# Development of a digital twin of the DLR EDEN ISS greenhouse for data enabled analysis

**Entwicklung eines digitalen Zwillings des DLR EDEN ISS Gewächshauses für die datengetriebene Analyse**

Bachelor thesis by Sven Steinert, Matrikelnummer: 2708229

Date of submission: March 1, 2021

1. Review: Prof. Dr.-Ing. Reiner Anderl
2. Review: Dr.-Ing. Paul Zabel
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

# Development of a digital twin of the DLR EDEN ISS greenhouse for data enabled analysis

Bachelorarbeit
Sven Steinert | 2708229
Fachgebiet Datenverarbeitung in der Konstruktion

TECHNISCHE UNIVERSITÄT DARMSTADT


Sven Steinert
Matrikelnr.: 2708229
Studiengang: B.Sc. Wirtschaftsingenieurwesen Maschinenbau

Bachelorarbeit
Thema: Development of a digital twin of the DLR EDEN ISS greenhouse for data enabled analysis

Eingereicht: 01.03.2020

Betreuer: Martin Wende, M. Sc.

Prof. Dr.-Ing. Reiner Anderl
Fachgebiet Datenverarbeitung in der Konstruktion
Fachbereich Maschinenbau
Technische Universität Darmstadt
Hochschulstraße 1
64289 Darmstadt

# Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

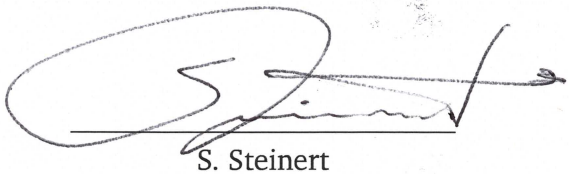
Hiermit versichere ich, Sven Steinert, Matrikelnummer: 2708229, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 1. März 2021



S. Steinert

Bachelor Thesis

für

Sven, Steinert (2708229)

TECHNISCHE UNIVERSITÄT DARMSTADT

# Development of a digital twin of the DLR EDEN ISS greenhouse for data enabled analysis

*Entwicklung eines digitalen Zwillings des DLR EDEN ISS Gewächshauses für die datengetriebene Analyse*

**Background**

The Research Initiative EDEN (Evolution & Design of Environmentally closed Nutrition-Sources) focuses on Bio-regenerative Life Support Systems (BLSS), especially greenhouse modules, and how these technologies can be integrated in future space habitats. The mobile test facility EDEN ISS, which is operated at Neumayer III Station in Antarctica, performed the ground demonstration as a semi- closed loop system and has laid the foundation for EDEN NEXT Gen, the development of a deployable greenhouse module for Moon and Mars. In action, the food supply of planetary outposts would be secured, for an important step towards independence and thus towards sustainability of life beyond Earth.

**Task Formulation**

The goal is to create a digital twin of the EDEN ISS module to predict biomass output. The twin should simulate the entire system and be able to predict data into the near future. The entirety of the module can be divided into subsystems which should be described independently. The real-world data set has to be selected and worked up for processing. The formatted data and its standardization should also serve as a foundation for future documentation and eventually AI implementation. For validating the accuracy of the simulation, sensor and general logging data from the real-world system EDEN ISS in Antarctica are processed and interacted with.

The content of the thesis is composed as follows:

- Literature review of Digital Twins and and their application on bio-regenerative life support systems
- Analysis of the EDEN ISS system architecture
- Definition of requirements and use-cases of the EDEN ISS Digital Twin
- Development of the EDEN ISS digital twin concept
- Development of relevant subsystem models
- Preparation of the real-world data and rules for standardization
- Integration of subsystems and real-world data into a digital twin
- Testing of the EDEN ISS digital twin
- Documentation and presentation of the results

Date of issue: 09.10.2020
Supervisor TU Darmstadt: Martin Wende, M.Sc.
Supervisor DLR: Dr.-Ing. Paul Zabel

Prof. Dr.-Ing. R. Anderl

Fachgebiet Datenverarbeitung in der Konstruktion

Department of Computer Integrated Design

Prof. Dr.-Ing. Reiner Anderl

Otto-Berndt-Str. 2
64287 Darmstadt

Tel. +49 6151 16-21791
Fax +49 6151 16-21793
anderl@dik.tu-darmstadt.de

DLR:
Institut für Raumfahrtsysteme
Robert-Hooke-Str. 7
28359 Bremen

Dr.-Ing. Paul Zabel
Tel. +49 421 24420 1273
Paul.Zabel@dlr.de

Datum
09.10.2020

## Abstract

Digital Twins as a virtual entity of a physical system open up a variety of new opportunities. Their development on the other hand can be very challenging especially for complex systems. The EDEN ISS Mobile Test Facility with its Future Exploration Greenhouse belongs to those more sophisticated setups.

During its operations since 2018 a great volume of data is collected which is including more than 200 sensors and a wide crew documentation. In this paper the development of a Digital Twin is approached by utilizing this data for machine learning.

Specifically, a Neural Network is developed that recreates the same changes on the features of the dataset as the physical system did in the past. In the event of success the Neural Network is holding a general representation of the physical system and also future or virtual changes can be simulated.

In particular, time series forecasting is chosen to perform the prediction making by taking the past 24 hours as input and returning the following 24 hours as an output, both in a 5 minute step resolution.

The development starts by the preparation of the dataset, where individual data tables are merged, missing data is interpolated and constant or corrupted features are identified and removed.

The general structure of the Digital Twin is divided into 3 sub-models to address categorical differences of the features. The first distinction is made according to the documentation style between the sensor data and the crew data, which is containing the information about the plants. A further differentiation is done inside the sensor data between time controlled variables and environment controlled variables whether the values are regulated by a time schedule or not. Those 3 sub-models are named ”time controlled model”, ”environment controlled model” and ”plant model”.

The identification of time controlled variables is carried out by sorting over the error term of a model that assumes a 24 hour periodic repetition. The remaining ones from this identification make up the environment controlled variables.

The first sub-model, the time controlled model is developed by applying a whole group of models on this subset and selecting the model with the highest precision by the error term they created. The previous mentioned repetition model was selected which offered a very small error in prediction.

The second sub-model, the environment controlled model is generally following the same approach. A smaller set of Neural Networks is applied on the new structure and again the best performing model is chosen for further investigation. The LSTM-model is offering the lowest error in prediction here and is expanded and regulated for a second training. Besides further improvement, this model is still ending up on a high level of error when averaged over all features. These errors in prediction vary greatly on the individual variables and range from decent precision up to a deterioration over the baseline.

The development of the third and final sub-model, the plant model resulted to be unsuccessful. The insufficient amount and resolution of the plant data is missing the requirements for the intended objective of bringing the plants biomass in correlation with system values.

In conclusion, the development of a Digital Twin for the EDEN ISS system is prototypically achieved, while the approach of building a Digital Twin without the system being designed for it encountered heavy limitations.

# Contents

# List of Figures

# List of Tables

## Abbreviation

**AMS** Atmosphere Management System

**DT** Digital Twin

**FEG** Future Exploration Greenhouse

**ILS** Illumination System

**LSTM** Long Short-Term Memory

**MAE** Mean Absolute Error

**MSE** Mean Squared Error

**MTF** Mobile Test Facility

**NDS** Nutrient Delivery System

**NN** Neural Network

**PCDS** Power Control and Distribution System

**TCS** Thermal Control System

# 1. Introduction

## 1.1. Mission Background

The EDEN ISS project started out with its kick-off in March 2015, setting the goal to develop a Mobile Test Facility (MTF) which should serve as a plant cultivation testsystem for on-board operations on the International Space Station and a Future Exploration Greenhouse (FEG) for extraterrestrial planetary applications such as for Moon or Mars. [1, p. 2] In order to do so, the research and application of those key technologies had to be done, which is the main focus of EDEN ISS. The history of events as visualized in the timeline on Figure 1.1 is containing 3 main phases: design, building and experiment.

During the design phase, the technological implementation for usage in the Antarctica was done. Antarctica was chosen because of its very small number of micro-organisms requiring an isolated environment for cultivation, and also because of its general similarity to space missions and the existing infrastructure.

In the building phase, assembly took place in Bremen at a DLR site first, followed up with some initial tests before deployment to the Neumayer-Station III. [2, p. 12] Together with the container shipment a crew was selected to carry out the deployment mission that was setting up the system for usage in the following season 1. [3, p. 8]

Having arrived in the experiment phase, multiple seasons, each lasting 8 months from March to November during the antarctic winter have been accomplished where data was created and tweaks got applyed over the years. [4, p. 4] The most recent of those seasons (season 4 in 2021) is going to be carried out in a collaboration by NASA and DLR. [5, p. 6]

While EDEN ISS has created enough substance for lessons learned, the design phase of EDEN NEXT Gen is getting started. EDEN NEXT Gens' focus is the adaptation for space transit and extraterrestrial deployment. This being a very big step in development, the timeframe is aiming for a mature concept in 2025 and a mission application at around 2030. [5, p. 8] Meanwhile, operations on EDEN ISS are planned to be ongoing until atleast 2023. [5, p. 6]

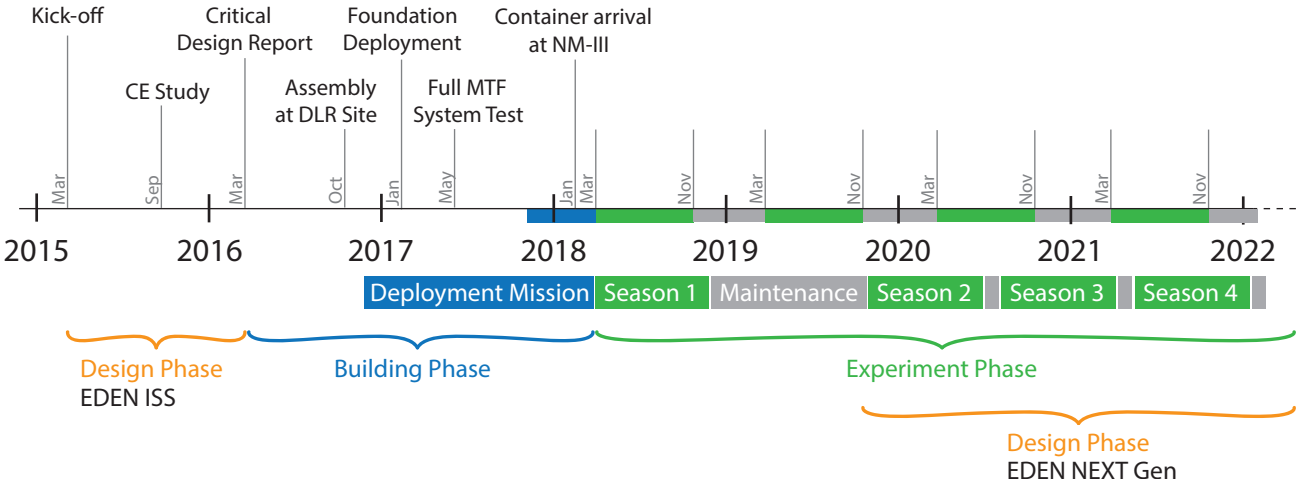



Figure 1.1: Timeline of the history of the EDEN project and its progress


Figure 1.2: The EDEN ISS MTF deployed 400m next to Neumayer Station III, Antarctica [4, p. 2]

## 1.2. Impact on the development of a digital twin

The fact that the EDEN ISS project is in a very mature state at the time of writing this paper has several advantages. First, the technological specifications of the whole system are set and the system is already constructed and operated. Secondly, a large amount of run-time data was already created during the growth phase in the Antarctic winter seasons which count up to 3 completed currently. Both of those points are important elements for the development of a Digital Twin (DT). To introduce a DT in the current project phase seems therefore appropriate. Even though EDEN ISS was designed without planning to use a DT, the multiple seasons and long run-times hold the opportunity to compensate for incomplete data by advantageous selection.

# 2. Literature Review

## 2.1. The Concept of Digital Twins

Due to the fairly new existence of the idea of a DT, a consistent definition of itself has yet to be established. The Industrial Internet Consortium which is an industry leading organisation with the goal of cross-sector interoperability and interconnectivity defines a DT as followed [6, p. 5]

> *Digital Twin = "Digital representation, sufficient to meet the requirements of a set of use cases."*
> *Digital Representation = "Information that represents attributes and behaviors of an entity"*

When such a DT is then linked to its physical entity and a bidirectional link is established, this setup of connections can form a regulating circle as seen on Figure 2.1 where the DT is taking the role of the central control unit in the system and the physical entity the role of the actor and sensor simultaneously.

