of the human brain, e.g., the eyes or the nerve endings in the hand, and are intended to replicate the way that humans learn and react to the external environment, for example, light, touch, or heat. They operate like a 'black box' model, requiring no detailed information about the inner system. Instead, they 'learn' the relationship between the input parameters and the variables by studying previously recorded data. ANNs usually perform successfully where other classical methods of analysis do not, e.g., non-linear problems such as pattern recognition. Among the main advantages of using ANNs is their ability to handle large and complex systems with many interrelated parameters, e.g., coupled control processes [30].

4.1 Configurations and ANN types

ANNs are composed of multiple nodes, which imitate biological neurons of the human brain and are organized in different layers. The minimum architecture includes one input layer with one neuron corresponding to each input parameter and one output layer with one neuron for each output parameter. Every other layer in between is called 'hidden layer' and is connected with the following by links associated with adaptable synaptic weights through which the nodes interact with each other. Each neuron in the network is able to receive input signals, to process them, and to send an output signal. Every neuron is connected at least with one neuron, and each connection is evaluated by a real number, called a weight coefficient, that reflects the degree of importance of the given connection in the neural network [62]. The last hidden layer is connected with the output layer. The basic structure of an ANN is represented in Fig. 6.

A key point in using ANN to describe nonlinear dynamics is to define the right topology of the network. There are many types of ANN, but the basic principles are very similar. The general rule states that the higher the number of hidden layers, the greater the ability to represent highly nonlinear hypotheses. However, at the same

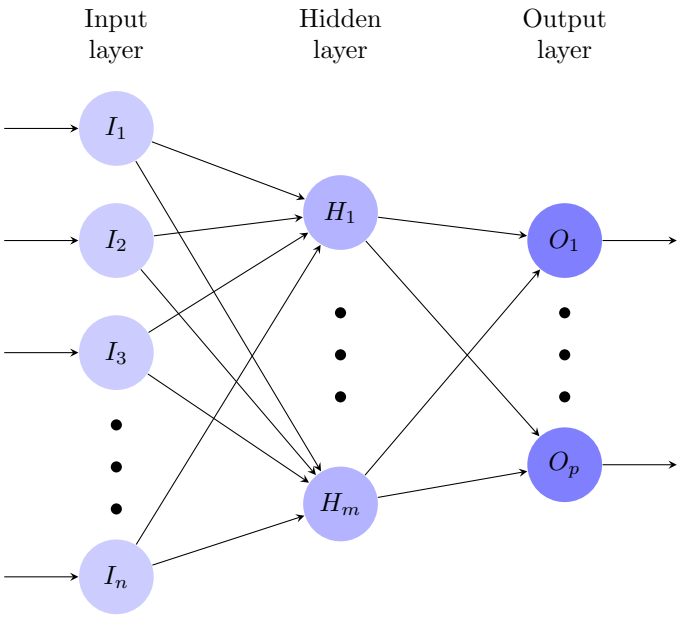

Figure 6: Structure of an ANN with n nodes in the input layer, m nodes in the hidden layer and p nodes in the output layer

time, too many hidden layers can result in 'overtraining' or 'overfitting', i.e., lack of generalization, and lead to large verification errors. In contrast, too few layers can result in 'underfitting', i.e., inability to adequately detect the signals in a complicated data set and consequently, also leading to verification errors. The number of neurons in the hidden layers is approximately the average of the inputs and outputs, but it also depends on the number of training cases [64].

4.1.1 Multi layer feed-forward neural netwoks

Multi-layer feed-forward neural networks (MLFNN) are the most popular neural networks and are the foundation of most deep learning models. MLFNNs are mainly used for supervised machine learning tasks where the target function is already known, i.e., the result to be achieved. This approach originates from the 50s and is applied to a wide variety of chemistry-related problems, computer vision, natural language processing (NLP), time series prediction, pattern recognition, financial prediction, autonomous driving, etc. The primary goal of an MLFNN is to approximate some function f^*, e.g., a regression function $y = f(x)$. The function maps an input x to a value y. An MLFNN defines a mapping $y = f(x;\theta)$ and learns the value of the parameters θ that result in the best function approximation. The flow of information takes place in the forward direction, as x is used to calculate some intermediate functions in the hidden layers, which in turn are used to calculate y. If feedback from the last hidden layer to the first hidden layer is added, it would represent a recurrent neural network.

4.1.2 Recurrent neural networks

Recurrent neural networks (RNNs) contain cyclic connections that make them a more powerful tool to model complex time-varying signals, e.g., speech, than MLFNNs. RNNs have demonstrated great success in sequence labeling and prediction tasks such as handwriting recognition and language modeling [54]. Unlike MLFNNs, RNNs can use their internal state, i.e., memory, to process sequences of inputs. There are many variations, e.g., passing the state to input nodes and variable delay. This type of NNs is mainly used when context is important, e.g., decisions from past or future iterations can influence current ones. For example in texts - a word can be analyzed only in context of previous words or sentences. The information persists using loops, which pass it from one step of the network to the next and so the network 'remembers' its importance in the following states. However, a problem occurs when the gap between the relevant information and the point where it is needed has become very large. The network 'forgets' slowly the information received and analyzed at far earlier stages and so the computation of the final result is not accurate and precise enough. This problem is also known as the vanishing gradient problem [22].

The solution is found using long short term memory networks (LSTMNs), which are type of RNNs with the ability to learn long-term dependencies. Every state of the NN is called a cell. In every cell the information forwarded to the next one is processed and transformed into a vector, persisting only parts of the input, e.g., filtering the most valuable information. This process is carefully regulated by structures called gates. They optionally let information through imitating a biological cell membrane's selective permeability. Gates are representing neural network layers, which use activation functions, e.g., *sigmoid* and *tanh*. The activation functions filter, update with the new input knowledge, and cut only the relevant parts to be outputted. This eliminates the vanishing gradient problem since the model is not forgetting the new input every single time but instead is keeping it and passing it down to the next time steps of the network. Thus, the potential long-distance dependencies are captured [7].

The gated recurrent unit (GRU) is a new approach recently proposed on sequence modeling [7]. Instead of using three gates, i.e., forget, input, and output gate, as described in the LSTMN's method, GRU uses only two - reset and update gates. This means that the model neither does include separate memory cells nor has any mechanism to control the degree to which its state is exposed, but exposes its full content each time. Another difference between both models is in the location of the input gate in the LSTMN and the location of the reset gate in the GRU. The LSTM unit has control over the amount of new memory content added to the memory cell independently from the forget gate. The GRU, on the other hand, controls the information flow from the previous activation when computing the new candidate activation, but does not independently control the amount of the candidate activation being added. Based on these comparisons, both structures show their advantages for different computation tasks. Chung et al. [7] cannot conclude which of the either is better, so they plan their future work conducting more thorough experiments.

4.1.3 Sequence to sequence neural networks

Another issue when working with sequences is their variable length, e.g., in speech recognition and machine translation problems. Using a multilayered RNN to map the input sequence to a vector of fixed dimensionality, and then another RNN to decode the target sequence from the vector eliminates the significant limitation of expressing sequences whose lengths are not known a-priori. This model is called sequence to sequence (Seq2Seq) and aims to map sequences of different lengths to each other. The model contains three submodels, i.e., encoder, encoder vector, and decoder. The encoder is formed as a stack of several recurrent units (LSTM or GRU cells). Each one accepts a single element of the input sequence. Then the recurrent unit collects information for that element and propagates it forward. The final hidden state produced from the encoder part of the model is the encoder vector. It plays the role of an initial hidden state of the decoder which is also formed as a stack of several recurrent units. Each one accepts a hidden state from the previous unit and produces an output as well as its hidden state at every time step.

4.1.4 Deep neural networks

Another type of NNs is deep neural networks (DNNs). DNNs have greater capabilities for image pattern recognition and are widely used in computer vision algorithms where the basis of most is to classify an image into known labels. A convolutional neural network (CNN) is a class of DNN that is most commonly applied to analyzing visual imagery and for text classification in NLP. ANN takes a very long time to be trained in case of processing images with fully connected hidden layers. Due to that fact, CNN was used first to reduce the size of images using convolutional and pooling layers and then feed the reduced data to fully connected layers. The convolutional and the pooling layer are used as a filter in order to downsize the input matrix by only selecting the highest value pixel present in the filter. This reduces the amount of computation required for training significantly [10].

4.2 Mathematical definition

The key benefit of ANNs is the fact that they are able to use some a-priori unknown information hidden in data. However, they do not have the ability to extract it. The process of 'capturing' the unknown information is called 'learning' or 'training' [62]. There exist two main types of training processes - supervised and unsupervised. When the desired output is not known, the system is provided with a group of facts, i.e., patterns, and let to settle down itself to a stable state in some number of iterations. This type of training is called unsupervised. Alternatively, in the supervised training, the ANN knows the desired output and adjusts the weight coefficients so that the calculated and desired outputs are as close as possible.

