Control of an air conditioning device by recurrent neural networks

1st Julius Haas
Institute of Automation Engineering (Faculty 07)
University of Applied Sciences Cologne (TH Köln)
Cologne, Germany
julius.haas@smail.th-koeln.de

2nd Prof. Karl Kohlhof
Institute of Automation Engineering (Faculty 07)
University of Applied Sciences Cologne (TH Köln)
Cologne, Germany
karl.kohlhof@th-koeln.de

***Abstract*—This paper deals with the creation of training data that is consecutively used to train a reinforcement learning model including a recurrent neural network with the aim of controlling the climate parameters of an interior space while minimizing energy consumption.**

***Index Terms*—fanger, pmv, ppd, climate control, artificial intelligence, neural networks, reinforcement learning**

I. Introduction

In 1962 Gordon Moore predicted that the numbers of transistors on an integrated circuit chip would double every two years [1]. His forecast materializes and the computer power increases exponentially, with improvements such as strong GPUs and quantum computing. As a result, artificial intelligence (AI) algorithms developed by mathematicians in the course of the last decades can now be applied in practice. Data driven methods such as machine learning, deep learning and reinforcement learning are therefore becoming more prevalent day by day, with practical applications in the field of computer vision, bioinformatics, robotics, business intelligence and more.

However, data driven approaches in the field of feedback control engineering are still in their infancy and hence offer the potential for a wide range of applied research. Artificial intelligence holds great potential in scenarios where only limited knowledge about the physical model of the system is available. A classical control feedback approach is not possible in these cases as the system's behaviour might not be analytically determinable at all or establishing a system model would be too time-consuming.

II. Problem Description

The climate of an inside room is to be controlled using a Samsung room climate device featuring different levels of heating, cooling and ventilation that correspond to different levels of energy consumption. The rooms' parameters, such as temperature, humidity and more are measured by sensors. The signals then serve as inputs to a feedforward neural network (chapter II-C) that has been implemented in a previous project by Marouen Abdi [16]. Based on its input signals, the feedforward network approximates a *predicted mean vote value* PMV (chapter II-A) for the thermal comfort. The output of the feedforward network is directed to a long short-term memory cell (chapter II-C) that predicts a future PMV_{t+1} value with an accuracy of 99%, based on a series of 10 PMV values [16]. The PMV_{t+1} value is then used as a input state variable for a software agent that is trained with reinforcement learning (chapter III).

A research group in Thailand implemented the calculation of PMV values using neural nets in a first approach in 2005. However, they did not proceed to using the PMV for a control model [21].

The control model is based on the parameters of the empirical equations introduced by Fanger [4] which are outlined in chapter II-A below. Different approaches are conceivable to solve this optimization task. While my team member Nils Bitzer uses feedforward neural nets to control the room climate [18], the objective of this paper is to control the room climate using a reinforcement learning agent. The concept of reinforcement learning is explained in chapter III. Initially, the goal is to create training data to train the algorithm. Therefore no sensor technology is needed yet. The training data is calculated using the Fanger equations [4]. In a later project stage, the algorithm for climate control will be implemented on hardware. When performing on a microcontroller, they will then be provided with sensor data by a *Smart Micro Sensor System* that was developed by Wajdi Araar in an earlier project [17].

In order to fully understand the content of the paper, some theoretical concepts will be outlined shortly. It does not claim to cover the topic in complete depth, as this would go beyond the scope of this paper. In order to comprehend the theory behind the covered topics in full detail, please have a look at the references listed at the end.

A. Predicted Mean Vote PMV

The Predicted Mean Vote (PMV) is an empirical value of human thermal comfort based on the sensation of heat and the dependence on thermal equilibrium of human beings in the form of skin temperature and sweat secretion

[8]. The PMV value was first described by Fanger after conducting an empirical experiment with a large number of persons [4]. His goal was to determine the most important biophysical prerequisites to impact the thermal comfort of a human being. Thermal comfort may therefore vary greatly between individuals based on their empirical perception of their comfort temperature. According to Fanger, the identified parameters are *air temperature, mean radiant temperature, relative humidity, air speed, clothing insulation and metabolic rate* [4].

The PMV value is defined for the following range:

$$D_f = \{PMV \in \mathbb{R} \mid -3 \leq PMV \leq 3\} \tag{1}$$

Thermal comfort is assumed for:

$$PMV = [-0.5, 0.5] \tag{2}$$

All values outside this interval are assumed as not optimal. With increasing value, the thermal comfort decreases [8]. The value *percentage of people dissatisfied* with a given room climate (PPD) yields the possibility of quantifying the number of people not feeling comfortable with the room climate. The PPD value correlates with the PMV value. As the PMV and the PPD value both represent empirical values, a certain amount of people will feel uncomfortable, even within the PMV range that is considered optimal (equation 2).