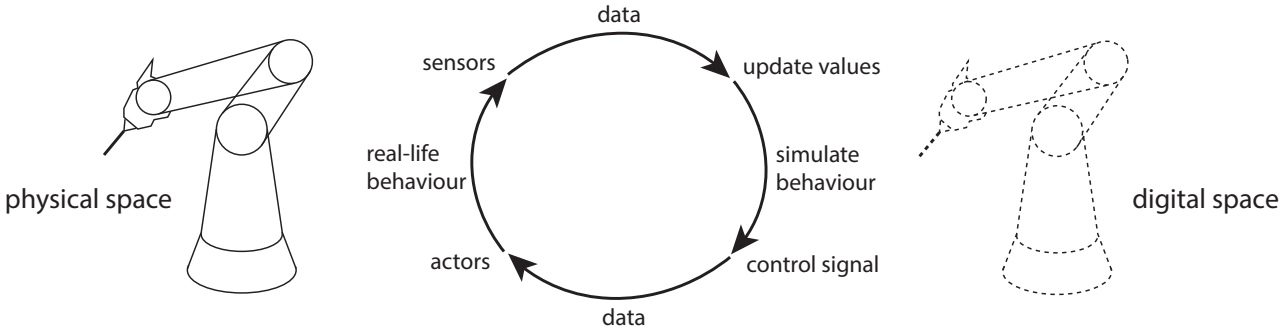



Figure 2.1: Bidirectional linked digital twin in a regulating stage

The feedback from the simulation however doesn't necessarily has to be actor-data. Because DT's can simulate any custom scenario, they can provide valuable information for product development. [7, p. 8]

In contrast, other definitions no longer refer a DT to the digital representation but rather name the whole concept the DT. [8, p. 1017] A further way to classify such systems is the separation by the degree of integration of their connections as seen in Figure 2.2. Therefore, a DT from a second standpoint is containing both a physical- and a digital model as well as their automated connections. [9, p. 2]

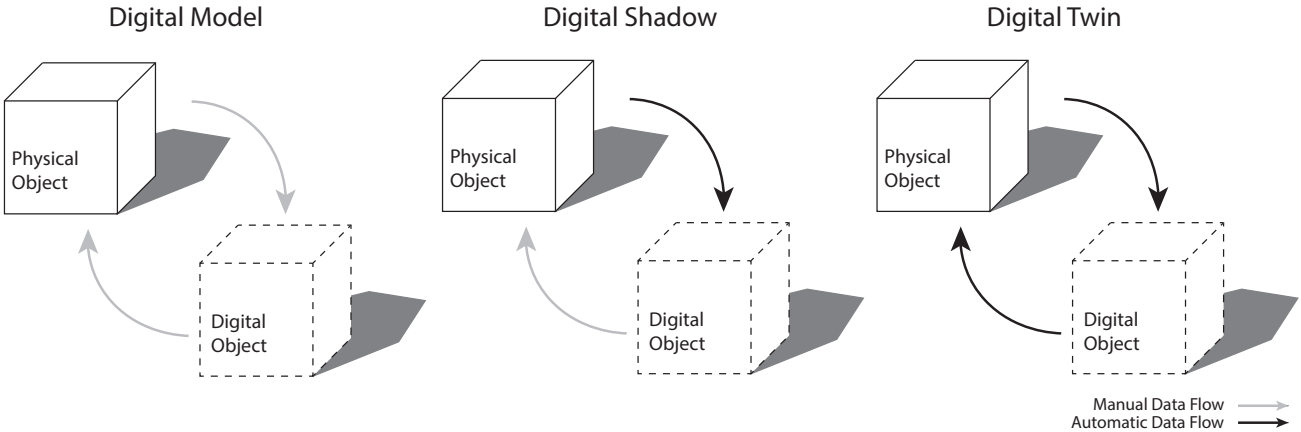



Figure 2.2: Differentiation of digital twin systems by the degree of integration [8, p. 1017]

## 2.2. Transfer to EDEN ISS

In terms of the second definition of a DT, our system will end up being either a digital model or a digital shadow instead, as the automated data flow back to the physical system will not occur in this work. Which one of those it will be depends on whether the live data is linked to the model or not, whereupon the data flow towards the digital system is either automated or not. Both however are a subset of a DT. In this paper we are following the first definition and continue to refer to a DT for its similar behaviour and its use cases.

The available points that we can access from the physical system are the archived sensor data, the crew log data, the system architecture and the live data. The different ways a digital counterpart could be developed out of those informations are visualized in Figure 2.3.

Here, a distinction can be made between 2 main methods for development, either the manual network construction from the system architecture or the neural network construction by regression of the data set. Both have advantages and disadvantages over the other, which will be discussed and decided later in chapter 4.

Regardless if manual or neural network, the general usage for our DT will be the same. In testing phase the input into the digital counterpart will be made by the archived data set that is initializing the parameters for the simulation. The outcome of our network is then being validated with the real values.

When the testing phase is finished, the initialization could be done by the live data which will make us able to take the output of our network as a prediction of the physical system. With this integration enabled, a digital shadow by definition is achieved.

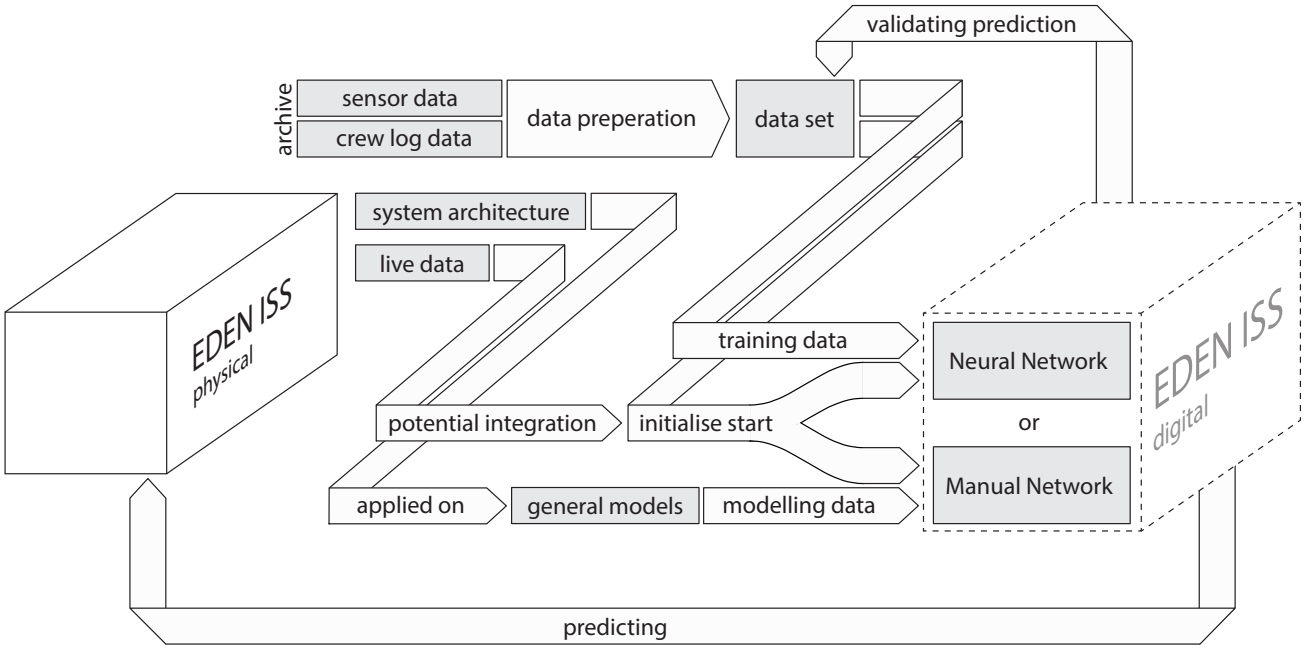



Figure 2.3: EDEN ISS digital twin development opportunities schematic

## 2.3. Manual Networks - Simulation of life support systems

One recent example of existing habitat simulations is the ”Virtual Habitat”, a work by the Institute of Astronautics at the TU Munich in 2015[10]. Here, the traditional method of a simulation is chosen, where subsystems are defined by models manually and get interconnected into a large network. All models are implemented in MATLAB© and aim to be modular in order to be adaptable for similar systems [10, p. 2706]. The top level structure with its model descriptions can be seen in Figure 2.4 followed up with the flow diagram and the model interconnections in Figure 2.5.

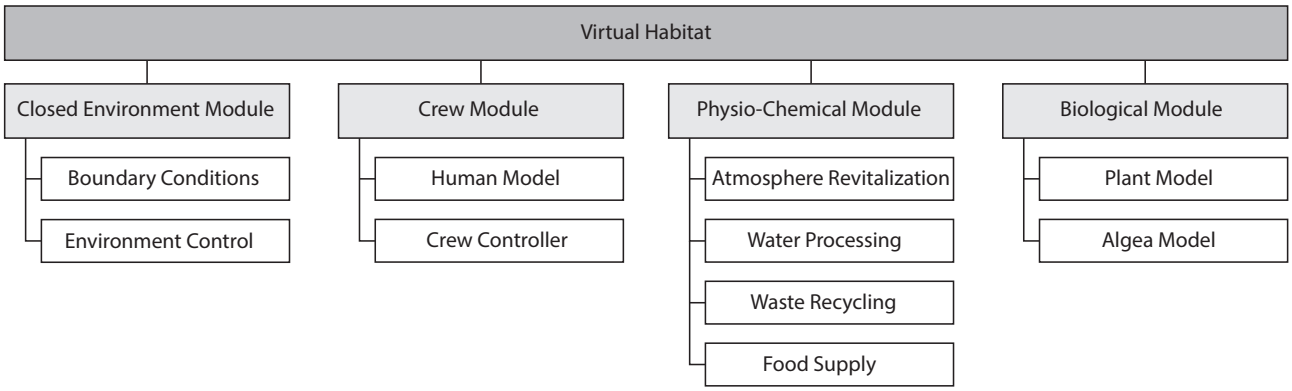



Figure 2.4: Top-level overview of the Virtual Habitat [10, p. 2685]

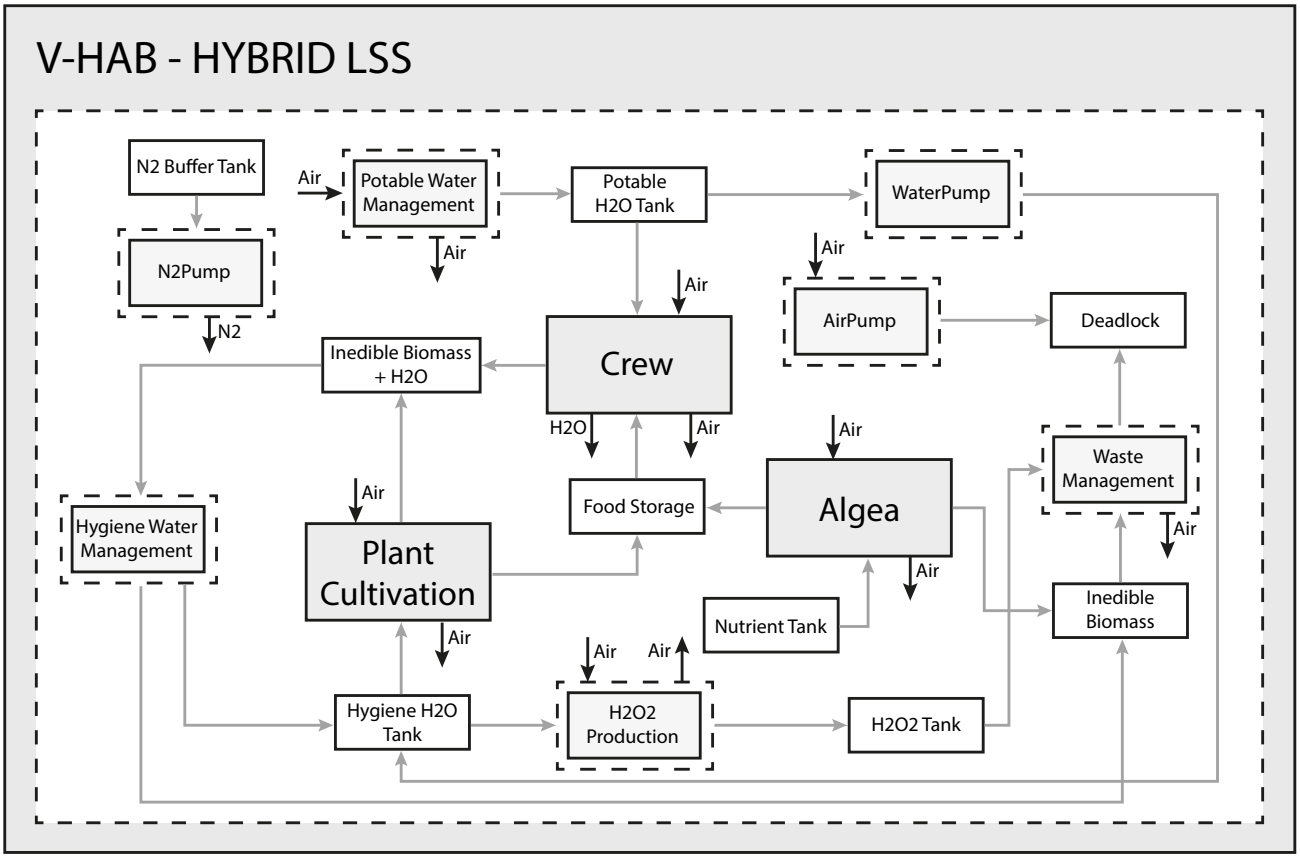



Figure 2.5: Flow diagram of the Virtual Habitat [10, p. 2701]

To perform an adaptation on the EDEN ISS MTF, the system architecture has to be aligned with the supplied reference model. Every subsystem that is not present or not wanted in our simulation, as the Algae- or the Crew-Model has to be removed and any missing subsystems need to be added. Continued by adapting the values inside a model in order to represent the correct number and technical specifications of the EDEN ISS greenhouse parts. All model assumptions need to be revisited as well in order to check transferability. Finally all deviating system interconnections need to be updated.

Therefore the ”Virtual Habitat” is able to serve as a foundation for manual modelling of the EDEN ISS MTF but not as an simply duplicable solution.

In order to make use of any sensor data or real world readings from the MTF a validating system is needed to supplement the information for model building, that is currently containing only the information of the system architecture.

Such a computation of an error term between the prediction made by the simulation and the actual data brings in an evaluation of performance. In order to further increase this performance, optimizing iterations have to be performed. Changes in the model resulting in a new error term that either discards or accepts the changes in direction of improvement.

This can be done manually or with the usage of algorithms. Which also raises the question of which values in our manual network can be changed for optimisation and which cannot.

## 2.4. Neural Networks - Machine Learning

We are going to use a simple example to engage with the general working properties of a Neural Network (NN). Here in Figure 2.6 the basic structure can be divided into 3 main parts that flow from the left to right: the input layer, the hidden layers and the output layer. In the example in Figure 2.6 there is 1 hidden layer instead of multiple, an input size of 3, a hidden layer size of 3 and an output size of 2. The input- and output layer are the embedding points to the rest of our system and can be understood as a functions argument and its result. To stick to this analogy, the function itself is the NN in its narrower sense, this is represented by the hidden layers and its connections. Every layer consists of a number of so called neurons, which can be imagined as a variable that holds a real number greater than 0. Zero would represent no activation and 1 or $\infty$ a full activation of this neuron depending on the domain of the activation function. How strong the activation of this neuron is influencing the next layer neurons depends on what is called weights, biases and the activation function. [11]

Since a neuron has multiple connections terminating from the previous layer, a weighted sum is performed by multiplying a weight to each of the source neurons. Next, a bias gets subtracted which serves as a threshold the sum has to reach in order to have a notable impact. Finally, this real number is plugged into the activation function which converts the range of real numbers between 0 and 1 in case of the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and between 0 and $\infty$ for the rectified linear unit $ReLu(x) = max(0, x)$ .