4.2.1 Activation functions

For training the network a training data is used, i.e., a group of matched input and output patterns. The outputs are the dependent variables that the network produces for the corresponding input. Starting from an initially randomized weighted network, the nodes on the input layer take data and perform simple operations on it, e.g., summation and activation functions. Among the most commonly used activation functions is the sigmoid function [6, 25, 64], having the following form:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{14}$$

and range [0,1]. The advantage of the sigmoid function is that it is a differentiable function. Differentiation is a necessary part of the learning process when applying the most widely used training algorithm - backpropagation [64]. It uses a gradient descent technique to minimize the cost function. The details of the backpropagation algorithm are rather complex but do not contribute significantly to the understanding of this work. Therefore, for details, readers can refer to [19]. The sigmoid function is monotonic, but it's derivative is not, which might give rise to a vanishing gradient problem. Consequently, it can result in the network refusing to learn further, causing the NN to get stuck at the training time or being too slow to reach an accurate prediction.

Another possibility is to use a hyperbolic tangent function which is also sigmoidal:

$$tanh(x) = 2\, sigmoid(2x) - 1 \tag{15}$$

The range of the function is [-1,1]. The advantage is that the negative inputs are mapped strongly negative and the zero inputs near zero in the *tanh* graph. This function is also differentiable and monotonic, but then again its derivative is not monotonic. Both logistic *sigmoid* activation and *tanh* functions are used in MLFNNs.

The most used in the world right now is the ReLU (Rectified Linear Unit) activation function:

$$R(x) = max(0, x) \tag{16}$$

It's range is [0, ∞). The function and its derivative are monotonic. The constant gradient of ReLU results in faster learning, controversial to the gradient of sigmoids which are shrinking as the absolute value of x increases. Due to its simplicity and effectiveness, ReLU has become the default activation function used across the deep learning community [49]. The main disadvantage of this function is that all negative values are mapped to zero, also known as the Dying ReLU problem. For activations in that region of ReLu, the gradient is 0 because of which the weights do not get adjusted. Consequently, those neurons which go into that state stop responding to variations in error and make a substantial part of the network passive. The solution here is to draw a slightly inclined line rather than a horizontal line for inputs less than 0. This variation of the ReLU is the Leaky ReLU activation function:

$$LR(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise} \end{cases} \tag{17}$$

4.2.2 Feed-forward technique

When the result of the function is computed, it is passed to the neurons on the next layer, then multiplied with the weight values of the next layer and summed up. The data is propagated through the network to provide an estimate of the output value. *Equation 15* describes a minimum architecture, e.g., ANN including only the input and output layer with no hidden layers:

$$Y_j = f(\sum_{i=1}^{n} W_{ij} X_i) \qquad \text{for } j \in [1, m] \tag{18}$$

where X_i stands for the input variables and Y_j denotes the output values. W_{ij} identifies the weights between the input layer and the output layer.
Analogously, *Equation 16* describes a two-layered ANN:

$$Y_k = f(\sum_{j=1}^{m} W_{jk} \times f(\sum_{i=1}^{n} W_{ij} X_i)) \qquad \text{for } k \in [1, p] \tag{19}$$

The output is then compared to the training pattern, i.e., the correct or desired output, i.e., the label. The ANN performances could be evaluated using the mean square error (MSE) based on the difference between the input X_i and the output Y_i:

$$MSE = \sum_{i=1}^{n} (\frac{(Y_i - X_i)^2}{N}) \tag{20}$$

If there is a difference, the connection weights are altered in order to decrease the error. If all the input patterns are already run through and the error is still greater than the maximum desired tolerance, the procedure is repeated until the required tolerance is reached. After the termination of the training phase, the weights are held constant. Consequently, the ANN ignores superfluous data that is of minimal significance and concentrate instead on the more important and relevant inputs. Hence, knowledge is usually stored as a set in the synaptic weights [30]. Further, the trained network is capable of making decisions, identifying patterns, or defining associations in new input data sets not used to train it.

4.2.3 ANN model

Leveraging the benefits of ANNs, we once again upgrade our solution aiming to accelerate the optimization process. Applying this powerful tool leads our research to a new level of computation speed. Our final model can be seen in Fig. 7.

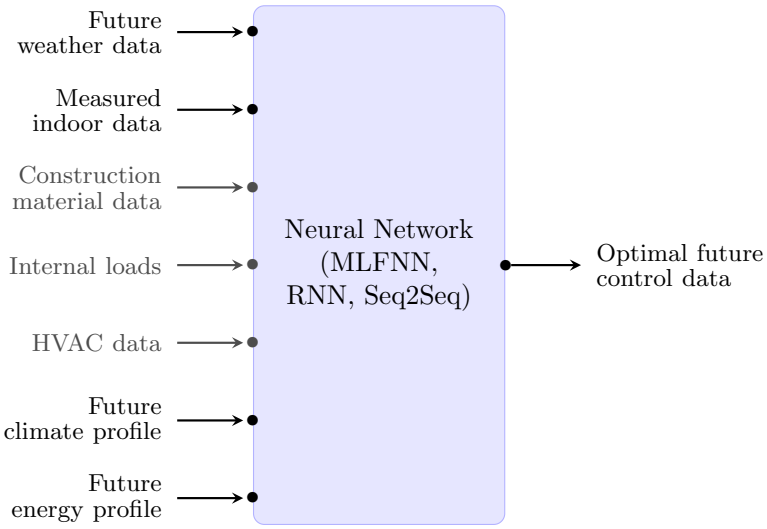

Figure 7: Construction of the model including an ANN with inputs: future weather data, measured indoor data, future climate profile, future energy profile, and output - optimal future control data; the construction material data, the internal loads, and the HVAC data can be held constant as they stay the same when observing only one thermal zone, otherwise when a multi-zone building is a part of the optimization process, these data sets are treated as input variables

5 Data

To verify the effectiveness of the ANN models, simulations were conducted on real-world data. The dataset used in this research is provided by the facility management company MeteoViva. MeteoViva controls the indoor climate in buildings in a most efficient and cost-effective manner. Their solution results in reduced costs, a stable indoor climate, and lower CO_2 emissions.

5.1 Data description

The HVAC system of the building chosen for the research purposes has only a heating mode. In such a way the complexity of the task is decreased, as we examine the compatibility of this particular type of optimization task with the neural network approach. The data describes the weather conditions and the behavior of the HVAC system of one thermal zone. This method can later easily be applied to a multiple thermal zone building. Hence, we also propose this technique for more sophisticated cases. However, our example is a real-world task given by a client of MeteoViva. The data is collected in the last four years (2016-2019), 24 hours a day, seven days a week in time series of 15 minutes.

5.2 Parameter selection

Parameter selection is performed to eliminate parameters of less importance to the model of the neural network to increase comprehensibility, simplicity, and the accuracy of the resulting model. Table 10 includes seven parameters that were selected as the candidates for building the models to be discussed later in the paper.

Variable	Unit
Future weather data	
Outdoor temperature	°C
Global radiation	W/m^2
Wind speed	m/s
Measured indoor data	
Indoor zone temperature	°C
Future climate profile	
Climate profile upper limit	°C
Climate profile lower limit	°C
Future energy profile	
Energy price	$cents/kWh$

Table 10: Input variables of the NN

In our test case, the energy price was considered constant. Thus, we omitted this variable later in the simulations. As mentioned above, the HVAC system of the reviewed building has only one mode, i.e., heating. Therefore, only the inlet temperature is considered as an output value of the neural network instead of the combination of both, i.e., inlet temperature and the mode to be maintained, i.e., heating, ventilation or cooling (see Table 1). The output is depicted in Table 11.

Variable	Unit
Future control data	
Inlet temperature	°C

Table 11: Output variables of the NN

5.3 Data processing

Real-world data sets can be messy, incomplete, and in a variety of formats. MeteoViva has provided our study with a significantly big data set which serves its purpose, i.e., to train and test our neural network on it. For every month of the year, there is a separate *csv* file, containing more variables than needed for the conduction of this survey. Additionally, the used separator between the values was ';', which makes it difficult to work with the data set.

In order to prepare the data for our case study, we set the delimiter of the values to ',', so we could have a better overview of the data. The data is cleaned out of

cells containing *NaN* values which causes errors when computations are carried out. Moreover, only the relevant columns of parameters are extracted. Then the *csv* files are concatenated into one file. Further, we normalize the data, changing its values to a common scale, without distorting differences in their ranges. This step is of great importance since it is required in case the dataset features vary significantly, e.g., in our case the value of global radiation fluctuates in the range [0, 1000] and wind speed is less than 18. Thus, we prevent oscillation of the ANN, making training less sensitive to the scale of features, preserving consistency for comparing results across models, and improving the convergence rate of gradient descent. These tasks were performed using *Python* (version 3.7.3) and *Jupyter Notebook* (version 6.0.0).

5.4 Data visualization

We also visualize the data which will help us to gain insight into it. For the visualization of the main features we have chosen the time period 8-10 May 2019 (see Figure 8). In May the outdoor temperatures climb every day to a certain peak and then go down in the night hours of the day. This observation gives us the opportunity to point out the relations between the features. It is clear to see that the lower limit of the climate profile fluctuates between 18°C and 22°C and the upper limit is held constant at 34°C. The measured temperature in the thermal zone is maintained between the limits of the climate profile and whenever it draws near the lower one, the inlet temperature is increased in order to preserve the indoor temperature acceptable, i.e., in the boundaries required by the customer. The outdoor temperature has an influence on the indoor temperature as well. Its highest points correspond to the highest points of the indoor temperature.