B. Control Model

The climate control task covered in this paper can be described with a classical feedback control model (figure 1). A feedback control model is a model whose output is controlled using a feedback signal y_m which is compared with a reference signal r to generate an error signal e. The error signal e is then used by a controller to produce the system's control input u. The system responds with a signal y according to its input signal u and its behavior [5].

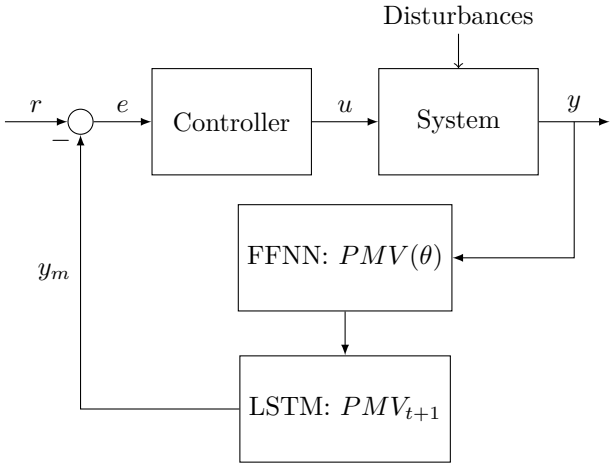

Fig. 1. Control model

C. Neural Networks

Neural networks are a construct to approximate functions without the necessity of a complete model description. They consist of a number of interconnected neurons. Each neuron provides an output value O based on the sum of its input signals, their respective weights, its activation function φ and the corresponding threshold value κ. The indices i and j represent the number of neurons and layers [18].

$$O_j = \varphi(\sum_{i=0}^{n} x_i * w_{ij} - \kappa) \tag{3}$$

The intercommunication of synapses inside the human brain serves as a template model for neural networks. The functionality of the artificial neuron is based on the biological neuron. Neural nets normally consist of an input layer, one or more hidden layers and an output layer. Neural nets with more than 2 hidden layers are considered *deep neural networks*. Single actions that take place in the hidden layers are not comprehensible. The variation of different activation functions, number of neurons per layer and number of layers results in a great number of variations of neural networks.

a) Feedforward neural networks (FFNN): Feedforward networks are the quintessential deep learning models. The goal of a feedforward network is to approximate some function f^*. For example, for a classifier, $y = f^*(x)$ maps an input x to a category y. A feedforward network defines a mapping $y = f(x; \kappa)$ and learns the value of the parameters κ that result in the best function approximation. These models are called *feedforward* because information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself. When feedforward neural networks are extended to include feedback connections, they are called *recurrent neural networks* [3].

b) Recurrent neural networks (RNN): Recurrent neural networks are a family of neural networks that include feedback connections and are best for processing sequential data [10]. While a convolutional neural network is specialized for processing a grid of values, such as an image, a recurrent neural network is specialized for processing a sequence of input values $x^{(0)}, x^{(1)}, ..., x^{(\tau)}$ [3]. It calculates an output state $h^{(0)}, h^{(1)}, ..., h^{(\tau)}$, as a function of the previous state $h^{(t-1)}$ and the current input signal x^t. The recurrent net learns the value of the parameters θ that result in the best function approximation.

$$h^{(t)} = f(h^{(t-1)}, x^t; \theta) \tag{4}$$

Figure 2 demonstrates an unfolded recurrent network.

When training artificial neural networks with gradient-based learning methods and backpropagation, the neuron's weights are updated proportional to the partial

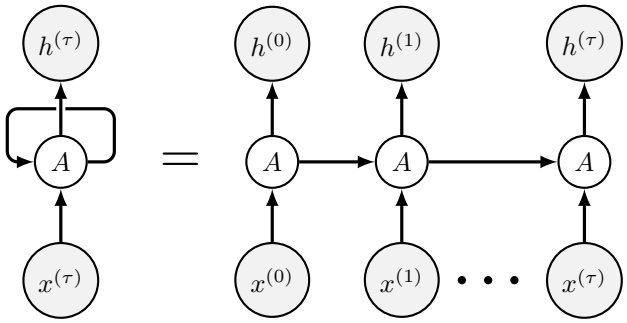

Fig. 2. Recurrent neural network

derivate of the error function with respect to the current weight in each iteration of training. Because the partial derivates for each layer are multiplied with each other in order to determine the gradient of the error function, the gradient is vanishing for values below 1 and exploding for values greater than 1. This phenomenon was first formally described as the *vanishing gradient problem* by Josef Hochreiter in his diploma thesis [13]. In order to solve the vanishing gradient problem, Hochreiter and Schmidhuber introduced *long short-term memory* (LSTM) cells [11] based on the idea of creating paths through time that have derivates that neither vanish nor explode [3]. This was accomplished by introducing gated recurrent units with a self-loop.