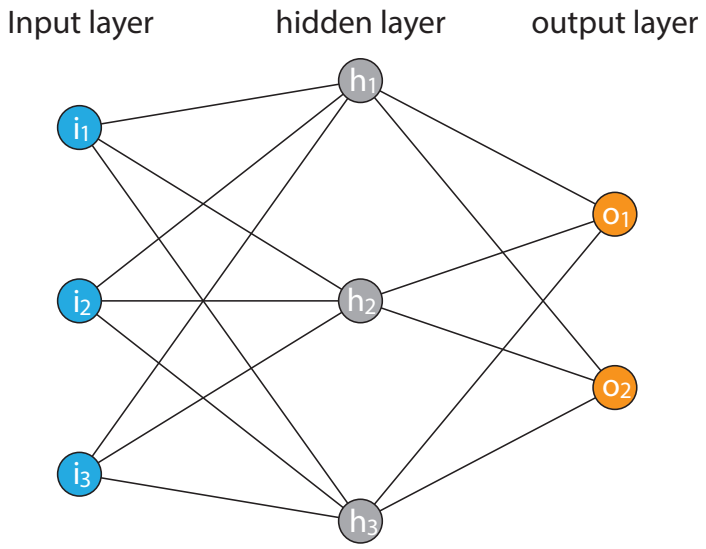



Figure 2.6: Visualized structure of a simple dense Neural Network

$$\sigma\left(\begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{bmatrix} - \begin{bmatrix} Bias_1 \\ Bias_2 \\ Bias_3 \end{bmatrix}\right) = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

$$\sigma\left(\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} \begin{bmatrix} \widehat{w}_{1,1} & \widehat{w}_{1,2} & \widehat{w}_{1,3} \\ \widehat{w}_{2,1} & \widehat{w}_{2,2} & \widehat{w}_{2,3} \end{bmatrix} - \begin{bmatrix} \hat{Bias}_1 \\ \hat{Bias}_2 \end{bmatrix}\right) = \begin{bmatrix} o_1 \\ o_2 \end{bmatrix}$$

$$\begin{aligned} i_i &\in [0,1] \\ w_i &\in \mathbb{R} \\ Bias_i &\in \mathbb{R} \\ h_i &\in [0,1] \\ \widehat{w}_i &\in \mathbb{R} \\ \hat{Bias}_i &\in \mathbb{R} \\ o_i &\in [0,1] \end{aligned}$$

The full computation of our example structure can be seen above in matrix form, starting with the input vector $\vec{i}$ and ending with the output vector $\vec{o}$. In this case with scaled inputs and the sigmoid activation function for both layer conversions.

The way such a NN can be trained starts with initializing its weights and biases randomly. Now, the NN is able to produce an output with a given input from our training data. To measure the performance of the network a cost function is introduced, which is a value for the difference between what the network predicted versus the correct value it should have predicted. This can be computed for example with the Mean Squared Error (MSE) along the output averaged over all its predictions of the training dataset. Generally, this cost function takes all weights and biases as an input with all the training data as parameters and outputs a single number, the cost. The cost function for our example network with averaged MSE is shown next.

$$C(w_1, ..., w_{15}, Bias_1, ..., Bias_5) = \frac{1}{n}\sum_{j=1}^{n}(\vec{o}_j^{\,*} - \vec{o}_j)^2 \qquad \begin{array}{l} n = \text{size of the training data} \\ \vec{o}_j^{\,*} = \text{correct output} \end{array}$$

Our interest is now to minimize this cost function in order to fit our training data better. This is approached by optimization algorithms.

One of those optimizers is gradient descent. Since the cost function is a multi variable function the gradient from a point on it tells us which vector has the steepest ascent. Therefore, to get the highest decrease on its cost value the negative of this vector has to be taken. The direction of change for our example network would thus be $-\nabla C(\vec{W})$ with $\vec{W} = (w_1, ..., w_{15}, Bias_1, ..., Bias_5)$ .

A more complex optimizer is Adam which stands for Adaptive Moment Estimation, which has the idea of more carefully approaching minima by carrying a momentum of first and second order as well as an exponentially decaying average of the past gradients.[12] This ultimately leads to it often being the most efficient optimiser of all currently available and is therefore the standard choice.

To make a prediction is known under forward usage of the NN and to determine the cost and changing the weights and biases upon it is called backwards usage. One of this iteration of forward and backward is done with multiple training examples as once which is called a batch. This batch size has an impact on the learning behaviour where small batches tend to break out faster but use less memory than big batches. After the full training data was run in batches, an epoch is completed. Following this pattern, a minimum is reached at some point. Whether it is a local or the global minimum is not feasible to determine for now.

Having reached a minimum in our cost function will introduce us to a next important step, showing why dataset separation is required here. A perfectly fitted network on training data is useless if it doesn't perform well on new data that it never has seen before. Such an ability means how well it generalizes this problem. A valid approach for higher generalisation is to split the dataset into 3 parts: training data, validation data and testing data.

The training dataset will be used for the adjustment of weights and biases towards a minimum. The validation dataset is introduced afterwards to rate the network for its current generalisation ability and epochs are judged upon it. The testing dataset was neither used to train the weights and biases nor has the network been optimized on its performance on it, therefore it serves best for a final evaluation. Common data separation rates are 70% training, 20% validation and 10% testing.

If a model improves its performance on training data but decreases its performance on validation data, this is a sign of overfitting. A common regularisation method to attack this problem is called dropout, where a percentage of the network neurons are disabled while training is performed.[13] Further regularisation methods are L1 (Lasso Regression) and L2 (Ridge Regression) which both are adding a lambda parametrised term to the cost function that is acting as a penalty for a high number of significant weights leading to a more simple model.[14] This strategy is also known as weight decay.

The transfer to the application on digital twins is made by achieving the attribute of similar behaviour. Because a DT is given when the digital counterpart is acting identical as the physical system would do, a DT is also a perfect forecasting model. Such a forecasting action can be performed manually by a given function or by a NN that is trying to find its own function through optimisation.

In reverse this implies, when all system attributes are perfectly predicted by a NN, the NN is a digital twin. Therefore is the development of a system related NN also a development of a DT in its forecasting abilities.

This task of time series forecasting brings in the question how the structure of the NN should be composed, in order to represent physical interconnections.

First, a model selection can be pinned down by the general properties about the predictability of the system. When the last state of a system is theoretically enough to compute the following state accurately, the Markov property is given and a single time step input is a suiting input shape. For a physical system this is usually not the case since many real life values carry some kind of momentum that require the rate of change or multiple time steps as an input to be picked up.

Regarding the internal structure of the here introduced feedforward NN, there is always a sequential structure of several layers, where one layer feeds into the next.

A network where data flows additionally from one layer to its own or to the previous layer and therefore creating some kind of a loop, is described as a Recurrent Neural Network (RNN). For example can a Long Short-Term Memory (LSTM) cell carry a state along multiple calculation iterations while processing data sequentially.

The last structure that is presented here is the Convolutional Neural Network (CNN) where a matrix form can be used as an input rather than a flattened vector. This is highly beneficial for example to analyse images since its recognition becomes space invariant which is not the case for a flattened vector. The input matrix gets then filtered in multiple ways which represent the core of the convolution since they stack up multiple frames in themselves.

There are many frameworks and libraries for machine learning that could be used for implementation such as Tensorflow, PyTorch, Keras or Scikit-learn. Due to the excellent documentation that even provides a base structure for time series forecasting with multi-step-inputs we are choosing Tensorflow with Python as its native programming language. [15]

# 3. EDEN ISS System Architecture

## 3.1. Subsystems

The EDEN ISS MTF can be divided into 2 main parts, the service section (Figure 3.1) that holds most of the maintaining electronics as well as a working space for the crew and the FEG itself (Figure 3.2), where the plants are grown in a vertical tray setup with 4 racks on both the left (L) and the right (R) side.

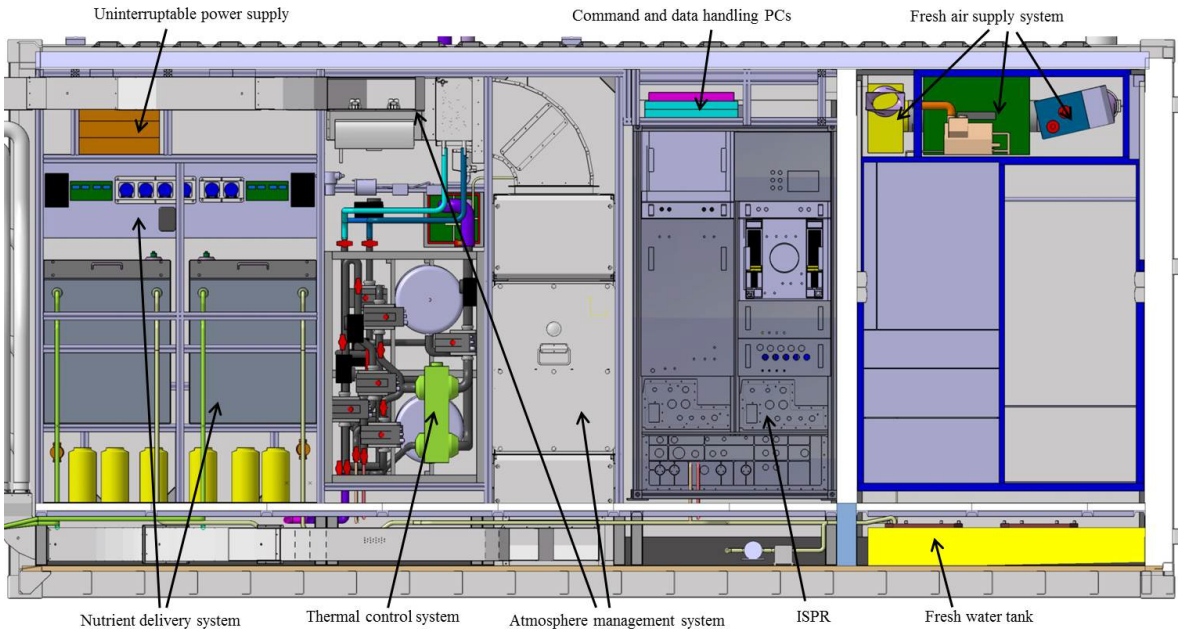



Figure 3.1: EDEN ISS Service Section cut view looking on the right side [16, p. 4]

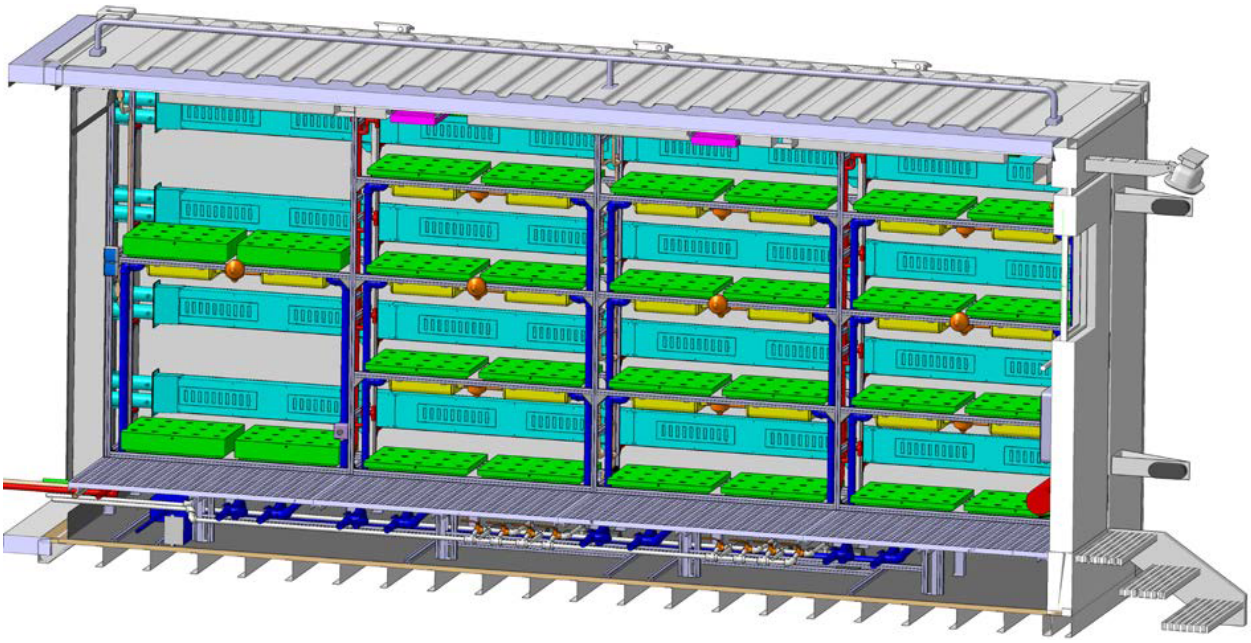


Figure 3.2: EDEN ISS FEG cut view looking on the left side [17, p. 72]

For the general setup, the MTF stands on a foundation above the ground that supplies protection from storms so the system can not get buried in snow. Inside, the plants are grown without soil in boxes where their roots are getting sprayed by a nutrition solution. The system is automatically regulating target parameters as well as handling the events as watering and illumination.

A small crew that changes for every season is housed at the Neumayer III station and is maintaining the FEG in clean up, installation, sowing and harvesting. Together with the rest of the team back at the DLR site in Bremen operations are carried out.

The main technical subsystems are categorized into Thermal Control System (TCS), Power Control and Distribution System (PCDS), Illumination System (ILS), Atmosphere Management System (AMS) and Nutrient Delivery System (NDS).

In the previous illustration from the cut view of the service section in Figure 3.1 the NDS, TCS and AMS are labelled in their installed place.

Besides providing power, the PCDS is also connecting to the control network, where the MTF is linked to the Neumayer III station and from there over satellite to the system control center of DLR in Bremen.

The ILS is build out of custom lamp boxes that each hold 4 LEDs which values will later be found in our ILS dataset. Those are a 450 nm (BLUE), 660 nm (RED), 735 nm (FAR-RED) and a broadband white 5700K LED [18, p. 6]. The LEDs are mounted in a water-cooled containment, one for each tray whose arrangement can be seen in Figure 3.3.