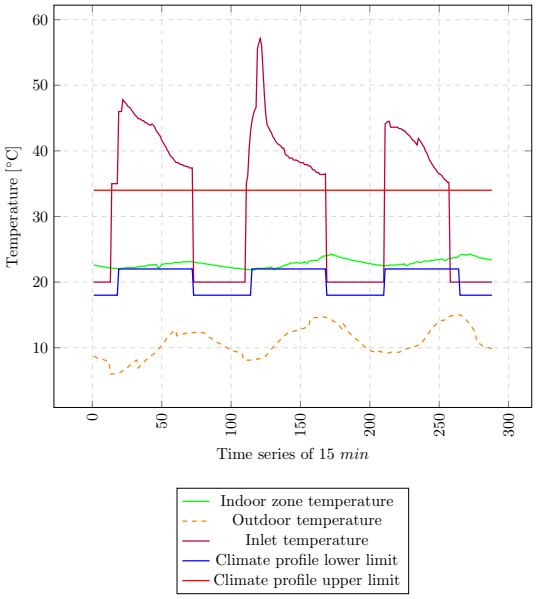

Figure 8: Data visualization for 8-10 May 2019; the relation between the indoor zone, outdoor and inlet temperature as well as the lower and the upper limit

6 Case study

6.1 Case description

This case study investigates the most appropriate ANN type, structure and configurations to be used in finding the optimal control parameters which serve as initialization variables for the optimization task of the HVAC system. In the next sections our approach for the configurations testing, the simulations of different ANN architectures and configurations, and the obtained results are going to be discussed in greater detail.

6.2 Configurations

ANNs have many internal parameters that control the structure of the network. In the following are listed the main hyperparameters:

- Type of ANN
- Number of layers
- Number of nodes
- Activation function
- Loss function
- Optimization algorithm
- Number of epochs
- Patience

When configuring the network's topology, we must specify the values of these parameters as they play an important role in training the model efficiently and effectively. They have a significant influence on the learning process and on the output.

6.2.1 Type

Firstly, the ANN type must be determined. We chose the Multi-Layer Feed-forward Neural Network (MLFNN) and the Recurrent Neural Network (RNN) for the conduction of this research. The reason behind the decision of choosing these types of ANNs is that the MLFNN is the simplest and most straightforward ANN structure and as such, provides a great base case for the study. However, all inputs are independent of each other and the information about previous computations is not preserved for future predictions. On the other side, the RNN takes a series of inputs with no predetermined limit on size. This feature enables the opportunity to investigate serialization problems with unlimited length of sequences as an input. RNNs are considered more complex and hard to train.

6.2.2 Layers

Secondly, we must choose the number of layers which is one of the most significant hyperparameters that control the topology of the network. The most reliable way of finding the optimal number for our specific predictive problem is through thorough and systematic testing of different combinations. The universal approximation theorem states that a feedforward network with a linear output layer and at least one hidden layer with any activation function can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units [18]. Further theoretical finding reveals that although a single hidden layer is optimal for some functions, there are others for which a single-hidden-layer-solution is very inefficient compared to solutions with more layers [58]. Therefore, we begin our testing with an MLFNN with just two hidden layers but then extend the investigation two steps further by testing architectures with three and four hidden layers.

6.2.3 Nodes

The number of nodes is another hyperparameter to be estimated through testing. However, the number of nodes in the input and output layer is fixed. The number of inputs corresponds with the number of nodes in the input layer of the ANN and the number of outputs with the number of nodes in the output layer. Depending on the different cases we examine, i.e., with and without index data for the time, we have six or seven inputs and therefore, six or seven nodes in the input layer (see section 6.3 Simulations for more information about the three types of datasets). Since we analyze a regression problem, i.e., the output is one predicted integer value and thus, there is only one node on the last layer. In general, there is no analytical approach to calculate the number of nodes to use. Given the fact that every specific real-world problem has its unique nature, the predictive modeling problem must be addressed with caution, systematic robust test harness, and controlled experiments. Therefore, we choose to examine a different number of nodes and combinations of them per layer, only following the dependency that every subsequent has equal or less nodes than the previous, as it helps promote generalizability. There exist many thumb rules used as a good starting point for different problems in the machine learning community. Along the most commonly used is the following upper bound for the number of hidden neurons given by:

$$N_h = \frac{N_s}{\alpha.(N_i + N_o)} \tag{21}$$

N_h:= number of hidden neurons
N_i:= number of input neurons
N_o:= number of output neurons
N_s:= number of samples in training data set
α:= an arbitrary scaling factor, usually $\alpha \in [2, 10]$.

The number of input neurons is six or seven (see section 6.3 Simulations for more information about the three types of datasets), the number of output nodes is one and the training dataset consists of 87732 samples (90% of all samples). Therefore, following this thumb rule, we choose our starting point, using a different number of nodes and combinations of them among the hidden layers. The possible configurations of layers consist of the number of nodes in the set {50, 100, 150, 200, 300, 400, 600}. Consequently, the minimum number of nodes we use is 50 and the maximum is 600.

6.2.4 Activation function

As already discussed in section 4.2.1, there are multiple options for activation function. Thus, we test all of the described, excluding the ReLU activation function, since we choose to use its variation, i.e., the Leaky ReLU, in order to avoid the Dying ReLU problem.

6.2.5 Loss function

To evaluate a candidate solution, objective functions are used. In our research, we seek to minimize the error. As such, the objective function is often called an error or a loss function. The value calculated by the loss function is referred to as loss. The loss function reduces all the various good and bad aspects of a possibly complex system down to a single number, which allows candidate solutions to be ranked and compared [58]. As already mentioned, we observe a regression problem. The final layer of the neural network has one neuron and the value it returns is a continuous numerical value. Consequently, to evaluate the accuracy of the prediction, it is compared with the actual value, which is also a continuous number. We choose to use a linear function for the output layer, as recommended in the machine learning community. The spectrum of the most widely used regression loss functions consists of the Mean Squared Error (MSE), the Mean Squared Logarithmic Error (MSLE), and the Mean Absolute Error (MAE).

The MSE is calculated as the average of the squared differences between the predicted and the actual values. The output is always positive, regardless of the sign of the predicted and actual values. The perfect value is 0. The larger the error is, the larger the result of the MSE is. The purpose of the squaring is that the model is evaluated lowly for making larger mistakes. Models trained with the MSE converge reasonably quickly and both train and test performance remain almost equivalent.

Another possibility is to use the MSLE, which has the effect of relaxing the punishing effect, i.e., when we do not want to penalize the model for large differences between the predicted and the exact values. MSLE is appropriate when the model is predicting unscaled quantities directly.

The third option is the MAE. The benefit here is that it is more robust to outliers since it does not make use of the square.

We chose first to scale the data and then to use the MSE, considering the fact that large errors have much bigger consequences than equivalent smaller ones.

6.2.6 Optimization algorithm

As deep learning represents an iterative process and the amount of parameters to be set is significantly large, the importance of the training speed is great as well. The optimization algorithms minimize the loss function and update the learnable parameters, i.e., the weights, with the direction of finding the optimal solution of the problem. Some methods show better performance than others in terms of speed and accuracy [51]. Among the most used in machine learning are the stochastic gradient descent (SGD), Adagrad, and Adam optimizer.

SGD is often implemented with a method called momentum, which accelerates gradients vectors in the right direction and dampens oscillations, thus leading to faster converging, i.e., taking more steps towards the minimum [48]. When using SGD, we can also tune the learning rate, which determines the size of the performed update, i.e., the steps we take to reach a (local) minimum. The learning rate controls how quickly the model is adapted to the problem. If a value too small is chosen, it may result in a long training process that could get stuck, given the smaller changes made to the weights each update. Otherwise, a value too large may result in learning a sub-optimal set of weights in fewer training epochs which makes the training process unstable.

Adagrad adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters. Its main weakness is the accumulation of the squared gradients. The consequence is a fast-shrinking learning rate, eventually becoming so infinitesimally small that at some point the model will not learn again [51].

The adaptive moment estimation (Adam) computes adaptive learning rates for each parameter and keeps an exponentially decaying average of past gradients. This makes it the easiest to use since it tunes the learning rate itself. Adam is well suited for problems that are large in terms of data or parameters, outperforms other adaptive techniques and achieves the most precise results [33, 51].

Implementing SGD and Adam led us to the same conclusion, that Adam is much faster than the alternative and gives better results, even when setting the momentum and the learning rate to different values for the SGD optimizer. Thus, we further consider the Adam optimizer as the best choice for optimization algorithm and proceed with our research to refine the remaining configurations.

6.2.7 Epochs and patience

Another configuration of great importance is the number of training epochs. An epoch is one forward and one backward pass of all the training examples through the ANN. Too many epochs can lead to overfitting of the training dataset, i.e., the model showing poor performance on the test set. Too few epochs may result in underfitting the model, i.e., the training set error is significantly larger than the expected error of an ideal model.