III. Reinforcement Learning Algorithm

In reinforcement learning an autonomous software agent must learn to perform a task by trial and error without any guidance from the human operator. An environment is modelled as a set of states $s_t \in \mathbb{S}$. The software agent performes actions $a_t \in \mathbb{A}(s)$ to control the system's state and receives the updated state s_{t+1} and a reward value $r_t \in \mathbb{R}$ in a feedback loop [6]. The feedback variables then determine the agents next action.

In the application of climate comfort, action a_t is one or a combination of the heating, cooling or ventilation levels defining the action space $\mathbb{A}$. State s_t is the PMV value provided by the LSTM cell and the reward r_t is a value corresponding to the energy consumption to reach state s_t. Reinforcement learning requires choosing a trade-off between exploration of unknown actions and exploitation of known actions [3].

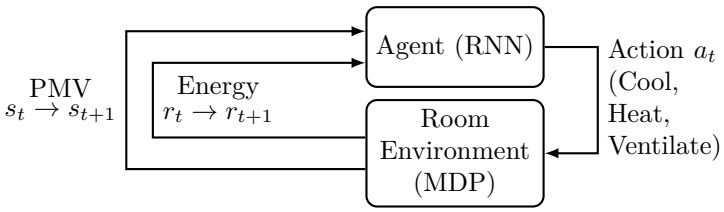

Fig. 3. Reinforcement learning

The environment is often modelled as a discrete-time stochastic control *Markov decision process* (MDP) [14]. The concept of reinforcement learning gained attention in 2015 when AlphaGo won in the ancient game of Go against the multiple European champion Fan Hui [15].

A. Mathematical Description

The agent's action selection is modeled as a map called *policy* π.

$$\pi : A \times B \rightarrow [0, 1] \tag{5}$$

The *policy* is a function of the current state and action and is defined as the conditional probability P of an action a_t taking place, given the state s_t.

$$\pi(s, a) = P(a_t|s_t) \tag{6}$$

Summing up the reward r for each step and multiplying it by the *discount* γ, representing a weight, yields the *cumulative reward* G_t.

$$G_t = \sum_{k=0}^{T} \gamma_k \cdot r_{t+k+1} \quad with \quad 0 \leq \gamma \leq 1 \tag{7}$$

The *state-value function* for a given *policy* π is denoted V^π. For each state $s \in S$, it yields the expected return if the agent starts in state s and then uses the policy to choose its *actions* for future time steps. We refer to $V_\pi(s)$ as the value of *state* s, under *policy* π.

$$V^\pi(s) = E_{a\sim\pi}[G_t|S_t = s] \tag{8}$$

The *action-value function* for a given policy π is denoted Q^π. For each state $s \in S$ and action $a \in A$, it yields the expected return if the agent starts in state s, takes action a and then uses the policy to choose its actions for future time steps. We refer to $Q_\pi(s)$ as the value of *state-action pair* s, a, under policy π.

$$Q^\pi(s, a) = E_{a\sim\pi}[R_t|S_t = s, A_t = a] \tag{9}$$

The *advantage function* A^π determines the quality of a random action over the currently promoted optimum solution by policy π [20].

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \tag{10}$$

One option for reinforcement learning algorithms is to adjust the parameters of the neural net, in its function as the agent, is to use *gradient descent* methods. Generally speaking, gradient descent methods minimize the value of some function $f(x)$ by altering x in small steps. At best, the value of x is altered in a way to most appropriately minimize the function value. In order to determine that the best change of the input variable x, the function derivate $\frac{d}{dx}f(x)$ is calculated. For multivariate functions, the gradient $\nabla_x f(x)$ is calculated by calculating all partial derivates $\frac{\partial}{\partial x_i}f(x)$ and generalizing them to a vector by multiplying the partial derivate with its unit vector $\hat{e}_i$, as in the equation below.

$$\nabla_x f(x) = \sum_{i=0}^{n} \left[\frac{\partial}{\partial x_i}\right] \hat{e_i} = \left[\frac{\partial}{\partial x_1}\right] \hat{e_1} + ... + \left[\frac{\partial}{\partial x_n}\right] \hat{e_n} \quad (11)$$

In the case of reinforcement learning, the function $f(x)$ is often referred to as the *cost function*, *loss function* or *error function* [3].