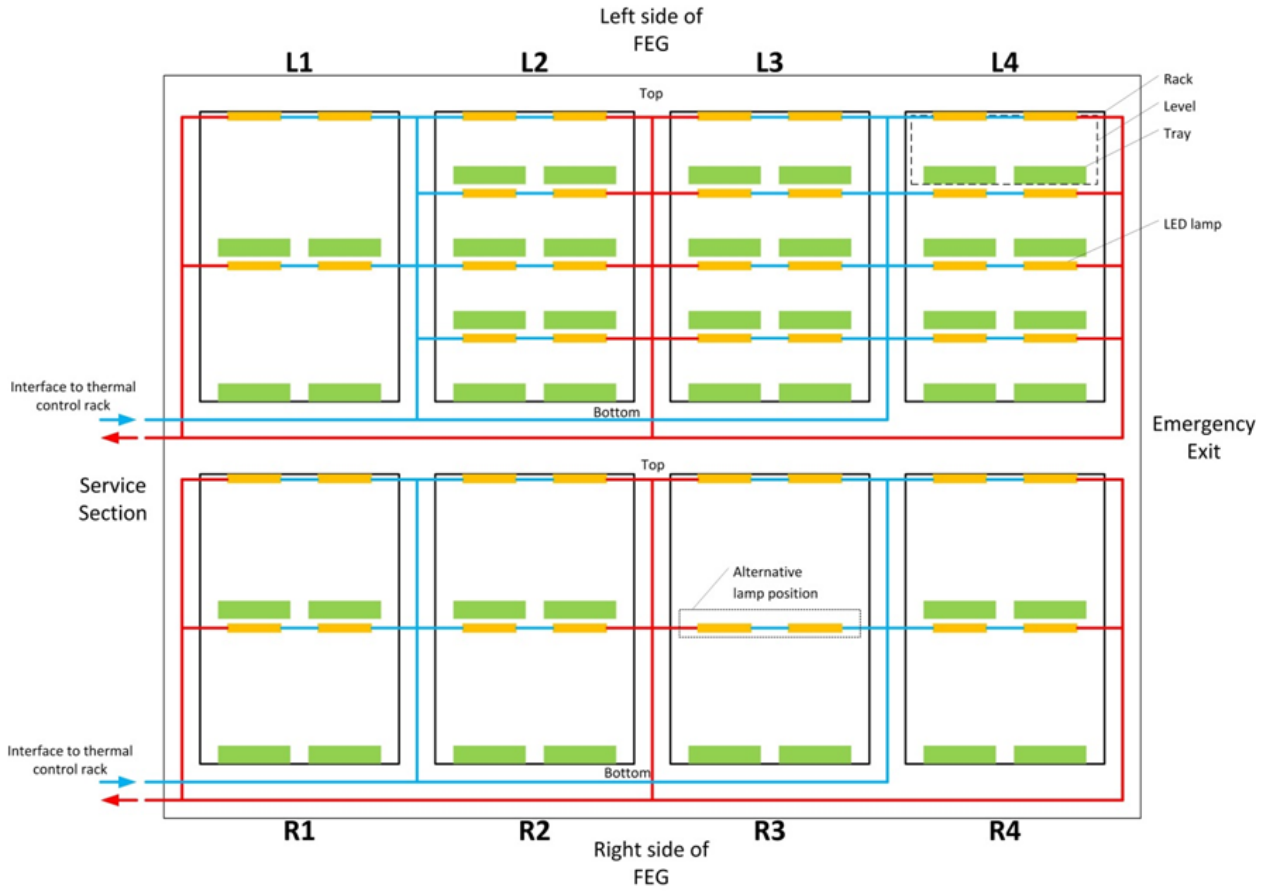



Figure 3.3: Structure of the ILS [18, p. 10]

The AMS is a closed system filtration stage that connects its air inlet and outlet to the FEG. It features a stacked setup of filters and sterilisation through ultraviolet lamps as well as heaters for temperature regulation.

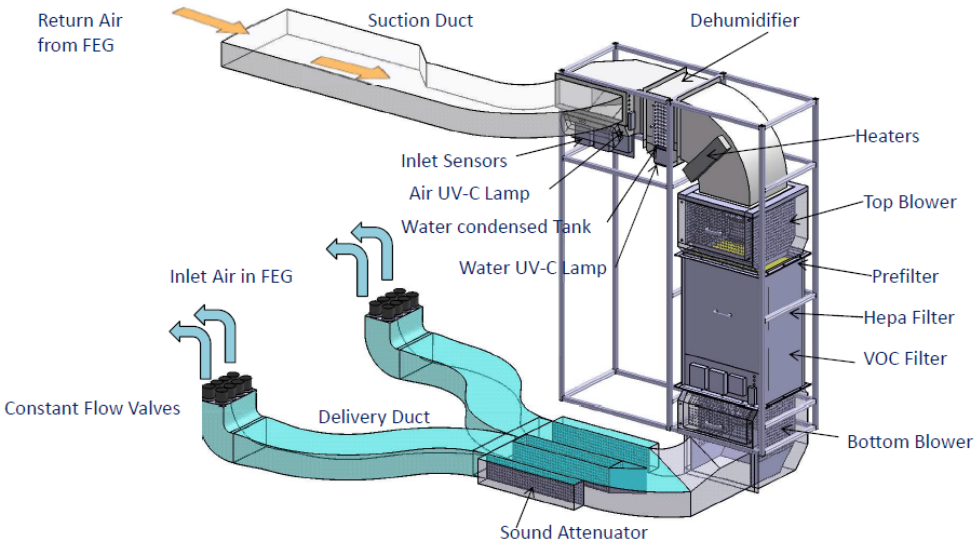



Figure 3.4: Structure of the AMS [17, p. 21]

The NDS operates in 2 main loops. Tank 1 and Tank 2 are enriched individually by the different dosing pumps that feature several types of nutritions as well as acid and base. From there, the solution is getting pumped into the trays where the plants are reached and the leftover solution returns to the bulk tank. Overflow is redirected into a waste water tank which is compromised by the fresh water tank. The AMS recovers water from condensation that feeds into the fresh water tank, which closes the water cycle under normal circumstances.

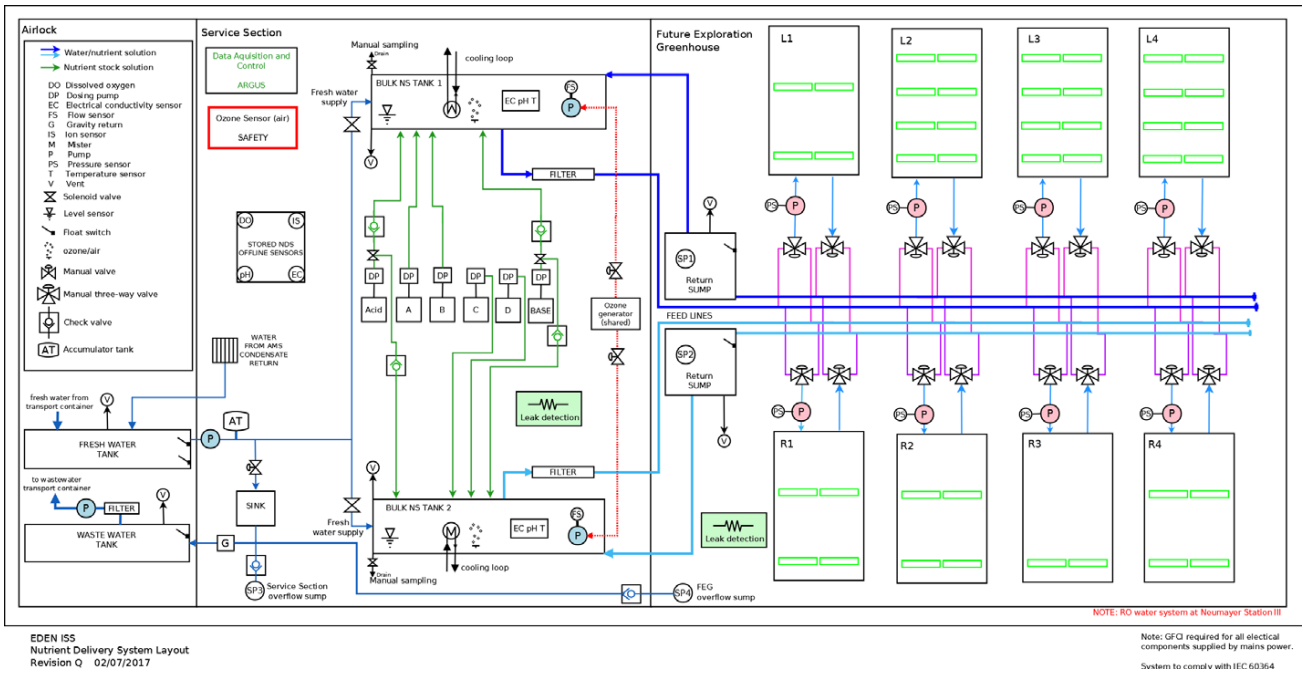



Figure 3.5: System components and block diagram of the NDS [19, p. 9]

# 4. Concept Modelling

## 4.1. Objectives

| id | content |
|---|---|
| O1 | reproduce the timeline so a similar behaviour is achieved → approximating a digital twin |
| O2 | represent core system behaviour in the connections of the simulation |
| O3 | integrate plants into the system of forecasting to connect them to the system variables |

Table 4.1: Model Objectives

## 4.2. Decision for Neural Network Models

Depending on this fundamental design decision the model is having different advantages and drawbacks over the other. The benefits of a manual network are, that the systems which are fully comprehensible can be very accurately described. This requires the complete set of influences to be known together with the function that is producing a certain value. For a case where this in not given, simplifications and estimations have to be made. If the set of influencing variables is not clear, manual selection holds the risk of misappropriating variables.

The NN method can theoretically minimize that risk by considering a bigger set of variables and therefore reach a lower error term in such cases. On the other side this behaviour is very vulnerable to overfitting which is tried to be counterweighted by generalisation filters. To let the NN identify system connections by itself can save a lot of time compared to manual modelling, especially for large systems. On the other hand, an NN can only be as good as the data it is supplied with. It also holds a great danger of generating spurious relationships since it is only minimizing the error term.

In short, it is a choice between a more deterministic and a stochastic explanation. In which machine learning seems to be the better heuristic approach. Even though the EDEN ISS MTF consists of many small fully understood components, their collective interaction in the FEG and the plants itself skyrockets the functions probably needed to describe them and the result is the case of incomplete information.

Because the quality of the DT is benchmarked by the error term here, our choice goes to using a NN.

## 4.3. Requirements

| id | headline | explanation | applied on example |
|---|---|---|---|
| R1 | a continuous data set | ongoing data points for every time step | no data holes |
| R2 | a complete data set | all information that wants to be identified, can be found in the data that is supplied | the recorded length and variables are sufficient |
| R3 | adequate neural structure | the inner neural network offers the right space and shape to emerge objective behaviour | enough neurons and connections |
| R4 | general predictability of the system | the system has to be deterministic in order to be predictable at all | theoretically and practically |

Table 4.2: Model Requirements

## 4.4. Use Cases

In a perfect implementation with excellent data, the following applications are be feasible. All system events can be predicted and therefore simulated. Normal behaviour as well as any system experiment can be carried out virtually. System robustness can be challenged by intentionally failing individual systems. The biomass can be predicted based on the current system status, which can be compared with the mission target and actions can be taken to adapt. System failures can be detected before they occur in the physical device, providing time for prevention.

## 4.5. The Model

In order to meet objective 1, the DT has to perform a forecasting action. In this paper this is achieved by time series forecasting where the DT is fed with an input and outputs the following future. This requires time to be discretised which is done by introducing time-steps. The smaller these time-steps are, the higher the resolution can be for our prediction. Consequently, the time-step size is set as low as possible to the resolution the data is available in. In case of the EDEN ISS MTF the current accessible resolution is 5 minutes and therefore becomes the time-step size of our model.

In terms of the input shape, a single time-step input for each variable only holds the information of absolute level and has no indication of the rate of change or more complex patterns a system could be driven into. A time-span of time-steps can carry such information and is therefore chosen here for both input and output. We are referring to those as the input- and output-window.

To discuss the size of the input and output window (the number of time-steps they contain), it comes advantageous when they share the same size, because the output can then be directly reused as a new input. Regarding the size of the output window, it comes down to the question what scale of predictions we are interested in. That the window size has an impact on our quality of prediction can be explained by the way learning is performed, specifically that the Mean Absolute Error (MAE) is minimised on the output window. Therefore, if we are interested in predicting only the very next time-steps precisely but nothing else, a short-termed prediction would suit a small window size. In our case, to represent the core system behaviours from objective 2, a good mid-term prediction like 24 hours is more meaningful about the system than to predict the next 10 minutes well.

Together with a reason that gets introduced in section 6.1, we are setting the size of the input and output window to 288, which equals 24 hours with a step size of 5 minutes. The working principle of these settings is exemplified for n features in Figure 4.1 where the true values to predict are called Labels.

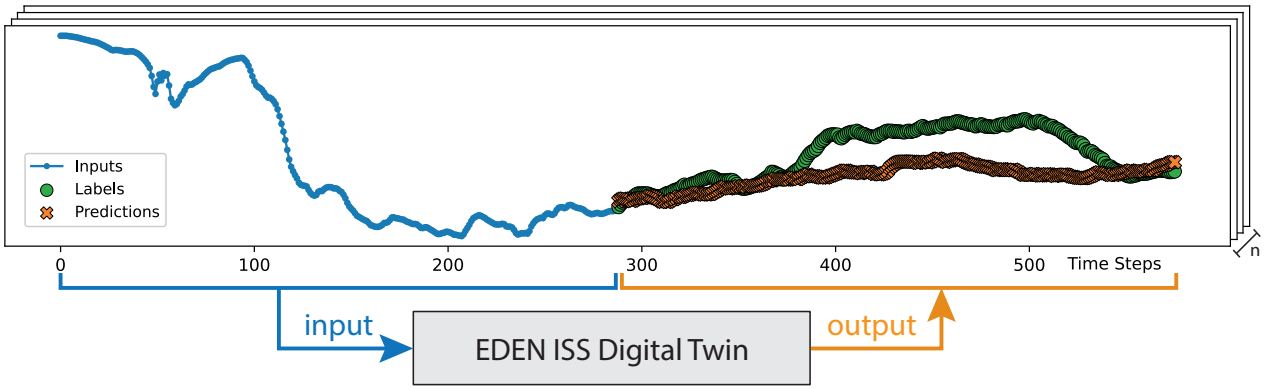



Figure 4.1: EDEN ISS digital twin working principle

Regarding the inner architecture of the EDEN ISS DT, a sub-model setup was chosen.

The reason for the first separation is, that the data from the MTF comes in two very different forms. One is the system variables being recorded automatically by the control unit with a high time resolution. Two is the crew variables being recorded manually with a very low resolution of only sowing and harvest information. Therefore, a different model structure is needed for the crew- and the system variables.