To prevent 'memorization' of training examples, we apply the early stopping method that is used to avoid overfitting when training an ANN model. It allows the specifi-

cation of a sufficiently large number of training epochs and stops training once the model performance stops improving after a certain amount of epochs. This permitted number of epochs is denoted as patience and is set in advance. We chose three different values for the patience to test, i.e., 20, 25 and 30. The early stopping procedure was repeated several times. The epoch number at which training was stopped was recorded. Patience with a value of 20 resulted in models which stopped training before reaching a reasonable accuracy, i.e., converging, and the models with patience values set to 30 needed a great amount of time to be trained and moreover, did not show a higher accuracy than the models trained with patience value set to 25. Consequently, the patience values were set to 25 for all of the test cases.
Further, a much higher value than the average of the epoch number across all repeats of early stopping was used when fitting a final model. The maximum amount of epochs the models could be trained was set to 1000, as this number was never reached. However, it is reasonable to choose such a large value in order to prevent underfitting the model, since it may need more time to be trained and still not show any signs of overfitting.

6.3 Simulations

We implement every ANN structure on Python (version 3.7.3), using Keras (version 2.2.4) - Python deep learning library with TensorFlow (version 1.14.0) - open-source machine learning library for research and production, and Theano (version 1.0.3) - a Python library for defining, optimizing, and evaluating mathematical expressions, as backend. Other libraries we make use of during this research are numpy (version 1.17.1), pandas (version 0.24.2), sklearn (version 0.21.2), and matplotlib (version 3.1.0).

We perform a number of experiments on 235 different networks and three datasets in order to investigate the interrelations between the time of the year and the behavior of the HVAC system:

- Dataset without time index
- Dataset with seasonal index, i.e. winter:= 1, spring:= 2, summer:= 3, autumn:= 4
- Dataset with month index, i.e. january:= 1, february:= 2, march:= 3, etc.

This is done because we cannot have strings in the input of the ANN, since we are not able to carry out simulation calculations on strings.

Every dataset and every activation are also denoted by an index (see Table 12 and Table 13). This notation is later used in the name of each produced model when training every of the examined ANN architectures.

Dataset	Index
No index	0
Seasonal index	1
Month index	2

Table 12: Dataset indices

Activation function	Index
Leaky ReLU	0
Sigmoid	1
Tanh	2

Table 13: Activation function indices

We first conducted a very comprehensive research to find the most optimal hyperparameters for the MLFNN architecture. Afterwards, the RNN structure was also implemented and taking into consideration the results of the MLFNN, we set the hyperparameters of the RNN. The training phase of an RNN is more time consuming and complex, since it consists of substructures, i.e., the LSTMNs, and has more hyperparameters, e.g., sequential length, future period prediction, dropout layers, etc.

The results of both approaches are presented in the following section, as well as an analysis of their performance.

6.4 Results

The following three subsections represent the results of the simulations on both ANNs - MLFNN and RNN, and the analysis used to test the trained models. The purpose of this study is to find an ANN structure that suits best the specific problem of finding the optimal control parameters to be used as initialization for the optimization task.

6.4.1 MLFNN

To assess the MLFNN, we distinguish between different layer structures - three, four and five-layered. Every version is tested on each of the three datasets - without time index, with seasonal index, and with month index. Every simulation was carried out with each of the activation functions - leaky ReLU, sigmoid, and tanh. Additionally, the three, four and five-layered models vary in their node distribution in the layers.

The first set of analyses investigates the three-layered architecture of the MLFNN. The data does not show any uphill or downhill pattern as we increase the number of nodes in the network. From the scatter plot on Fig. 9 we can see that only in the simulation with the sigmoid activation function, there is a slight decreasement in the MSE as we add more nodes. However, the time index, together with the type of activation function, seem to be the most significant factor which influences the MSE. In each of the three plots, the MSE is reduced as we include the time index in the input variables, and more precisely, when the index describes the **season** in which the data was collected. From the three activation function possibilities **sigmoid** shows the best results with an MSE on the dataset with seasonal index preserved in the range of [0.287, 0.352].

The second MLFNN structure we investigate is the four-layered (see Fig. 10). If we compare solely the results of the three and the four-layered MLFNNs, our statement would like to suggest that the MSE declines with a difference of 0.065 for the best

MLFNN models of each category, i.e., showing the minimum MSE of all other alternatives. These best MLFNN models among the three and four-layered architectures are namely the ones trained on the dataset with the **seasonal index** with the **sigmoid** function. Nevertheless, we still state that the improvement of the overall results is mainly achieved by the activation function and the dataset used for the simulations and not by the number of layers.

The last MLFNN structure to be tested is the five-layered. Here the model with the best performance on the test data is again the one simulated on the dataset with the **seasonal index** with the **sigmoid** activation function. Surprisingly, the achieved an MSE is higher than the one reached from the best four-layered model. Concretely, this model performed with an MSE value of 0.227, while the four-layered model achieved 0.222. Furthermore, the other two activation functions show improvement when increasing the number of layers of the network. The best five-layered model trained with tanh accomplishes an MSE of 0.257, while the four-layered only reaches 0.266. The leaky ReLU simulations provide even greater difference in the results, 0.270 and 0.309 for the four and the five-layered models accordingly.

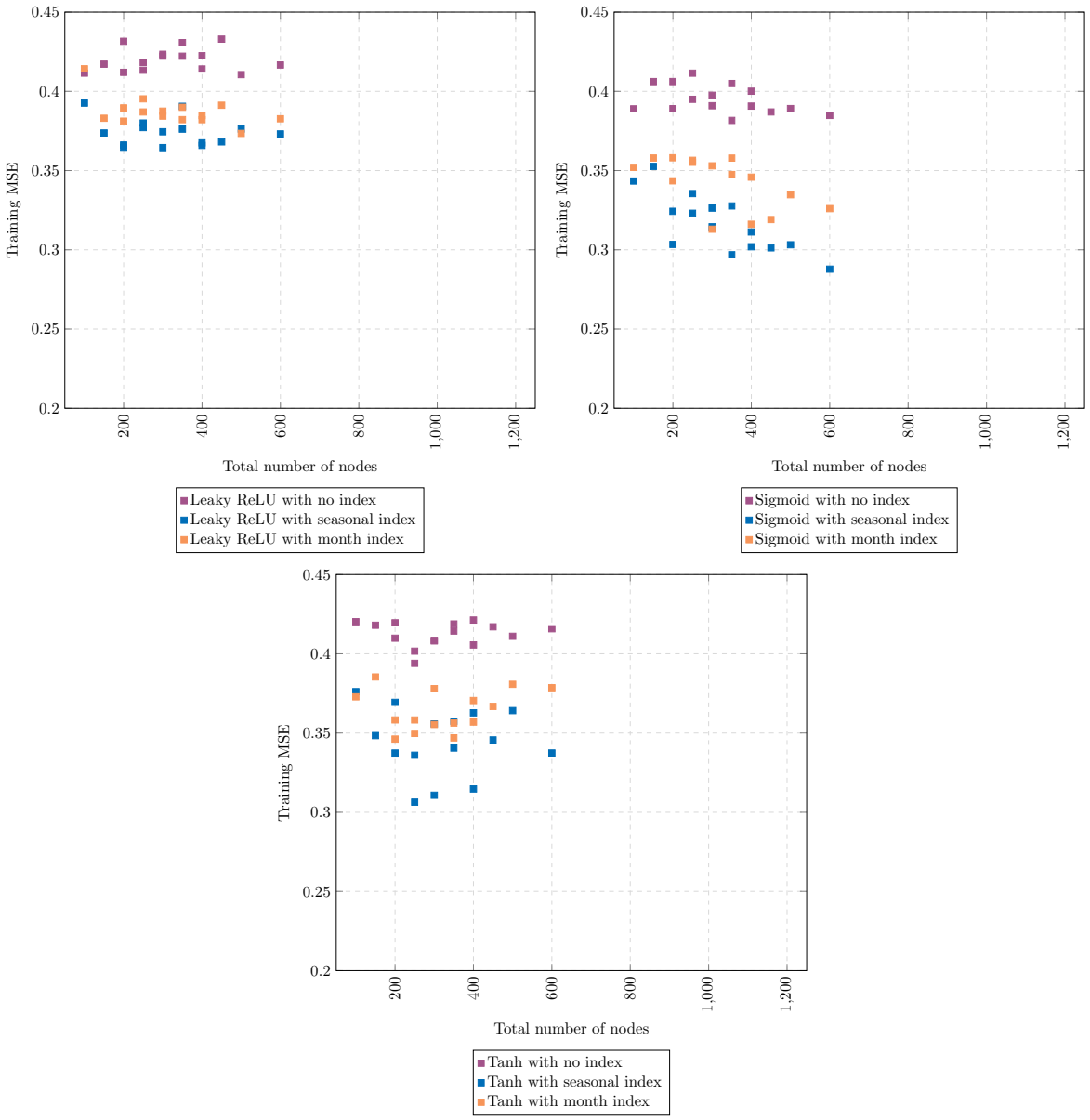

Figure 9: MLFNN with three layers and various node distributions in both hidden layers, tested on the three datasets, respectively with different activation function: leaky ReLU, sigmoid and tanh.