IV. Hardware

The Samsung Maledives R32 [9] will ultimately be used to control the room climate in a later stage of the project. As emphasized in chapter II, the initial goal is to create training data for the reinforcement learning agent and Nils Bitzer's feedforward network [18]. As the agent's actions $a_t \in \mathbb{A}(s)$ depend on the hardware that is used in a later stage of the project, the creation of training data is based on the information provided in the hardware specification.

The relevant Samsung Maledives R32 specifications are as follows:

- Heating power:
  $H = 3.8kW \quad (1.3kW \sim 4.9kW)$ (5 levels)
- Cooling power:
  $C = 3.5kW \quad (0.9kW \sim 4.0kW)$ (3 levels)
- Ventilation power:
  $V = 678\frac{m^3}{h}$ (3 levels)

Therefore, the following heating-, cooling- and ventilation-levels will be used for the training data creation:

- Heating levels (5 levels):
  $H_{lvlI} = 1.3kW$, $H_{lvlII} = 2.2kW$, $H_{lvlIII} = 3.1kW$, $H_{lvlIV} = 4.0kW$, $H_{lvlV} = 4.9kW$
- Cooling levels (3 levels):
  $C_{lvlI} = 0.9kW$, $C_{lvlII} = 2.5kW$, $C_{lvlIII} = 4.0kW$
- Ventilation levels (3 levels):
  $V_{lvlI} = 0.15\frac{m}{s}$, $V_{lvlII} = 0.30\frac{m}{s}$, $V_{lvlIII} = 0.45\frac{m}{s}$

V. Training Data Creation

In order to successfully train the algorithms, training data is needed. The reinforcement learning algorithm can be either trained *online* or *offline*. In an online training process a python simulator constantly delivers training data for the agent to perform and maximize his reward function. In an offline training process, a batch of training data is created upfront for the agent to train on. In this project, the concept of offline training was used. In doing so, different algorithms such as Nils Bitzer's feedforward network [18] and possible future project could benefit from the data.

In a first step, the PMV values for different input parameters are calculated. Subsequently, the different possible actions $a_t \in \mathbb{A}(s)$ are be added by a python script to each line of PMV values. Finally, a dataset (see table III) is compiled.

The algorithmic steps of the training data creation are outlined and the python implementation is discussed hereafter.

A. Fanger's PMV calculation

Firstly, the PMV values are calculated by wrapping the code for the PMV value-calculation into a python script that calculates the values for a given set of parameters and parameter resolution. The code for the PMV calculation is based on the Fanger's equations [4] and is presented in the ISO 7730 standard [8].

The following parameter ranges are used for the PMV calculation of the training data:

- Temperature and mean radiant temperature:
  $T_{min} = 16\,^{\circ}\text{C}$, $T_{max} = 30\,^{\circ}\text{C}$, $T_{step} = 0.1\text{K}$
- Relative humidity:
  $H_{rel,min} = 30\%$, $H_{rel,max} = 80\%$, $H_{rel,step} = 1\%$
- Air velocity:
  $AV_{min} = 0\frac{m}{s}$, $AV_{max} = 0.7\frac{m}{s}$, $AV_{step} = 0.05\frac{m}{s}$
- Metabolism rate:
  $MET = 1$
- Clothing insulation:
  $CLO = 1$

The parameters for the metabolism rate MET and the clothing insulation factor CLO are kept at the fixed value 1, which is considered as most representative [8]. The calculation results in a dataset of 107864 PMV values.

B. Actions and power consumption

The created dataset of PMV values is subsequently complemented with the different action levels and the respective energy consumption.

The flowchart in figure 4 demonstrates the overall process of training data creation in a general manner. In the case of a PMV value within $[-0.5, 0.5]$, as defined in equation 2, no action is needed since the room climate is already considered comfortable. If the PMV value is below -0.5, the heating will be activated. In cases of a PMV value greater than 0.5, either the ventilation, cooling or a combination of ventilation and cooling will be activated. For complete detail depth please see the flowcharts in figure 5, figure 7 and figure 6.

In some cases, ventilation might be sufficient to lower the PMV value, while other scenarios require both ventilation and cooling. The threshold value of a maximum air ventilation speed of $0.7\frac{m}{s}$ in inside rooms [8] cannot be exceeded. On the condition that the air ventilation threshold value is not exceeded, ventilation is prioritized over cooling for its lower power consumption.