The second separation is made inside the system variables. As further discussed in chapter 6 we are dividing the system variables into time controlled and environment controlled features to assist their way of computation.

This results in the EDEN ISS DT model structure that is placed below in Figure 4.2.

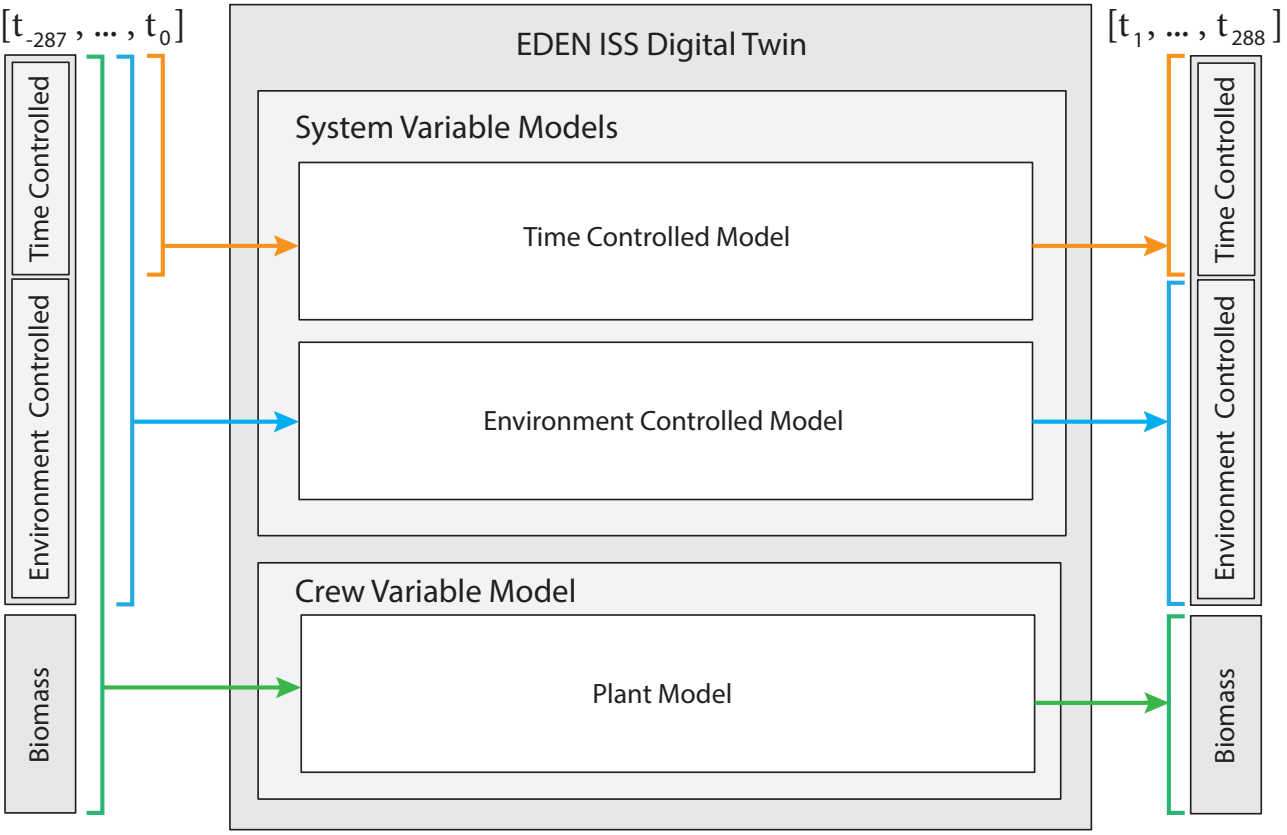



Figure 4.2: EDEN ISS digital twin model structure

As a whole, this model is starting with a window of the full vector of variables and is outputting the subsequent window of the same full vector of variables. Each part of the output vector is getting explained by a different sub model.

The sub-models development is carried out individually. The time controlled model in chapter 6.1, the environment controlled model in chapter 6.2 and the plant model in chapter 7.

In terms of data selection the first season in 2018 is chosen because all the system components are just recently installed at that point and nominal behaviour is expected due to their new condition.

# 5. Data Processing

## 5.1. Data Preparation

The starting point for the system variables that got recorded automatically are the subsystem data tables. Those are named GMTF (General MTF), AMS, ILS, NDS, and TCS. The records start on 07.02.18 and are cut off on 13.08.18 due to a variable change caused by a system update that took place on that date.

The resolution of data is shared among all tables with 5 minute steps but their exact timestamp varies due to the individual recording. In order to merge them into a single big data frame those timestamps have to match. To achieve this, all timestamps are getting rounded into flat 5 minutes. For example ”2018-02-07 0:05:47” → ”2018-02-07 00:05:00”. This induces a shift depending on the distance to the closest flat 5 minutes which can be 2 minutes and 30 seconds at maximum.

Under normal conditions a new data line was created every 5 minutes but due to system failures this has not always been the case. To be able to find those missing lines, each subsystem gets merged with a time-frame that is holding the complete set of 5 minute timestamps. The resulting table is containing the data and the missing entries as NaN (Not a Number). These NaN are counted to compute the percentage of missing data. For the most subsystems this missing data is around 2.5% with only AMS at around 6% and TCS at 99%. This makes us discard the TCS table at this point.

In order to fill the missing lines, which is necessary to fulfil Requirement 1, a linear interpolation is applied. This distorts the data and even brings in system behaviour that is not possible otherwise e.g. when a voltage can only be either high or low but gets interpolated in between.

After the remaining subsystem tables have been merged, the desired time span is cut out and constants and corruptive variables are removed. From this, 3 different data sets are saved, a full version that contains all variables, one containing only the time controlled variables and one containing only the environment controlled variables.

The full flow of the dataset creation is visualized in Figure 5.1. The corresponding implementation is described in page 35. The separation and identification of the time controlled variables is discussed in Chapter 6.1 and how the constants and corrupted variables are identified is stated at the next page.

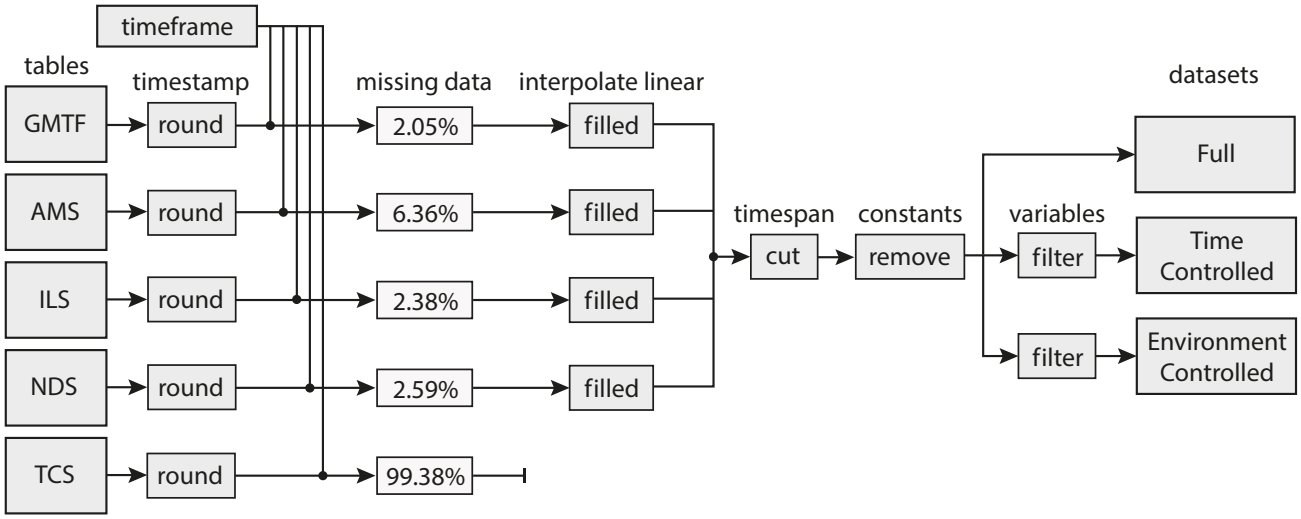



Figure 5.1: Flowchart of building the datasets

To build the dataset suiting for the usage of model training, we filter our dataset for constant or corruptive variables. Variables that are constant over the whole dataset can not contribute to any explanation in a neural network and are dropped because of that. We identify constant variables through calculating the mean and standard deviation of all of our variables as well as the MAE of a model that assumes constance, which we call Last. All the variables are normed beforehand so the MAE is comparable. The resulting table is sorted by MAE Last and the section of MAE Last $< 0.2$ is listed in Table A.1.

All variables that have a standard deviation of 0 are by definition constant and get marked as dropped. The variables whose standard deviation is unequal to zero but have a very high rate of explanation by the Last model are also discarded, the threshold is set at MAE Last $< 0.02$. The next 20 entries are evaluated manually by judging them according to their graph, which is plotted in Figure A.1. Here is distinguished between faulty read-ins like in feature ”SES TEMP AIR IN 2” and binary state controlling variables such as ”NDS-A DOSING PUMP” and their validity. The decision results are again listed in Table A.1.

The reason why an additional selection on the timespan was made is due to irregularities in the run-in phase in February 2018. As an example the feature ”TCS TEMP OUT EXIT” is plotted in Figure 5.2 where questionable values as 2000 degree Celsius appear. As a solution, this run-in section is cut off with a buffer so that our data set starts on 01.03.2018.

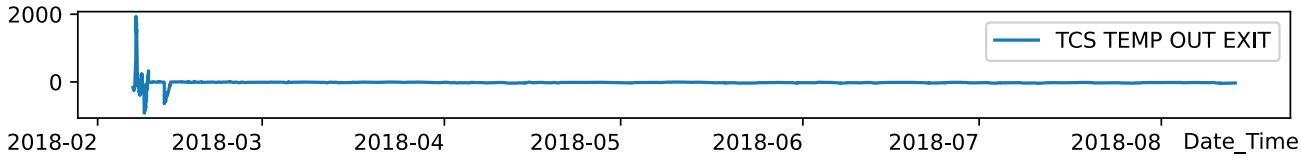



Figure 5.2: Example feature for initial irregularities

In addition to the corrupted and constant features, all ”[...]HP PUMP IRRIGATION SCHEDULE[...]” variables (from NDS) which had some kind of ”HH:MM:SS” format are dropped because their values meaning remains unclear.

Ambiguous feature names as ”EC Setpoint” and ”ph Setpoint” (from NDS) which appear twice in the original data table but have different values got renamed to ”EC Setpoint” and ”EC Setpoint 2” as well as ”ph Setpoint” and ”ph Setpoint 2”.

Another indication for double naming or wrong enumeration naming appeared for the ”L2-4R” and ”L2-4 R” feature (from ILS) which is usually the temperature of the light system in that tray. Since their values are not identical but only one temperature read-in per lamp is expected, they might need to get readdressed to suit the naming convention. However, they remained unchanged in our dataset but should be considered when lamp values are connected with individual trays.

Also mainly the ILS original data table is holding many corrupted lines, where a new line started in the middle of a previous one (Figure 5.3). This got identified while parsing the tables content into numbers. A few examples for the ILS table are sample number 26371, 49589, 51719 along with many more.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| [...] | 9000 | 6000 | 6000 | 6000 | 6000 | 9000 | 9000 | 9000 | 9000 | 9000 |
| | 9000 | 6000 | 6000 | 6000 | 6000 | 9000 | 9000 | 9000 | 9000 | 9000 |
| | 9000 | 6000 | 6000 | 600026371 | 25.04.2018 | 18:02:57 | 450 | 450 | 450 | 450 |
| | 9000 | 6000 | 6000 | 6000 | 6000 | 9000 | 9000 | 9000 | 9000 | 9000 |
| | 9000 | 6000 | 6000 | 6000 | 6000 | 9000 | 9000 | 9000 | 9000 | 9000 |

[...]

Figure 5.3: Example for corrupted lines in the original ILS data table

## 5.2. Data Standardization

In this chapter we want to propose techniques that are improving the documentation for transition to a data-frame that can be used for machine learning or other analysis.

The first issue can be found in the way data is currently accessed. The system generated data is saved on site in the Argus system of the MTF in Antarctica and is downloaded through a request that packs and creates the desired table. Due to the number of samples and the instability of the internet connection this method requires many hours and a great risk of abortion through connection loss. Why this can be a problem shows also in this work, where the data was only reasonably obtainable in a 5 minute resolution even though theoretically a 1 minute resolution is available on site in Antarctica that would be preferred. A faster and more reliable connection to the data base would increase accessibility and remove obstacles.

Many of the previously applied preparations such as removal of corrupted lines, double naming, data hole identification and value validation will be required to do for any work that is using these tables. An automated function that checks for these issues and either resolves them right away or flags them would contribute significantly to the quality of the documentation.

When hardware updates are conducted on the MTF the internal Argus system is assigning new identification numbers resulting in a dataset shift. To resolve the new identification numbers to their corresponding previous number would offer a longer and more consistent data base even if new features are added.

The solution proposal that solves all of the previous mentioned issues is, to parse the live data directly into a local database. An appropriate location to build this database is the DLR site at the institute in Bremen, where live data is already sent to for mission control. An implementation in SQL for example can handle the automated checks as well as the desired parsing and identification resolving.

The reason why data holes are created in general is not entirely clear. If one reason is an internal saving error of the Argus system but not on live data that got send, then such a solution would even prevent some data holes that would have been created otherwise.

The second issue is the crew documentation of plant data which is currently conducted in a way that suits manual data processing by utilizing multiple Excel sheets. For example are all lettuce statistics clustered into their genetic breed and listed next to each other but have individual tables for either batch- or spread-harvest. Any rearrangement such as sorting by tray or time across the whole dataset means a lot of manual work.

A solution where any kind of sorting can be performed and where even batch harvest and spread harvest share the same documentation structure can be seen in Figure 5.4 below. The core idea is to create a register where every plant is defined and a second table for the harvest timeline which both are connected through the plants unique identifier. Therefore can spread harvest easily be covered by appearing multiple times in the harvest timeline. An implementation as a SQL database is again highly advantageous here.