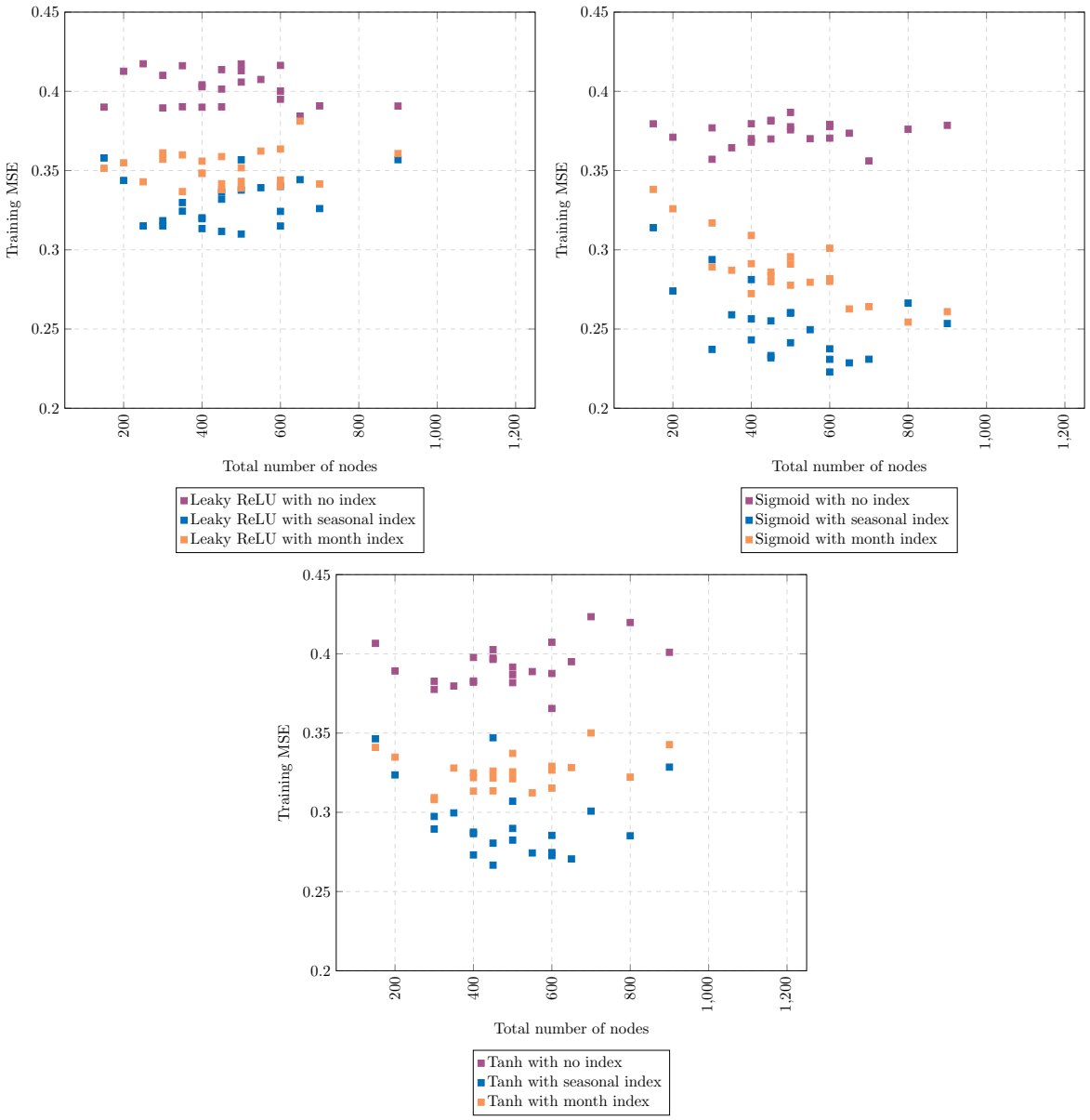

Figure 10: MLFNN with four layers and various node distributions in the layers, tested on the three datasets, respectively with different activation function: leaky ReLU, sigmoid and tanh.

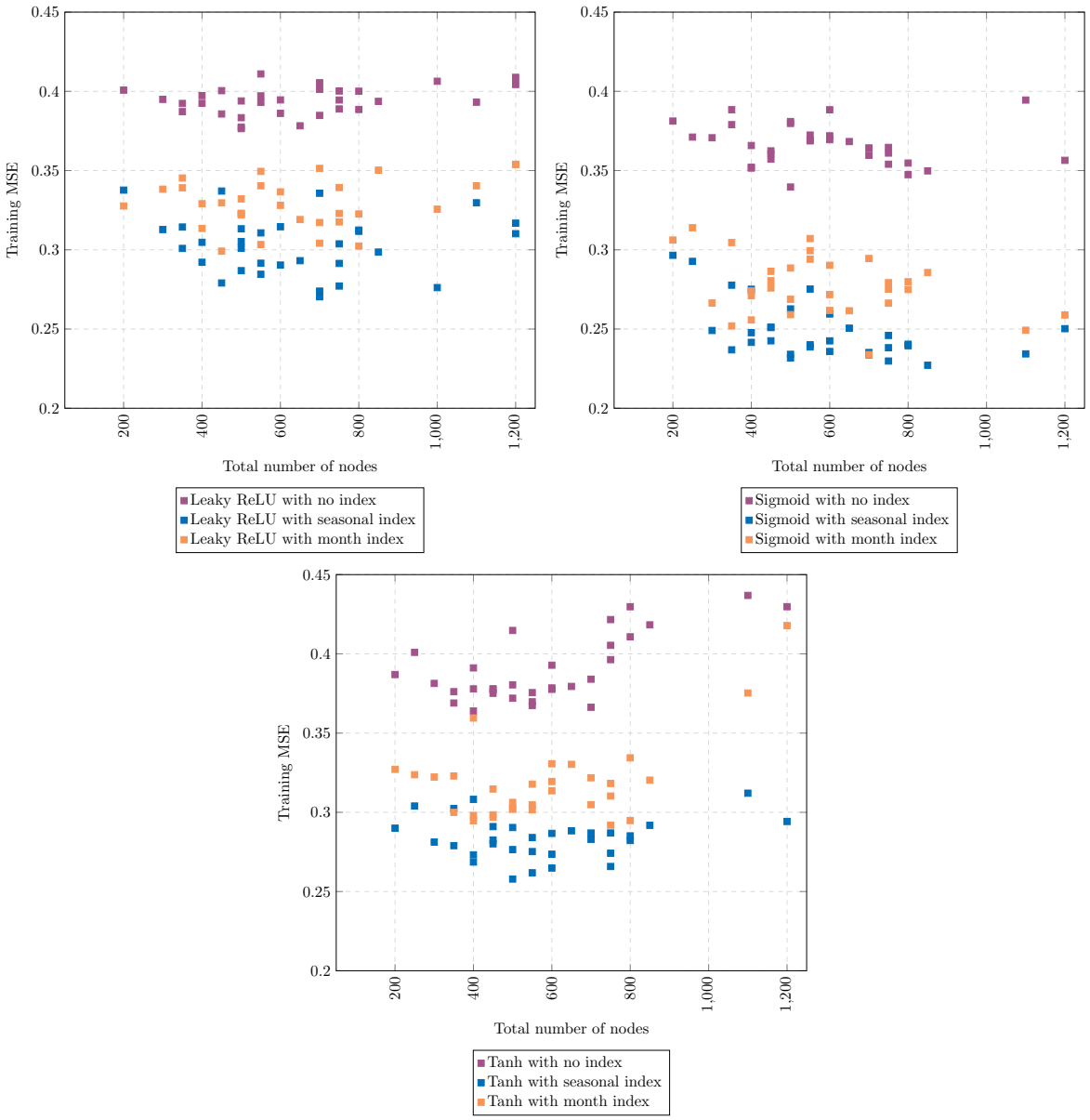

Figure 11: MLFNN with five layers and various node distributions in the layers, tested on the three datasets, respectively with different activation function: leaky ReLU, sigmoid and tanh.

6.4.2 RNN

An alternative solution to the prediction problem is using RNNs. As is well known, RNNs are extremely time-consuming to train and if one epoch of training for the MLFNN takes about 20 to 60 seconds, the time needed for one epoch of RNN to train is from 20 to 60 minutes. One limitation of our research is that the surveys were not conducted in the same period. Thus, we have succeeded in testing only 25 different structures of RNNs. We first chose to set the activation function to tanh, as this was the default one when using the GPU version of TensorFlow to calculate the predictions. However, we did not have access to GPUs and consequently, trained the RNNs on CPUs, which was much slower than expected. To train all of the RNN models with tanh as activation function, we needed two days. Further, we implemented RNNs with leaky ReLU as activation function. These took more than three days. We tested the RNN models with tanh using two different configurations regarding the sequence length and the batch size of the network. Both tests predicted the next 24 time series, and more precisely the next six hours, i.e., six values for the inlet temperature. The first test was carried out with a sequence length of the data set to 288, i.e., representing the previous three days, and batch size of 512. The second one was conducted with a sequence length set to 96, i.e., representing the previous day, and a batch size of 288. We observe from Fig. 12 that there is no clear trend set regarding neither the total number of nodes nor the dataset used (with or without time index). However, the RNN model with a sequence length of 96 and a batch size of 288 does show slightly better results.