Table I and II demonstrate the potential of regulating the PMV value with ventilation, without any other additional action. Table I shows the different PMV values at a certain temperature and air velocity value. Humidity is fixed at 50% in this example, all other room climate parameters are not considered.

Table II indicates the change of PMV (ΔPMV) in respect to the operating point of each temperature value. It is evident, that ventilation has its greatest impact in negative PMV ranges, where it is unfortunately not of

TABLE I
PMV VALUES AT 50% HUMIDITY

PMV

Air velocity [m/s] \ Temperature [°C]	16	18	20	22	24	26	28	30
0.45	-2.8	-2.2	-1.5	-0.9	-0.3	0.35	0.99	1.6
0.3	-2.5	-2	-1.4	-0.8	-0.2	0.44	1.1	1.7
0.15	-2.2	-1.6	-1.1	-0.5	0.03	0.6	1.2	1.8
0	-2	-1.5	-0.9	-0.4	0.18	0.74	1.4	1.9

TABLE II
ΔPMV VALUES AT 50% HUMIDITY

ΔPMV

Air velocity [m/s] \ Temperature [°C]	16	18	20	22	24	26	28	30
0.45	-0.7	-0.7	-0.6	-0.5	-0.5	-0.4	-0.3	-0.2
0.3	-0.5	-0.5	-0.4	-0.4	-0.4	-0.3	-0.2	-0.2
0.15	-0.2	-0.2	-0.2	-0.2	-0.2	-0.1	-0.1	-0.1
0	0	0	0	0	0	0	0	0

use, since ventilation is only used to lower the PMV value. In positive PMV ranges, ventilation has a minor impact of only $\Delta PMV = [0.3, 0.4]$. Controlling the room climate with only ventilation is therefore only possible in PMV ranges close to optimal room climate.

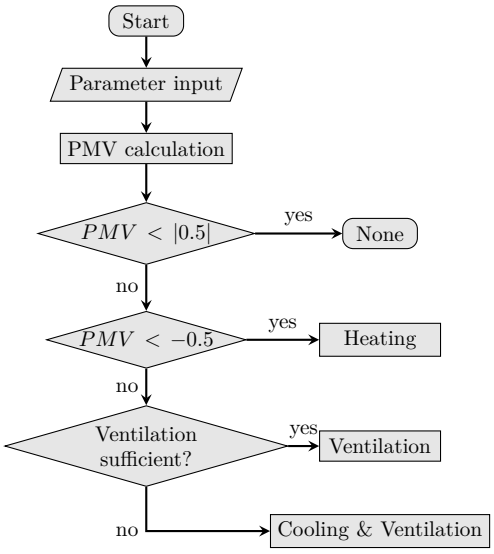

Fig. 4. Data creation

In scenarios where heating is activated, it has to be determined which heating level is needed to raise the PMV to a level that is considered as comfortable room climate, as defined in equation 2.

In cases of PMV values greater than 0.5 and room air velocity lower than $0.7\frac{m}{s}$, the ventilation is activated. It is verified if any of the three ventilation levels are sufficient to reach the PMV interval as defined in equation 2, without violating the maximum air velocity criterion [8]. In cases of a violated criterion, the ventilation is not activated and only the cooling is activated. In figure 6 it is consecutively determined which cooling level is needed to sufficiently cool the room.