**Plant Register**

| unique identifier | | | | Transfer |
|---|---|---|---|---|
| Type ID | Plant ID | Tray | Sowing date | date |
| Batavia | 1 | L2-4L | 07.02.2018 | 27.02.2018 |

**Harvest Timeline**

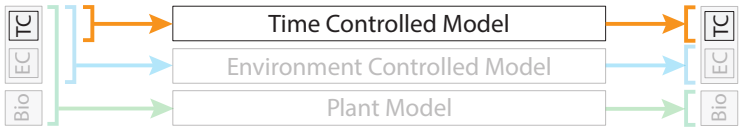
| unique identifier | | | Edible FW | Inedible FW | Number of | Cycle length | normalized |
|---|---|---|---|---|---|---|---|
| Type ID | Plant ID | timestamp | [g] | [g] | features | [d] | FW [g/d] |
| Batavia | 1 | 20.03.2018 | 539.5 | 204.3 | 6 | 41 | 13.16 |

Figure 5.4: Proposing structure to improve plant harvest documentation

# 6. System Variables Model

Due to the function of a greenhouse some values are scheduled by time, such as the lights. In contrast to that some variables are influenced by interrelationships when systems interact with each other. When this mixture of time- and environment controlled data is fed together in a training set for a neural network, it tries to explain the time controlled variables also with all the other values it has available. To prevent this unwanted behaviour the system variables model is split into the following 2 categories.

## 6.1. Time Controlled Model




To identify the set of variables that are time controlled we start out with the light voltage values from the ILS subsystem from ”L1-2L BLUE” to ”L1-4L FAR RED” and isolate them into a new dataset. Since this set shares a high similarity along all features one sample is picked for analysis and transferability is assumed for now because of high visual similarity. The representative feature is plotted in Figure 6.1 with its clearly repeating patterns. To forecast those patterns efficiently several types of models are tried out on it.

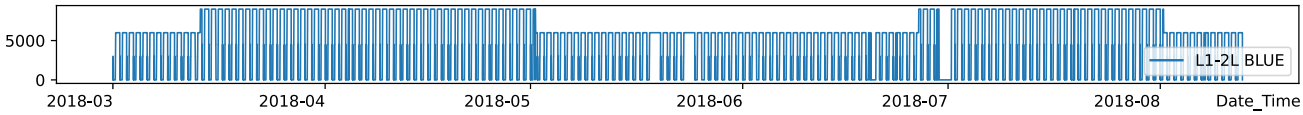



Figure 6.1: Representative feature ”L1-2L BLUE” for time controlled system variables

The first approach is to just output the input window. Even though this method seems inconclusive at first, it evolves to be very promising when the window size is set exactly equal to the period length. This way its natural cycle is matched and to just return the input turns into a pretty good estimator. We are going to call this model ”Repeat”.

A different approach is to let a model learn on this pattern with the hope to catch the repetition and even average out some small variations between them. For such a model however, every time controlled feature has to be individualized since they try to explain each other again otherwise. The different learning models that are tried out here are simple Linear, Dense, Convolutional, LSTM and Autoregressive LSTM models. A full structural overview of all model types is visualized in A.2 and the code on page 36.

The learning model is having a structural advantage over the Repeat model if some of the time controlled variables differ in periodic length. Because a suiting learning model can still predict those well as long the periodic length is smaller than the input window size, while the Repeat model has an offset when the periodic length does not exactly match the input window size. In our case all time controlled variables currently share the 24h periodic cycle and therefore eliminate this advantage for now.

To provide a decision, the representative feature is applied to the Repeat model and multiple learning models all with a 24h (288 steps each 5min) input window. The learning models are executed with ”early stopping” and the standard structure [15] to get a first impression how well they perform on this task. The comparison of their result performance is visualized in the following benchmark histogram in Figure 6.2 together with a prediction sample of the best 2 models.

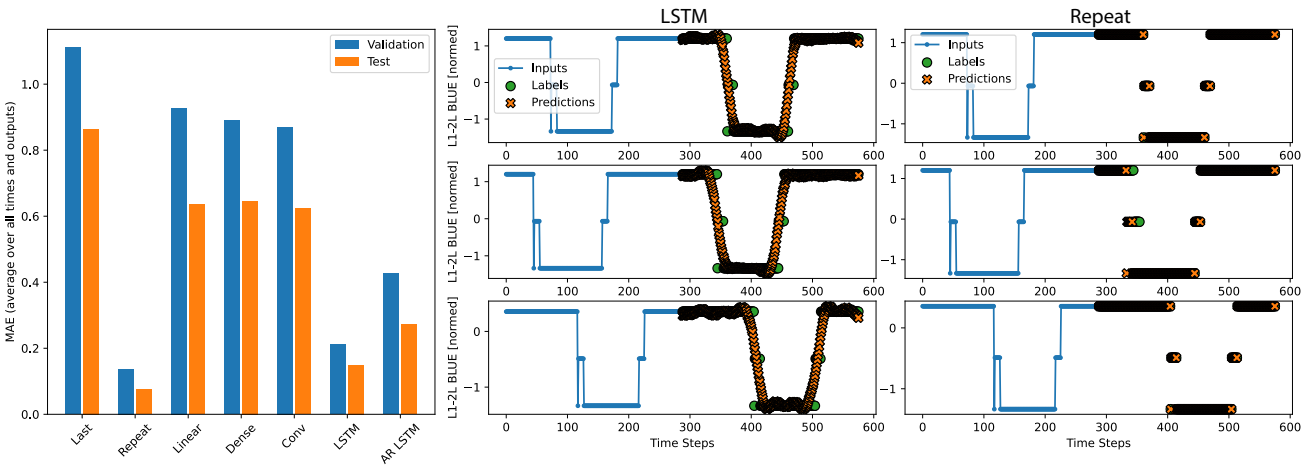



Figure 6.2: Performance summary of different models to explain time controlled system variables

Usually a Test MAE is expected to be bigger than the Validation MAE since the network was optimized upon the validation dataset, but in our case the test data shares more similarity with the train data than the validation data. Interpolated data that appears in the timespan of the validation data influences that.

In relative means a 0.0755 Test MAE for the Repeat model is an outstanding low error in prediction and therefore a very reliable forecast for our time controlled variables.

To discuss the leftover error, the interpolated data holes (for ILS around 2% of the total data), which are also visible in Figure 6.1 induce mostly a training and validating error. The biggest source of error of the test set however can be found in the change of the lighting patterns maximum altitude, here because a different plant has been planted in this tray.

Those changing patterns over time are represented surprisingly well by the Repeat model since the most recent pattern is very likely to be the next pattern too. But at the transition between those, quite an error is made. Comparing the training time, listed in Figure A.4 the Repeat model is simply a re-routing action and requires therefore almost no compute power at all compared to the LSTM model with 114 seconds per feature.

This brings us to the conclusion to use a Repeat model to forecast the time controlled system variables.

To test our dataset for more 24h periodic time controlled variables, this Repeat model is applied together with a baseline model Last on all of our features over all time. Now sorting by the smallest Repeat MAE a selection worth inspecting is created, which is Table A.2.

Unsurprisingly our light voltage variables are listed here, but also some new variables that aren't obviously time controlled like the "FEG CO2 TARGET". When a high MAE in the baseline model Last comes together with a very low MAE in the Repeat model, there is a strong indication for a time controlled variable. Our deciding rule will be MAE Repeat $< 0.15$ and MAE Last $>>$ MAE Repeat.

Resulting in the decision weather taking this variable in the list of time controlled variables or not (column tc). Therefore, the complete list of time controlled variables are also the ones that got identified in Table A.2.

To give a final value on performance of this model on all the time controlled variables the MAE over all of them is computed and since the Repeat model is no learning model, it can be tested on the full dataset range again. This performance value is: Time Controlled total MAE $= 0.065$.

## 6.2. Environment Controlled Model


As environment controlled variables we define the remaining features that have neither been filtered by constance and corruption nor identified as time controlled in chapter 6.1. Examples for such are ”FEG TEMPERATURE 1”, ”FEG HUMIDITY 1” and ”FEG CO2 1” which are sensor readings from the FEG and describe air as a central exchange medium.

To represent these more complex interconnections a learning model is used. The full set of models that was used before is now shortened. The Repeat model is discarded since it is not a learning model and representing a time controlled behaviour. For a less mandatory reason the Autoregressive LSTM model is also dropped due to its very high demand of computation power and therefore training time. An indication of the amount of time that is required with given hardware can be found in the computation statistics of the time controlled model in Table A.4 which only factored in one single feature, which is significantly less than the number of environment controlled features alone which count up to 96 (A.3).

Having multiple variables as an input and output in combination with our given interconnections, all features have the potential to influence any other features and themselves.

Each models detailed layers composition is printed in its summary at Figure A.3 where the lambda layer represents the input layer and the last 2 are the output layer. The corresponding implementation can be found on page -ref-. These settings are based on the multi-step time series forecasting template supplied in the Tensorflow documentation [15] and have been slightly adjusted for this study case.

In order to balance underfitting and overfitting the size of the hidden neurons is essential. To estimate a suitable neuron size in the hidden layers can be done by rough guidelines as a starting point or statistically developed methods which only apply for specific use cases.

Multiple of those statistical estimation methods are presented and compared in [20, p. 9] where the number of hidden neurons $N_h$ are estimated by the number of input parameters $n_{in}$ resulted in the papers proposed equation $N_h = (4n_{in}^2 + 3)/(n_{in}^2 - 8)$. In our case however we are dealing with a huge number of inputs as high as $n_{in} = 288 * 96 = 27648$ for the EC only model and $n_{in} = 288 * 183 = 52704$ for the EC&TC model which either converges the function like above on a low level or grows towards infinite with other estimators as $N_h = 2^{n_{in}} - 1$ [20, p. 9]. Therefore a more use case suited estimator is needed or no statement can be made through this approach.

Considering rough unproven guidelines as $N_h$ should be between $n_{in}$ and $n_{out}$, $N_h = \frac{2}{3}n_{in} + n_{out}$ and $N_h < 2n_{in}$ we end up with a smallest upper bound of $N_h \leq n_{in} = 52704$ in our case with $n_{out} = 27648$. For comparison the values the template started out for the hidden layers are $512$ (Dense model), $256$ (Convolutional model) and $32$ (LSTM model) where the convolutional and LSTM numbers contains more parameters than the dense number which makes it not entirely comparable with its hidden neural count. So the aim is roughly an increase from the starting point.

By slowly raising these values we are getting limited by the graphics card memory size (which is 4 Gigabyte in our case) that is needed to hold this model efficiently during training before we can get reasonably close to the previous upper bound. This lets us settle down on $1024$ (Dense model), $512$ (Convolutional model) and $128$ (LSTM model) which is between a 2x and 4x increase of the hidden layer that came along with a significant performance increase.

In conclusion, the system might still be in an underfitted stage.

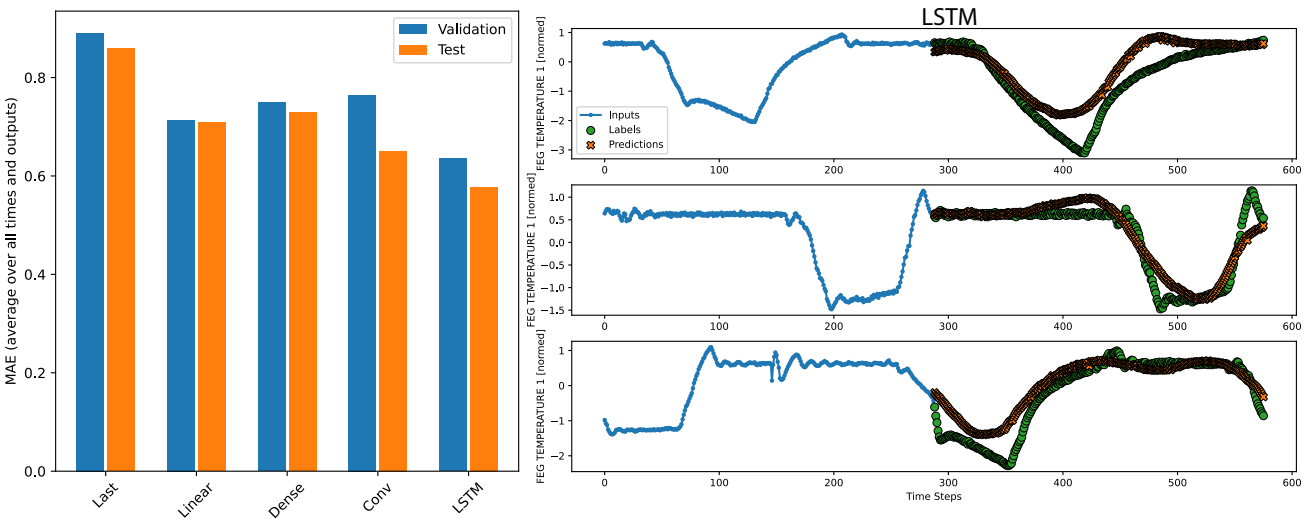



Figure 6.3: Performance summary of different models to explain environment controlled variables

In Figure 6.3 is shown the result of the selected set of models under an early stopping method. The more complex models as Convolutional and LSTM are outperforming the more simple models as Linear and Dense. This can be explained by the fact that the Linear and the Dense model only use the last time step for prediction. This makes us select the Convolutional and LSTM model for further testing.

The best performing model under these settings, the Convolutional model, has a Test MAE of 0.5347 (Table A.5) which is still a high level of inaccuracy. However, these settings caused the validation MAE to rise while the training MAE declined, resulting in the first Epochs being the best while further training even reduced the performance on the validation data. This behaviour can be counterweighted by what is called regularization of neural networks [21] where we are using a combination of dropouts and L2 weight decay to achieve a combined improvement on both validation and training data.

With the regularization and further structure expansion applied (A.4) on a stronger graphics card the LSTM model is trained again for 15 Epochs which training history is presented in Figure 6.4 below.