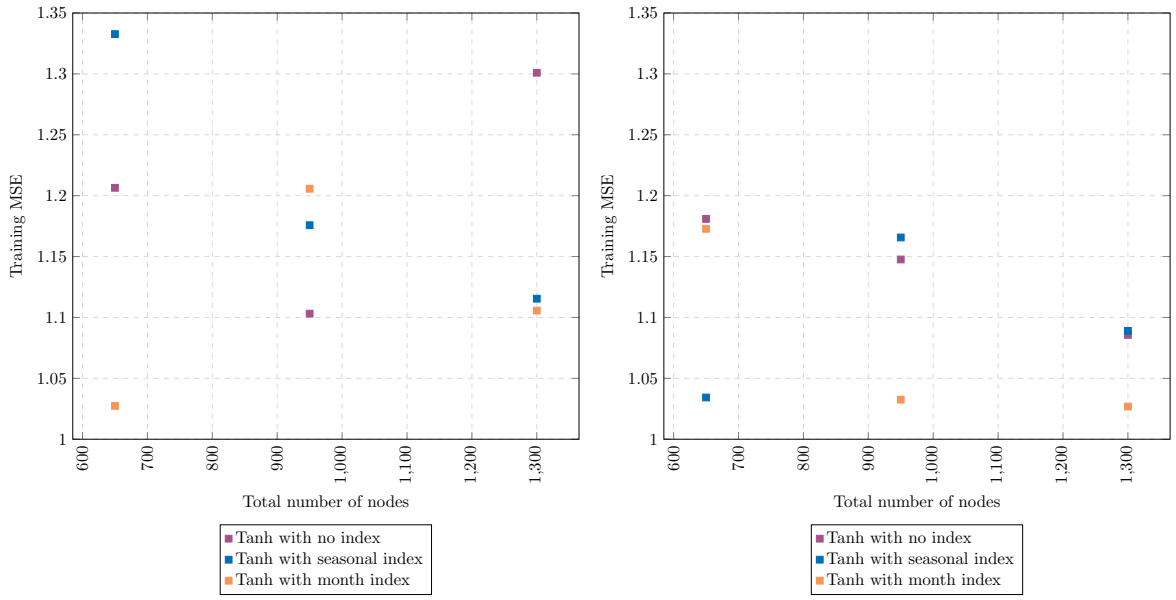

Figure 12: RNN with five layers and various node distributions in the layers, tested on the three datasets, using tanh as activation function; on the left figure we used sequence length of 288 and batch size of 512; on the right figure we used sequence length of 96 and batch size of 288.

Table 14 compares the results of the better RNN model configuration with tanh regarding the MSE during training and during testing. The training MSE is about two times higher than the testing MSE. It is apparent that the best model is the one with the configuration 400/400/300/200. However, the difference between both MSE values remains quite large. Additionally, the testing MSE is not considered as a good result, since values closer to zero are better.

Layer 1	Layer 2	Layer 3	Layer 4	Dataset	Train MSE	Test MSE
200	200	150	100	0	0.414100006	1.180930207
200	200	150	100	1	0.417502378	1.034280589
200	200	150	100	2	0.50042631	1.172782502
300	300	200	150	0	0.498871568	1.147611099
300	300	200	150	1	0.401906525	1.165682642
300	300	200	150	2	0.417380867	1.032484549
400	400	300	200	0	0.436355539	1.085691187
400	400	300	200	1	0.483980823	1.089005239
400	400	300	200	2	0.437019167	1.026820914

Table 14: RNNs with four hidden layers and the results from their training/testing on the three datasets using tanh, sequence length: 96, and batch size: 288 (see Table 12)

6.4.3 Discussion

The tests revealed that no significant correlation was found between the measured MSE value and the total number of the nodes in the layers of the MLFNN. In fact, in contrast with what was previously thought, we found that the MSE stays on the same level with almost no fluctuations regardless of the total number of nodes. Initially, we considered that the more nodes were included, the higher the accuracy of the results would be, as widening consistently improves performance across networks of different depth [67]. However, a closer inspection revealed that this intuition is not always correct. These findings need to be interpreted with caution because our research involves ANNs no deeper than five layers. We did not continue to conduct experiments in that direction since our specific problem was not influenced by these modifications.

It is interesting to note that adding depth to the ANNs was not the key feature that improved the accuracy of the achieved results but rather the choice of input variables contained in the dataset on which it was trained on. The current study was not specifically designed to test different input combinations. Instead, our goal was to test types of ANNs and to study the influence of the hyperparameters on the obtained results. Nevertheless, we note that the input variables are of just as much or sometimes even more significance than the establishment of some hyperparameters. Consequently, we can state that the simulations run on the dataset without time index are clearly more inaccurate than these on the datasets with time index for all of the

simulations without exception. It may easily be verified that the seasonal index gives more accuracy to the predictions than the month index (see Fig. 9, Fig. 10, and Fig. 11). A reasonable explanation for this is that the seasonal index is more general than the month index. Simultaneously, the outdoor conditions are very similar in each season and apparently, this information is sufficient to the model.

On combining the results of the MLFNN model, we conclude that if we have to evaluate each activation function, sigmoid is definitely the winner, followed by tanh. However, further analysis showed that while training sigmoid is the slowest, which is consistent with previous results [16]. Despite this, models are trained only once and thus, there is no difference in the length of the prediction process. Therefore, this would not represent any problem for the application of this approach, since there are no limitations regarding the training time.

As far as the MSE can show us how close a regression line is to a set of points, we want to get a better overview of the quality of the obtained predictions. With this said, we take the best MLFNN model - four-layered MLFNN with 200 nodes per layer, with sigmoid activation function, trained on the data set with seasonal index (see Fig. 13).

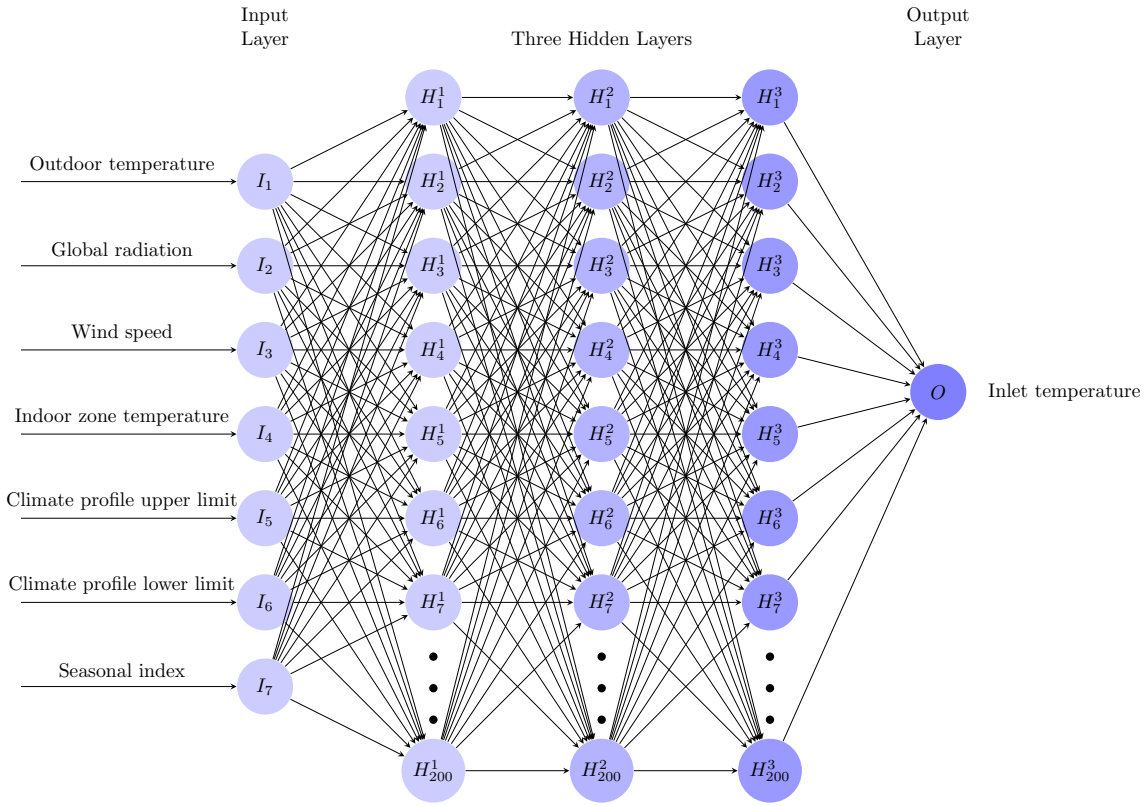

Figure 13: The best MLFNN model structure in terms of reached minimal MSE on the dataset with a seasonal index

Using this model, we estimated the values of the predicted inlet temperature. Then we compared the results to the exact values, calculating the relative error and multiplying it by 100 in order to work in percentage. The results show that the highest relative error is 470.9220835% and the lowest is 0.001507492%. However, 89.17% of the data's relative error lays in the range of [-50%, 50%]. As illustrated on Fig. 14, the highest peaks are within the range [0%, 7%], where 48.5% of the data lays.

Although the performance of the MLFNN models was not ideal, we still believe that the experimental results show a reasonable low error and the proposed MLFNN approach can be used for prediction of the initialization variables of the optimizer.