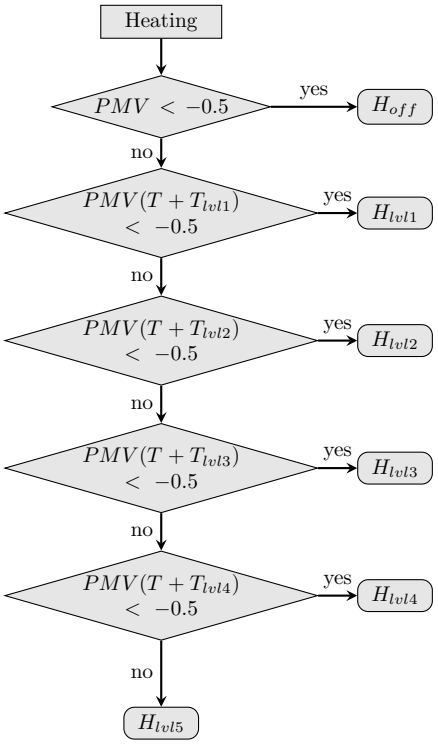

Fig. 5. Heating

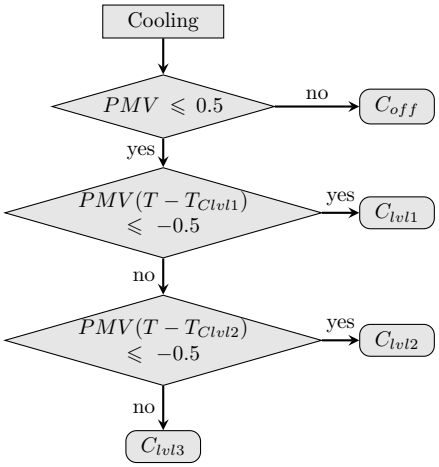

Fig. 6. Cooling

VI. Python Implementation

All functionality, including the calculation of PMV values, the creation of training data and visualisation has been implemented in python v3.7, using the libraries scipy, numpy, pandas, matplotlib and more. The code was tested in Jupyter notebook and Google Colab and versioned in

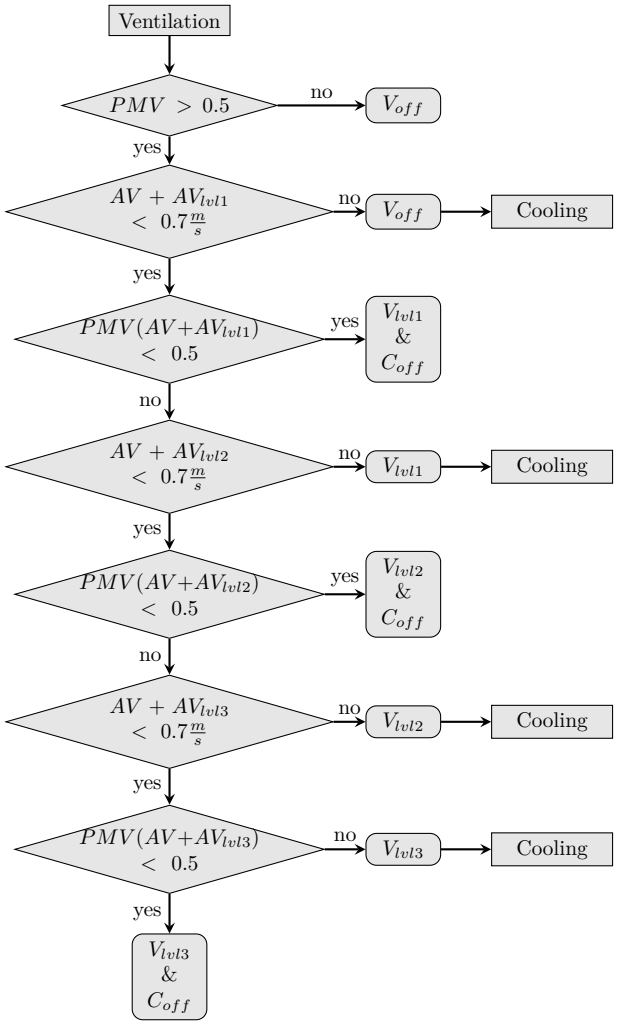

Fig. 7. Ventilation

git. All code is hosted on GitLab [1]. To cover all scenarios of the training data creation, over 500 lines of python code were implemented.

A. Automated calculation of PMV values

The calculation of PMV values is done by automating the Fanger equation function calls in python with several nested loops as described in detail in chapter V-A.