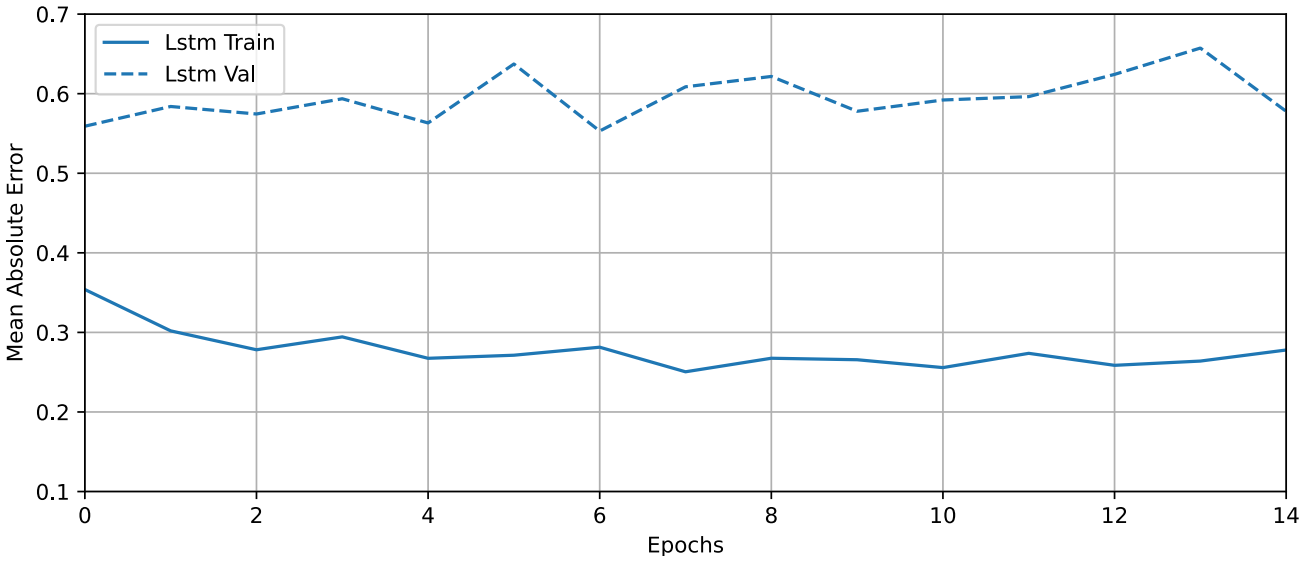



Figure 6.4: Training history of the final environment controlled models

In order to get a visual impression on the prediction the LSTM model makes, key features about the FEG and the MTF are plotted in Figure 6.5 below. The selected features are in particular ”FEG TEMPERATURE 1”, ”FEG HUMIDITY 1”, ”FEG CO2 1”, ”FEG OXYGEN”, ”pH 1 TANK 1”, ”EC 1 TANK 1” and ”VOLUME TANK 1”. The individual performance on every feature is listed in Table A.7 and result statistics in A.6.

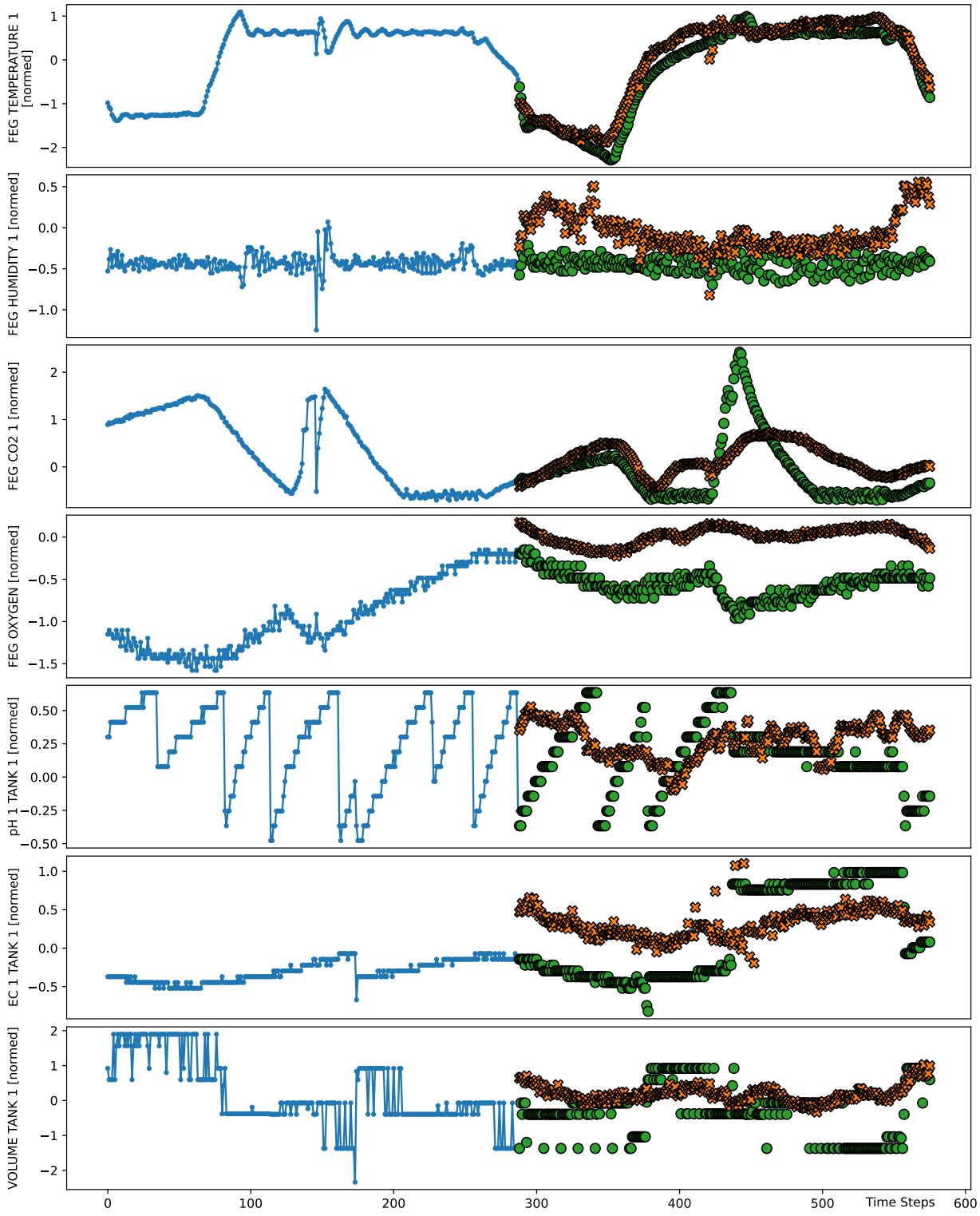
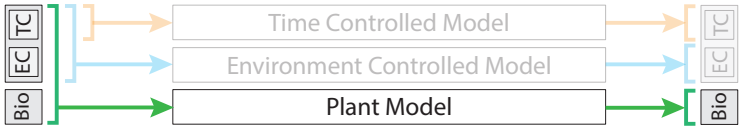



Figure 6.5: Prediction of key features for environment controlled system variables on test data

# 7. Plant Model




In this section the connection from the system to the biomass is attempted by explaining the plants biomass through the time controlled and environment controlled variables.

The first difficulty appears for creating a plant vector that addresses the biomass information that we want to predict. A first option is to map this vector to the plant trays, since one tray is the smallest measured unit to hold a plant entity. Where the vector, containing all 42 trays, can be connected to the full set of variables from the system into a single big model. A second option is to individualize every tray into its own model instance. One instance would only contain corresponding features as the specific lamp that enables this tray and the system variables as the CO2 level that generally apply to all. This smaller model can be designed in a universal way to fit every individual tray, so the same structure is shared among them. This way every smaller model connects to a subset of the system variables.

The second point is to determine the window size of the vector and therefore the time span of prediction.

The data that is available from the crew documentation is giving the absolute weight for edible and inedible biomass when harvest is performed. A growth period from sowing to harvest takes around 37 days for Batavia lettuce for example but varies every time. To set the time span equal to one growth period as the predictions window size, we are trying to develop a model that is forecasting the biomass for each step between sowing and harvesting.

In order to do that however, the plant data must be available for every time step which we are planning to compute so that a model can learn the influences from the system variables on the biomass. Requirement 2 is where this problem is addressed, that the information that wants to get identified has to be contained in some way in the data that is supplied. In our case the data that is supplied, is the system variables in a resolution of 5 minutes and the crew variables in a resolution of 37 days. In order to merge both into a continuous data set as mentioned in requirement 1, the minimum resolution (highest time step) has to be chosen. Therefore, it is not possible to make predictions in a higher resolution than the 37 days which is supplied. Even a prediction with a step size of 37 days reduces our sample number to such a low level that a NN might not be able anymore to be reasonably trained by it.

An attempt to fix this could be, to interpolate the missing data through a growth function of the specific plant. However, this does not help our goal, since a NN would not learn the influences from the system variables on the biomass but would rather learn that the system variables have no influence at all, since the interpolated data is created by a closed function.

A second attempt to fix this could be, to recover the missing data by the photos that have been taken automatically from each tray which are available in a higher time resolution. A model that takes the pictures as an input and outputs the current biomass could indeed represent the actual data and influences could transfer. However this requires the model to have a higher precision than the influences are from the system on the plant, in order for the influences to be preserved. Even an optimistic guess of influences from the system by a few grams per time step would require an insane level of precision for the photo model. Even with a perfect 3D model of the plant an inaccuracy in the range of grams is likely, but in the case of the MTF we are relying on a single perspective photo series.

Resulting in the conclusion that a suiting plant model cannot be developed with the available data.

# 8. Model Validation

## 8.1. Conclusions

The developed DT has its limitations and accomplishments that we want to address in this section.

Because of multiple system hardware updates that took place between the first season and today, our developed shape of variables is not identically shared by the currently running system. This affects the integration of live data where the live feed of the MTF was planned to use as an input where the DTs output would have been displayed in mission control.

Objective 3, to integrate plants into the forecasting system and therefore add an explanation about the biomass was not achieved with the available data, which did not meet the requirements in terms of length and information containment.

In retrospective, the long term use cases as they got stated at chapter 4.4 are requiring a mature stage of a DT where this work is providing rather an early prototype as a proof of concept. Therefore are uses cases as to carry out entire experiments virtually, certainly not possible in this stage of development.

By going over the feature names in Table A.6 we observe that ”FEG TEMPERATURE TARGET” and ”SES TEMPERATURE TARGET” are selected as environment controlled features but they are actually human set targets that follow a time schedule. Therefore it has to be considered that the filter methods we applied did not sort entirely correct. In fact also the term of ”time controlled” variables needs to be reconsidered since the lamp values are mostly time regulated but the length of the on period and its amplitude was switched by the crew according to the plant in that tray and therefore the plant schedule. Additionally are the ”environment controlled” variables not only dependent on all the system variables but also the plants themselves. The system has more bidirectional and non separateable influences than assumed and the simplifications in our model hold the cost of inaccuracy.

On the other side, the DT holds a significant improvement in explanation for the environment controlled variables over the Last baseline model that assumes constance where the DT reached a Test MAE of 0.5582 compared to the baseline Test MAE of 0.859 (Table A.6). Where for the time controlled system variables the predictions are highly accurate with a Test MAE of 0.065 that fulfils the order of magnitude that was striven for.

In total, objective 1 (to achieve a similar behaviour) and objective 2 (to represent core system behaviours inside the NN connections) are partly achieved. The individual MAE from Table A.7 show, that the DT performance varies greatly over the different variables, where for example various types of temperatures are explained better but EC and pH levels are explained worse than the baseline. By taking a look at the plotted ”pH TANK 1” feature in Figure 6.5 a saw wave form is recognizable which is apparently much harder for the NN to represent. In Figure 6.5 can also be seen that the NN did not always pick up that the last value of the input window is a good estimation for the starting point of its own prediction, which results in an offset as for feature ”FEG OXYGEN”.

After all, the proof of concept and the matching general trend that the model predicts for most of the features, is the core accomplishment.

## 8.2. Discussion

For a follow-up a lot of parameters are still open for testing and optimisation since the surface was only scratched of those methods. Specifically the model type, the layer structure, the neuron size, the optimiser, the batch size, the window size and the regularisation are all parameters that hold much potential for improvement over the results that have been produced in this work.

For example could a sparse regression model as CINDy (Conditional gradient-based Identification of Non-linear Dynamics) fit the desired application of a digital twin better than the time series forecasting approach we took.

Limitations that we encountered here can be removed by future work as well. For example can the live data integration be performed by training a DT with the new shape of the current data. And in order to connect the plants into the system, installing tray weight sensors could give insight to a biomass connection though precise high frequent biomass measurements.

Regarding the use case of product development, to be able to run the EDEN ISS system in a specific danger scenario and troubleshoot virtually, will need a full concept design. Identifying all required informations in order to pick the right sensors to produce a dataset that makes it possible for a NN to learn all of those behaviours. A system in development stage is more suitable for such an attempt than the EDEN ISS MTF which is in a mature phase of its life cycle already.

# A. Resources Appendix

| Variable name | mean | sd | MAE Last | dropped |
|---|---|---|---|---|
| CPO HUMIDITY | 0 | 0 | nan | yes |
| SES TEMPERATURE 2 | 0 | 0 | nan | yes |
| SES HUMIDITY 2 | 0 | 0 | nan | yes |
| AMS-SES-FAN AIR IN | 0 | 0 | nan | yes |
| AMS-SES-VALVE AIR IN | 0 | 0 | nan | yes |
| AMS-SES-HUMIDIFIER AIR IN | 0 | 0 | nan | yes |
| AMS-SES-HEATER AIR IN | 0 | 0 | nan | yes |
| FEG LVL SWITCH COND MAX | 0 | 0 | nan | yes |
| FEG TEMPERATURE PHM 2 | 0 | 0 | nan | yes |
| FEG HUMIDITY PHM 2 | 0 | 0 | nan | yes |
| FEG TEMPERATURE PHM 3 | 0 | 0 | nan | yes |
| FEG HUMIDITY PHM 3 | 0 | 0 | nan | yes |
| FEG TEMPERATURE PHM 4 | 0 | 0 | nan | yes |
| FEG HUMIDITY PHM 4 | 0 | 0 | nan | yes |
| FLOW METER TANK 1 | 100 | 0 | nan | yes |
| EC Setpoint | 220 | 0 | nan | yes |
| pH Setpoint | 590 | 0 | nan | yes |
| EC Setpoint 2 | 350 | 0 | nan | yes |
| pH Setpoint 2 | 590 | 0 | nan | yes |
| SES TEMPERATURE | -40 | 0.18 | 0.0093 | yes |
| SES HUMIDITY TARGET | 30 | 0.25 | 0.018 | yes |
| AMS-FEG-FANS CIRC L1.L2 | 99.99 | 0.83 | 0.018 | yes |
| AMS-FEG-FANS CIRC L3.L4 | 99.99 | 0.83 | 0.018 | yes |
| AMS-FEG-FANS CIRC R1.R2 | 99.99 | 0.83 | 0.018 | yes |
| AMS-FEG-FANS CIRC R3.R4 | 99.99 | 0.83 | 0.018 | yes |
| AMS-FEG-UV LAMP AIR | 99.99 | 0.83 | 0.018 | yes |
| NDS-BASE SOLENOID | 0.01 | 0.92 | 0.0186 | yes |
| FEG AIR FLOW | 0.76 | 6.78 | 0.0205 | yes |
| NDS-BASE DOSING PUMP | 0.01 | 1.03 | 0.0208 | yes |
| NDS-ACID SOLENOID | 0.01 | 1.21 | 0.0246 | yes |
| PDS TEMP CONTROL BOX | 169.58 | 503.41 | 0.0452 | yes |
| SUBFLOOR CPO 2 | 97.62 | 285.77 | 0.0484 | yes |
| AMS-FEG-FAN AIR LOOP 1 | 70.53 | 2.51 | 0.0551 | yes |
| AMS-FEG-FAN AIR LOOP 2 | 70.53 | 2.51 | 0.0551 | yes |
| SES TEMP AIR IN 1 | 181.01 | 567.44 | 0.0585 | yes |
| FEG HUMIDITY TARGET | 64.5 | 1.61 | 0.0612 | yes |
| NDS-ACID DOSING PUMP | 0.27 | 5.13 | 0.1029 | **no** |
| NDS-PUMP FW | 99.65 | 5.9 | 0.1034 | yes |
| FLOW METER TANK 2 | 99.28 | 8.48 | 0.1074 | yes |
| SES TEMP AIR IN 2 | 846.67 | 1114.15 | 0.1104 | yes |
| NDS-SOLENOID FW TANK 1 | 0.3 | 5.47 | 0.1106 | **no** |
| NDS-REC PUMP TANK 1 | 99.59 | 6.38 | 0.1133 | yes |
| NDS-REC PUMP TANK 2 | 99.59 | 6.36 | 0.1134 | yes |
| NDS-SOLENOID FW TANK 2 | 0.49 | 6.94 | 0.1414 | **no** |
| NDS-A DOSING PUMP | 0.51 | 7.13 | 0.1426 | **no** |
| NDS-B DOSING PUMP | 0.51 | 7.13 | 0.1426 | **no** |
| SES DOOR STATUS | 82.75 | 900.23 | 0.18 | **no** |