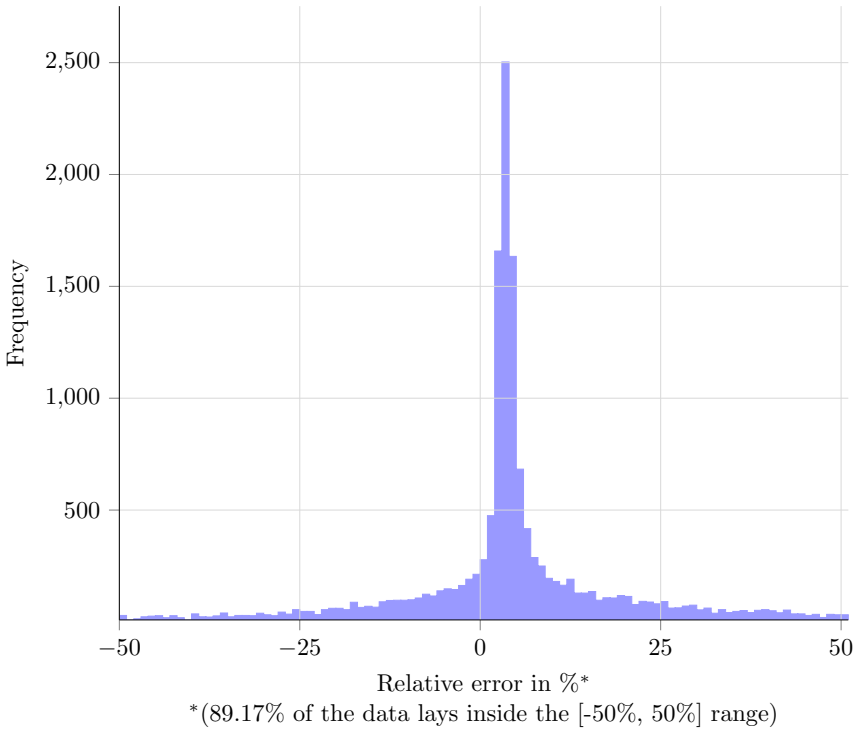

Figure 14: Frequency of the relative error of the predictions made by the best MLFNN model - three-layered with 200 nodes in each layer and sigmoid as activation function

Despite the limitations of our RNN testing method and consequently the poor results in our first tests, our findings do nevertheless suggest that there are several reasons for such high testing MSE values and they all lead to the main problem arising and namely overfitting, which can be more closely observed in Fig. 15 where our best RNN model with nodes distributed in the hidden layers as follows 300/300/200/150 and trained with leaky ReLU shows signs of overfitting. Overfitting occurs when a model is too complex and the overall cost is minimal, but the generalization of the model is unreliable. This makes the model informed by too many features. Consequently, the model memorizes training data rather than learning to generalize from trend. We will propose possible solutions to face the problem later in the next section.

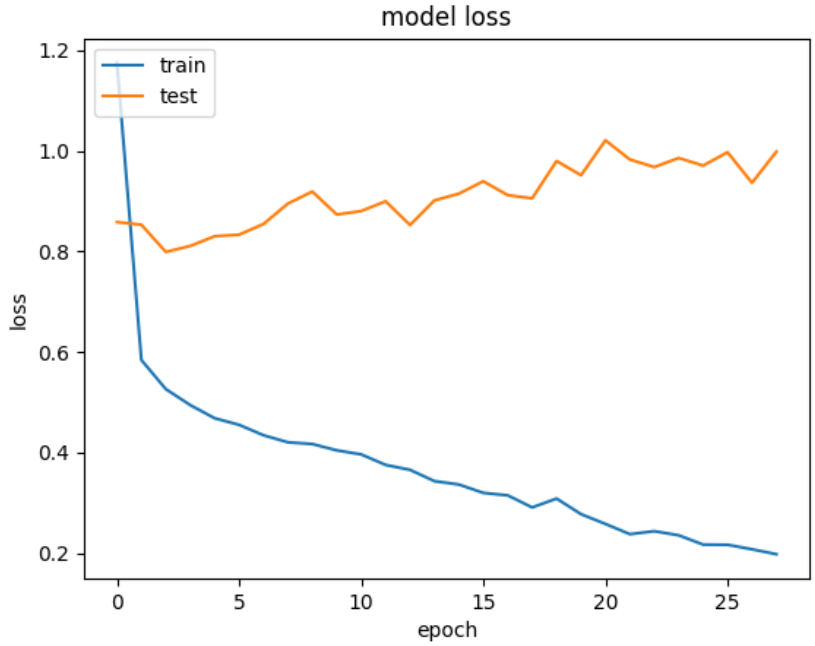

Figure 15: Training process of the 300/300/300/150 RNN model with leaky ReLU; the model shows signs of overfitting, since the training error decreases, converges and reaches fairly good results, whereas the testing error raises with every epoch

7 Future work

We hope that our research will serve as a base for future studies on finding the optimal values for the optimization task not only as initialization parameters but also as an accurate solution of the given problem. It is recommended that further research should be undertaken in the following areas.

7.1 Number of nodes

This research has given rise to the question if the number of nodes influences the punctuality of the ANN or it just adds unnecessary complexity to the structure of the model and hence, causes it to overfit. This apparent lack of correlation between the number of nodes and the MSE value can be attributed to the fact that we examined a very narrow range for the total amount of nodes in the networks, i.e., [100, 1200] for the MLFNN models and [650, 1300] for the RNN models. The reason for this rather contradictory result is still not completely clear, but our results are promising and should be validated by a larger sample size.

7.2 Overfitting

More broadly, research is also needed to determine if the ANN models are prone to overfit. If we examine closer our best MLFNN model which reaches an MSE of 0.222 during testing, we can identify that the MSE value during training is about 1.6 times lower, i.e., 0.137 and thus, overfitting is not inconceivable. Future work should also focus on enhancing the quality of the proposed RNN models because of the large gap between the training and the testing error (see Fig. 15). In fact, we have undertaken measures to prevent our models from overfitting by employing techniques such as early stopping and dropout. However, they do not provide us with tolerable values of the error, especially in the RNN case and must be investigated further.

The following subsections propose other possible solutions that future researchers should take into account when facing this problem.

7.2.1 More data

Such a solution is suggested due to the fact that the number of training examples may be insufficient for the model in order to learn the patterns properly. Increasing the size of the new observations of the training dataset may improve the accuracy of the model. Nevertheless, this is not always the case and may have no influence on the precision of the model.

7.2.2 Dropout

Dropout is a method for addressing overfitting by randomly dropping units and their connections from the ANN during training. This prevents units from co-adapting too much and significantly reduces overfitting by giving major improvements over other regularization methods [60]. In each training case, each hidden unit is randomly omitted from the network with a probability that can vary, depending on the rate of every layer [21]. The rate can be set to a value in the range [0, 1]. Therefore, a hidden unit cannot rely on other hidden units being present. We used this approach in the implementation of our RNN models with rates in the set $\{0.5, 0.7, 0.8, 0.9\}$ taking into consideration the fact that large networks are very slow to train. However, it did not show sufficiently good results and therefore, further experimental investigations are needed to estimate the optimal values of the rate for each layer.

7.2.3 Pruning

Pruning is another technique that produces smaller, faster and more power-efficient models that compute their predictions with minimal loss. Applying pruning, the neurons in the network are ranked according to the importance of their connections, i.e., how much they contribute to the overall result. Then the low ranked neurons are removed from the network. This leads to better generalization results, improved processing speed, and reduced size.

7.2.4 Ensembling

Another course of action to be taken is ensembling. Ensembling multiple neural networks, i.e., training them and then combining their predictions can significantly improve their generalization ability [68]. Ensembling also reduces the variance of predictions and may produce a higher prediction accuracy than any other single model.

7.2.5 Weight constraints

Large weights lead to unstable ANNs that are sensitive to changes in the input variables, i.e., overfit ANNs. In turn, the overfit networks have poor performance when making predictions on new unseen data. Therefore, a practical way to circumvent this issue is to use a weight constraint. A weight constraint triggers the minimization of the weight size and scales the weights according to a pre-defined threshold. Thus, the weights are forced to shrink, avoiding very large learning rates.

8 Acknowledgments

First and foremost, I have to thank my research supervisor Dr. rer. nat. Pascal Richter. Without his support and dedicated involvement in every step throughout the process, this thesis would have never been accomplished.

I gratefully acknowledge Ahmed Abida for the valuable suggestions and discussions, and Gereon Kremer for the provided help.

I also thank Prof. Dr. rer. nat. Erika Ábrahám for giving me the opportunity to work on this thesis.

References

[1] Nivine Attoue, Isam Shahrour, and Rafic Younes. Smart building: Use of the artificial neural network approach for indoor temperature forecasting. *Energies*, 11(2):395, 2018.

[2] Mesut Avci, Murat Erkoc, Amir Rahmani, and Shihab Asfour. Model predictive HVAC load control in buildings using real-time electricity pricing. *Energy and Buildings*, 60:199–209, 2013.

[3] U. Teoman Aksoy B. B. Ekici. Prediction of building energy consumption by using artificial neural networks. *Applied Energy*, 2008.

[4] Peder Bacher and Henrik Madsen. Identifying suitable models for the heat dynamics of buildings. *Energy and Buildings*, 43(7):1511–1522, 2011.

[5] Abdullatif E Ben-Nakhi and Mohamed A Mahmoud. Energy conservation in buildings through efficient a/c control using neural networks. *Applied Energy*, 73 (1):5–23, 2002.