B. Calculation of training data

The training data was calculated by implementing the algorithmic steps described in subsection V-B. In doing so, several functions were defined and implemented. The implementation of the heating function in the following code block serves as an example, as the entire code would exceed the scope of this paper.

```
def heating(data, index, pmv_val, heatval,
    digits_heat, tmp, hum, vel):
    """
    PARAMETERS:
    data:           data frame of PMV values
    index:          current index of dataframe
    pmv_val:        current pmv_value(index)
    heatval:        array with heat conditions
    digits_heat:    nr of digits
    tmp:            current temperature
    hum:            current humidity
    vel:            current velocity
    """
    # Energy levels in KW
    energy_lvl1 = 1.3
    energy_lvl2 = 2.2
    energy_lvl3 = 3.1
    energy_lvl4 = 4.0
    energy_lvl5 = 4.9

    # redundancy for security
    if (pmv_val <= -0.5):

        # ventilation OFF
        data.loc[index, 'PMV_vent']
        = data.loc[index, 'PMV']
        data.loc[index, 'd_PMV_vent'] = 0
        data.loc[index, 'vent_lvl'] = 0

        # cooling OFF
        data.loc[index, 'PMV_cool']
        = data.loc[index, 'PMV_vent']
        data.loc[index, 'd_PMV_cool'] = 0
        data.loc[index, 'cool_lvl'] = 0

        # heating level conditions
        heat_condition_1
        = data[((data['Temperature'] == round(tmp
 +
        heatval[0], digits_heat)) &
        (data['Humidity'] == hum) &
        (data['Air_velocity'] == vel))].PMV

        heat_condition_2
    # ...
        heat_condition_3
    # ...
        heat_condition_4
    # ...
        heat_condition_5

        if (float(heat_condition_1) >= -0.5):
            data.loc[index, 'heat_lvl'] = 1
            data.loc[index, 'PMV_heat']
            = float(heat_condition_1)
            data.loc[index, 'd_PMV_heat']
            = float(heat_condition_1) - pmv_val
            data.loc[index, 'heat_KW']
            = energy_lvl1

        elif (float(heat_condition_2) >= -0.5):
    # ...
        elif (float(heat_condition_3) >= -0.5):
    # ...
        elif (float(heat_condition_4) >= -0.5):
    # ...
        else:
        # ...
        return True, float(data.loc
        [index, 'PMV_heat'])
    else:
        return False, pmv_val
```

Listing 1. Function example

[1] https://gitlab.com/julhaas91/pmv-air-conditioning-control/

An extract of the training data is presented in table

III below. The whole dataset is hosted on https://gitlab.com/julhaas91/pmv-air-conditioning-control/ along with the full implementation.

TABLE III
Training data example

id	T[°C]	H_{rel}[%]	$AV[\frac{m}{s}]$	PMV_{in}	V_{lvl}	C_{lvl}	H_{lvl}	ΔPMV_{out}	PMV_{out}	PPD_{out}	P[kW]
1	16	30	0	-2.12	0	0	4	1.9	-0.22	6	4
2	16	30	0.05	-2.12	0	0	4	1.9	-0.22	6	4
3	16	30	0.1	-2.12	0	0	4	1.85	-0.27	6.51	4
4	16	30	0.15	-2.28	0	0	4	1.9	-0.38	8.01	4
5	16	30	0.2	-2.43	0	0	4	1.97	-0.46	9.42	4
...	...	...	...	...	...	...	...	...	...	...	...
69425	25	68	0.2	0.39	0	0	0	0	0.39	8.17	0
69426	25	68	0.25	0.34	0	0	0	0	0.34	7.4	0
69427	25	68	0.3	0.29	0	0	0	0	0.29	6.75	0
69428	25	68	0.35	0.25	0	0	0	0	0.25	6.3	0
69429	25	68	0.4	0.21	0	0	0	0	0.21	5.91	0
...	...	...	...	...	...	...	...	...	...	...	...
87532	27.4	51	0.3	0.89	2	1	0	-0.5	0.39	8.17	1.1
87533	27.4	51	0.35	0.86	2	1	0	-0.51	0.35	7.55	1.1
87534	27.4	51	0.4	0.83	2	1	0	-0.51	0.32	7.13	1.1
87535	27.4	51	0.45	0.81	1	1	0	-0.52	0.29	6.75	1
87536	27.4	51	0.5	0.79	1	1	0	-0.52	0.27	6.51	1
...	...	...	...	...	...	...	...	...	...	...	...
107412	30	50	0.55	1.62	1	2	0	-1.42	0.2	5.83	2.55
107413	30	50	0.6	1.61	0	2	0	-1.43	0.18	5.67	2.45
107414	30	50	0.65	1.6	0	2	0	-1.44	0.16	5.53	2.45
107415	30	50	0.7	1.59	0	2	0	-1.45	0.14	5.41	2.45
107416	30	51	0	1.88	3	3	0	-2	-0.12	5.3	4.3

VII. Conclusion

The climate control approach based on the empirical PMV value, rather than depending solely on a temperature value feedback, yields the advantage of a more representative control result. Calculating the current PMV value with a feedforward network based on the stated input parameters, and consecutively using that value to control the room climate with an AI approach, bares great advantage over the classical climate control approach. An software agent trained with a reinforcement learning algorithm is constantly improving its performance and is able to adapt to new room scenarios within a reasonable period of time.

During the course of this project, the focus was strongly on the creation of correct and extensive training data. The implementation and training of a reinforcement learning algorithm is not implemented yet. However, the presence of high quality training data is of significant importance to enable a satisfying algorithm performance.

As of now, the training data for offline training has successfully been created in accordance to the power levels of the Samsung R32 room climate device [9]. The dataset of 107864 records has been successfully used by Nils Bitzer to train a feedforward neural network to simulate the control of the room climate with a accuracy of up to 93.3% [18].

The 107864 records of training data that have been created in the course of this project offer a good prerequisite for the implementation and training of an artificial intelligence algorithm in a future project. One conceivable approach is to implement an A2C reinforcement learning algorithm that regulates the room climate based on the reward of minimum energy consumption. It would be sensible to test the performance of the implemented agent in the Google AI Gym[2] before running it on hardware. Ultimately, the algorithm has to be implemented on a microcontroller that will be connected to the Samsung R32 room climate device and the sensor board of Wajidi Araar [17].

In future projects, several questions could be of interest to determine the influence of ventilation on heating dispersion in inside scenarios. A question could be: *Does it make sense to turn on the ventilation in heating scenarios?* Additionally, instead of keeping the MET and CLO at a fixed value, one could train a convolutional neural network to detect clothing and activity of people inside the room. The output signal of the convolutional network could the be fed back to the feedforward network that is calculation the PMV value for the climate control. This could be especially useful for control scenarios in public buildings such as libraries or lecture halls.

Acknowledgment

I want to thank Prof. Dr. Kohlhof for the constant constructive feedback during the time of my research. Furthermore, I want to thank Nils Bitzer for the successful cooperation throughout the project.

References

[1] G. E. Moore: "Cramming more components onto integrated circuits." In: Electronics. 19, Nr. 3, 1965, S.114–117 (ftp://download.intel.com/research/silicon/moorespaper.pdf)

[2] G. James, D. Witten, T. Hastie, and R. Tibshirani, "An Introduction to Statistical Learning," Springer Science+Business Media New York, 2013.

[3] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," Massachusetts Institute of Technology, 2016.

[4] P. O. Fanger, 'Thermal Comfort: Analysis and Applications in Environmental Engineering" Mc. Graw-Hill Book company, 1972.

[5] N. Große, W. Schorn: "Taschenbuch der praktischen Regelungstechnik", Fachbuchverlag Leipzig im Carl Hanser Verlag, 2006.

[6] M. Wiering, M. v. Otterlo: "Reinforcement Learning: State-of-the-Art, Springer, 2012.

[7] ANSI/ASHRAE Standard 55-2017: "Thermal Environmental Conditions for Human Occupancy", 2017.

[8] DIN EN ISO 7730: "Ergonomics of the thermal environment - Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices and local thermal comfort criteria (ISO 7730:2005)", DIN Deutsches Institut für Normung e.V., 2005.

[9] Datasheet: "Samsung AR12RXFPEWQN/EU—R32—3,5 kW", www.samsung.com, accessed on 29.04.2020.

[10] D. E. Rumelhart, G.E. Hinton, R. J. Williams, "Learning representations by back-propagating errors", University of California, 1986.

[11] S. Hochreiter, J. Schmidhuber, "Long Short-Term Memory", Technische Universität München, 1998.

[12] S. Hochreiter, Y.Bengio, P. Fasconi, J. Schmidhuber, "Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies", IEEE Press, 2001.

[13] J. Hochreiter, "Diplomarbeit: Untersuchungen zu dynamischen neuronalen Netzen", Technische Universität München, 1991.

[14] R. Bellman, " A Markovian Decision Process", Journal of Mathematics and Mechanics 6, 1957.

[2] https://gym.openai.com/

[15] AlphaGo: “Mastering the ancient game Go with Machine Learning,” in: blogspot.com, accessed on 15.09.2020.
[16] M. Abdi, K. Kohlhof, “Smart air conditioning according ISO 7730 by neural networks,” University of Applied Sciences Cologne, 2018.
[17] W. Araar, T. Hofacker, K. Kohlhof, “Developing an IoT-based control system for existing air conditioner using MEMS,” University of Applied Sciences Cologne, 2018.
[18] N. Bitzer, K. Kohlhof, “Control of air conditioning device by feed forward neural networks,” University of Applied Sciences Cologne, 2020.
[19] M. Slippers, C. Dick, “Using AI for Closed Loop Control in a Buck Converter Application,” University of Applied Sciences Cologne, 2020.
[20] L. Weng: “Lil’Log,” in:https://lilianweng.github.io/lil-log/, accessed on 09.07.2020.
[21] S.Attahajariyakul, T. Leephakpreeda, “Neural computing thermal comfort index for HVAC systems”, Thammasat University, Elsevier Ltd., 2005.