[6] M Castilla, JD Álvarez, MG Ortega, and MR Arahal. Neural network and polynomial approximated thermal comfort models for hvac systems. *Building and Environment*, 59:107–115, 2013.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[8] Turgay Coşkun. The importance of internal heat gains for building cooling design. *Journal of Thermal Engineering*, 3(1):1060–1064, 2017.

[9] Organisation de coopération et de développement économiques. *Transition to sustainable buildings: Strategies and opportunities to 2050*. OECD Publishing, 2013.

[10] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[11] IEA Energy Efficiency. Market report series. *IEA: Paris, France*, 2017.

[12] B Egilegor, JP Uribe, G Arregi, E Pradilla, and L Susperregi. A fuzzy control adapted by a neural network to maintain a dwelling within thermal comfort. In *Proceedings of Building Simulation*, volume 97, pages 87–94. Citeseer, 1997.

[13] Ping Fang, Tingzhang Liu, Kai Liu, Yingqi Zhang, and Jianfei Zhao. A simulation model to calculate temperature distribution of an air-conditioned room. In *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 1, pages 378–381. IEEE, 2016.

[14] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

[15] Antoine Garnier, Julien Eynard, Matthieu Caussanel, and Stéphane Grieu. Low computational cost technique for predictive management of thermal comfort in non-residential buildings. *Journal of Process Control*, 24(6):750–762, 2014.

[16] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[17] Pedro A Gonzalez and Jesus M Zamarreno. Prediction of hourly energy consumption in buildings based on a feedback artificial neural network. *Energy and buildings*, 37(6):595–601, 2005.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[19] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[20] Hugo Hens, Wout Parijs, and Mieke Deurinck. Energy consumption for heating and rebound effects. *Energy and buildings*, 42(1):105–110, 2010.

[21] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[22] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[23] Richard Holz, Andrew Hourigan, Richard Sloop, Paul Monkman, and Moncef Krarti. Effects of standard energy conserving measures on thermal comfort. *Building and environment*, 32(1):31–43, 1997.

[24] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[25] Hao Huang, Lei Chen, and Eric Hu. A neural network-based multi-zone modelling approach for predictive control system design in commercial buildings. *Energy and buildings*, 97:86–97, 2015.

[26] Hao Huang, Lei Chen, and Eric Hu. A new model predictive control scheme for energy and cost savings in commercial buildings: An airport terminal building case study. *Building and environment*, 89:203–216, 2015.

[27] S Kalogirou, G Florides, C Neocleous, and C Schizas. Estimation of daily heating and cooling loads using artificial neural networks. In *2001 World Congress, Napoli (September)*, 2001.

[28] SA Kalogirou, CC Neocleous, and CN Schizas. Building heating load estimation using artificial neural networks. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, volume 8, page 14, 1997.

[29] Soteris A Kalogirou. Artificial intelligence in renewable energy systems modelling and prediction. *Proceedings of the World Renewable Energy Congress VII, Cologne, Ger*, 2002.

[30] Soteris A Kalogirou. Artificial neural networks in energy applications in buildings. *International Journal of Low-Carbon Technologies*, 1(3):201–216, 2006.

[31] S Karatasou, M Santamouris, and V Geros. Modeling and predicting building's energy use with artificial neural networks: Methods and results. *Energy and buildings*, 38(8):949–958, 2006.

[32] Tuğçe Kazanasmaz, Murat Günaydin, and Selcen Binol. Artificial neural networks to predict daylight illuminance in office buildings. *Building and Environment*, 44 (8):1751–1757, 2009.

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

[34] Kangji Li, Hongye Su, and Jian Chu. Forecasting building energy consumption using neural networks and hybrid neuro-fuzzy system: A comparative study. *Energy and Buildings*, 43(10):2893–2899, 2011.

[35] Nan Li, Jun-young Kwak, Burcin Becerik-Gerber, and Milind Tambe. Predicting hvac energy consumption in commercial buildings using multiagent systems. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 30, page 1. Vilnius Gediminas Technical University, Department of Construction Economics ..., 2013.

[36] Qiong Li, Qinglin Meng, Jiejin Cai, Hiroshi Yoshino, and Akashi Mochida. Predicting hourly cooling load in the building: A comparison of support vector machine and different artificial neural networks. *Energy Conversion and Management*, 50(1):90–96, 2009.

[37] Jian Liang and Ruxu Du. Thermal comfort control based on neural network for hvac application. In *Proceedings of 2005 IEEE Conference on Control Applications, 2005. CCA 2005.*, pages 819–824. IEEE, 2005.

[38] Tao Lu and Martti Viljanen. Prediction of indoor temperature and relative humidity using neural network models: model comparison. *Neural Computing and Applications*, 18(4):345, 2009.

[39] Mohamad Kheir Mohamad. developing a thermal model forresidential room using simulink/matlab. *SetkánìcatederlvlecbanikyTekutin A Termomechaniky26.-28. Cervna*, 2012.

[40] Nicolas Morel, Manuel Bauer, Mario El-Khoury, and Jens Krauss. Neurobat, a predictive and adaptive heating control system using artificial neural networks. *International Journal of solar energy*, 21(2-3):161–201, 2001.

[41] D Subbaram Naidu and Craig G Rieger. Advanced control strategies for hvac&r systems—an overview: Part ii: Soft and fusion control. *HVAC&R Research*, 17 (2):144–158, 2011.

[42] Nabil Nassif. Modeling and optimization of hvac systems using artificial neural network and genetic algorithm. In *Building Simulation*, volume 7, pages 237–245. Springer, 2014.

[43] Alberto Hernandez Neto and Flávio Augusto Sanzovo Fiorelli. Comparison between detailed model simulation and artificial neural network for forecasting building energy consumption. *Energy and buildings*, 40(12):2169–2176, 2008.

[44] Jennifer L Nguyen, Joel Schwartz, and Douglas W Dockery. The relationship between indoor and outdoor temperature, apparent temperature, relative humidity, and absolute humidity. *Indoor air*, 24(1):103–112, 2014.

[45] Vildan Ok. A procedure for calculating cooling load due to solar radiation: the shading effects from adjacent or nearby buildings. *Energy and buildings*, 19(1): 11–20, 1992.

[46] Linda Pedersen. Use of different methodologies for thermal load and energy estimations in buildings including meteorological and sociological input parameters. *Renewable and Sustainable Energy Reviews*, 11(5):998–1007, 2007.

[47] Luis Pérez-Lombard, José Ortiz, and Christine Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.

[48] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[49] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[50] Antonio E Ruano, Eduardo M Crispim, Eusébio ZE Conceiçao, and Ma Manuela JR Lúcio. Prediction of building's temperature using neural networks models. *Energy and Buildings*, 38(6):682–694, 2006.

[51] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[52] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[53] SH Saboksayr, RV Patel, and M Zaheer-Uddin. Energy-efficient operation of hvac systems using neural network based decentralized controllers. In *Proceedings of 1995 American Control Conference-ACC'95*, volume 6, pages 4321–4325. IEEE, 1995.

[54] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

[55] Gianluca Serale, Massimo Fiorentini, Alfonso Capozzoli, Daniele Bernardini, and Alberto Bemporad. Model predictive control (mpc) for enhancing building and hvac system energy efficiency: Problem formulation, applications and opportunities. *Energies*, 11(3):631, 2018.

[56] Nugroho Setiawan, I Wayan Mustika, Adha Imam Cahyadi, and Muhammad Fikri. State space modeling of thermal in a room for temperature estimation in wireless sensor network. In *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, pages 202–206. IEEE, 2017.

[57] Xiaoliang Shao, Xiaojun Ma, Xianting Li, and Chao Liang. Fast prediction of non-uniform temperature distribution: A concise expression and reliability analysis. *Energy and Buildings*, 141:295–307, 2017.

[58] Neural Smithing. Supervised learning in feedforward artificial neural networks, 1999.

[59] Mohsen Soleimani-Mohseni, Bertil Thomas, and Per Fahlen. Estimation of operative temperature in buildings using artificial neural networks. *Energy and Buildings*, 38(6):635–640, 2006.

[60] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[61] GM Stavrakakis, DP Karadimou, PL Zervas, H Sarimveis, and NC Markatos. Selection of window sizes for optimizing occupational comfort and hygiene based on computational fluid dynamics and neural networks. *Building and environment*, 46(2):298–314, 2011.

[62] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.

[63] Anders Thavlov and Henrik W Bindner. Thermal models for intelligent heating of buildings. In *4th International Conference on Applied Energy (ICAE 2012): Energy innovations for a sustainable world*, 2012.

[64] Jörn von Grabe. Potential of artificial neural networks to predict thermal sensation votes. *Applied energy*, 161:412–424, 2016.

[65] Ye Yao, Zhiwei Lian, Shiqing Liu, and Zhijian Hou. Hourly cooling load prediction by a combined forecasting model based on analytic hierarchy process. *International journal of thermal sciences*, 43(11):1107–1118, 2004.

[66] Kyungtae Yun, Rogelio Luck, Pedro J Mago, and Heejin Cho. Building hourly thermal load prediction using an indexed arx model. *Energy and Buildings*, 54: 225–233, 2012.

[67] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[68] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.