Table A.1: Identify constants and corrupted data: variables sorted ascending on MAE Last with <0.2

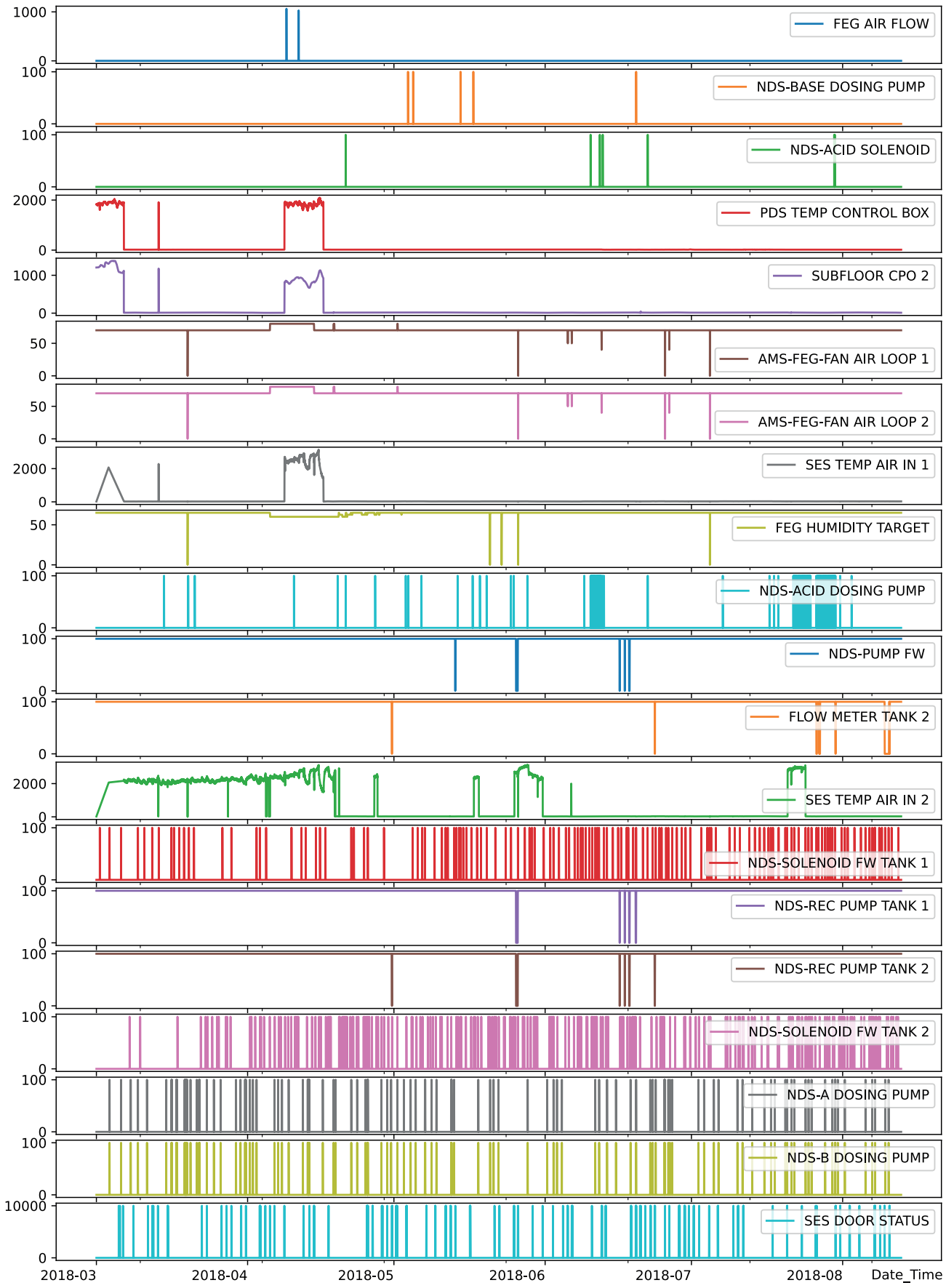



Figure A.1: Identify constants and corrupted data: data history of the last 20 entries of A.1

| Name | MAE Last | MAE Repeat | tc |
|---|---|---|---|
| SES TEMPERATURE TARGET | 0.7909 | 0.0309 | **y** |
| FEG CO2 TARGET | 0.8565 | 0.0333 | **y** |
| FEG TEMPERATURE TARGET | 0.8259 | 0.0383 | **y** |
| L4-3R RED | 0.9375 | 0.0557 | y |
| L4-3R BLUE | 0.939 | 0.0558 | y |
| L3-3L RED | 0.8455 | 0.0573 | y |
| L3-3R RED | 0.8455 | 0.0573 | y |
| NDS-OZONE GENERATOR | 0.7274 | 0.06 | **y** |
| L4-4R RED | 0.8512 | 0.0606 | y |
| L3-2R BLUE | 0.9649 | 0.0623 | y |
| L4-3L BLUE | 0.9649 | 0.0623 | y |
| R1-2R BLUE | 0.9649 | 0.0623 | y |
| R1-2L BLUE | 0.9649 | 0.0623 | y |
| R1-4R BLUE | 0.9649 | 0.0623 | y |
| R1-4L BLUE | 0.9649 | 0.0623 | y |
| R2-2R BLUE | 0.9649 | 0.0623 | y |
| R2-2L BLUE | 0.9649 | 0.0623 | y |
| R2-4R BLUE | 0.9649 | 0.0623 | y |
| R2-4L BLUE | 0.9649 | 0.0623 | y |
| R3-2/4R BLUE | 0.9649 | 0.0623 | y |
| R3-2/4L BLUE | 0.9649 | 0.0623 | y |
| L3-2R RED | 0.9649 | 0.0623 | y |
| L4-3L RED | 0.9649 | 0.0623 | y |
| R3-2/4R RED | 0.9649 | 0.0623 | y |
| R3-2/4L RED | 0.9649 | 0.0623 | y |
| R1-2R RED | 0.9637 | 0.0624 | y |
| R1-2L RED | 0.9637 | 0.0624 | y |
| R1-4R RED | 0.9637 | 0.0624 | y |
| R1-4L RED | 0.9637 | 0.0624 | y |
| R2-2R RED | 0.9637 | 0.0624 | y |
| R2-2L RED | 0.9637 | 0.0624 | y |
| R2-4R RED | 0.9637 | 0.0624 | y |
| R2-4L RED | 0.9637 | 0.0624 | y |
| L2-1L BLUE | 0.9649 | 0.0624 | y |
| L2-1R BLUE | 0.9649 | 0.0624 | y |
| L2-2L BLUE | 0.9649 | 0.0624 | y |
| L2-2R BLUE | 0.9649 | 0.0624 | y |
| L3-4L BLUE | 0.9649 | 0.0624 | y |
| L4-1L BLUE | 0.9649 | 0.0624 | y |
| L4-1R BLUE | 0.9649 | 0.0624 | y |
| L4-2L BLUE | 0.9649 | 0.0624 | y |
| L4-2R BLUE | 0.9649 | 0.0624 | y |
| L2-1L RED | 0.9649 | 0.0624 | y |
| L2-1R RED | 0.9649 | 0.0624 | y |
| L2-2L RED | 0.9649 | 0.0624 | y |
| L2-2R RED | 0.9649 | 0.0624 | y |
| L3-4L RED | 0.9649 | 0.0624 | y |
| L4-1L RED | 0.9649 | 0.0624 | y |
| L4-1R RED | 0.9649 | 0.0624 | y |

| Name | MAE Last | MAE Repeat | tc |
|---|---|---|---|
| L4-2L RED | 0.9649 | 0.0624 | y |
| L4-2R RED | 0.9649 | 0.0624 | y |
| L3-3L BLUE | 0.9626 | 0.0626 | y |
| L3-3R BLUE | 0.9626 | 0.0626 | y |
| L4-4R BLUE | 0.9633 | 0.063 | y |
| L4-4L BLUE | 0.9642 | 0.0636 | y |
| L3-1L BLUE | 0.9646 | 0.0636 | y |
| L3-1R BLUE | 0.9646 | 0.0636 | y |
| L1-2L FAR RED | 0.8941 | 0.0641 | y |
| L1-2R FAR RED | 0.8941 | 0.0641 | y |
| L1-4L FAR RED | 0.8941 | 0.0641 | y |
| L3-2L RED | 0.9405 | 0.0654 | y |
| L3-2L BLUE | 0.942 | 0.0654 | y |
| L3-4R BLUE | 0.9657 | 0.0663 | y |
| L3-4R RED | 0.9657 | 0.0664 | y |
| L2-3L RED | 0.9616 | 0.0671 | y |
| L2-3R RED | 0.9616 | 0.0671 | y |
| L2-3L BLUE | 0.9617 | 0.0671 | y |
| L2-3R BLUE | 0.9617 | 0.0671 | y |
| L2-4L BLUE | 0.9435 | 0.0673 | y |
| L2-4R BLUE | 0.9435 | 0.0673 | y |
| L4-4L RED | 0.8823 | 0.0674 | y |
| L2-4L RED | 0.9424 | 0.0674 | y |
| L2-4R RED | 0.9424 | 0.0674 | y |
| L3-1L RED | 0.9424 | 0.0674 | y |
| L3-1R RED | 0.9424 | 0.0674 | y |
| L1-2R BLUE | 0.922 | 0.0694 | y |
| **L1-2L BLUE** | 0.9221 | 0.0694 | y |
| L1-2L RED | 0.8376 | 0.07 | y |
| L1-2R RED | 0.8376 | 0.07 | y |
| R4-4R BLUE | 0.8494 | 0.0717 | y |
| R4-4L BLUE | 0.8495 | 0.0717 | y |
| R4-2L BLUE | 0.9287 | 0.0727 | y |
| L1-4L BLUE | 0.9202 | 0.0731 | y |
| L1-4R BLUE | 0.9202 | 0.0731 | y |
| R4-4L RED | 0.8591 | 0.0744 | y |
| R4-4R RED | 0.8592 | 0.0744 | y |
| R4-2R BLUE | 0.9199 | 0.075 | y |
| L1-4L RED | 0.8219 | 0.0757 | y |
| L1-4R RED | 0.8219 | 0.0757 | y |
| R4-2L RED | 0.864 | 0.076 | y |
| R4-2R RED | 0.719 | 0.08 | y |
| FEG PAR LIGHT 2 | 0.9341 | 0.0982 | **y** |
| NDS-SOLENOID FW TANK 1 | 0.1106 | 0.1025 | **n** |
| NDS-ACID DOSING PUMP | 0.1029 | 0.1051 | **n** |
| FEG PAR LIGHT 1 | 0.9409 | 0.1377 | **y** |
| NDS-SOLENOID FW TANK 2 | 0.1414 | 0.1401 | **n** |
| NDS-A DOSING PUMP | 0.1426 | 0.1432 | **n** |
| NDS-B DOSING PUMP | 0.1426 | 0.1432 | **n** |

Table A.2: Identify time controlled variables: sorted ascending on MAE Repeat with <0.15

Note: If not further stated, all training has been carried out on an NVIDIA GTX 970 with CUDA v11.0

| Class | Number of features |
|---|---|
| Constant or Corrupted | 41 |
| Time Controlled | 87 |
| Environment Controlled | 96 |
| Total | 224 |

Table A.3: Number of features by dataset class

| Model | Test MAE | Train time (in s) |
|---|---|---|
| Last | 0.8642 | 1 |
| Repeat | 0.0755 | 1 |
| Linear | 0.6356 | 17 |
| Dense | 0.6454 | 46 |
| Conv | 0.6239 | 31 |
| LSTM | 0.1485 | 165 |
| AR LSTM | 0.2719 | 3517 |

Table A.4: Time controlled model building result statistics

| Model | Test MAE | Train time (in s) | Epochs |
|---|---|---|---|
| Last | 0.8590 | - | - |
| Linear | 0.7088 | 378 | 10 |
| Dense | 0.7295 | 219 | 5 |
| Conv | 0.6508 | 433 | 5 |
| LSTM | 0.5778 | 287 | 5 |

Table A.5: Environment Controlled entry model building result statistics

| Model | Test MAE | Train time (in s) | Epochs | GPU |
|---|---|---|---|---|
| Last | 0.8590 | - | - | |
| LSTM | 0.5582 | 3195 | 15 | RTX 3070 |

Table A.6: Environment Controlled final model building result statistics

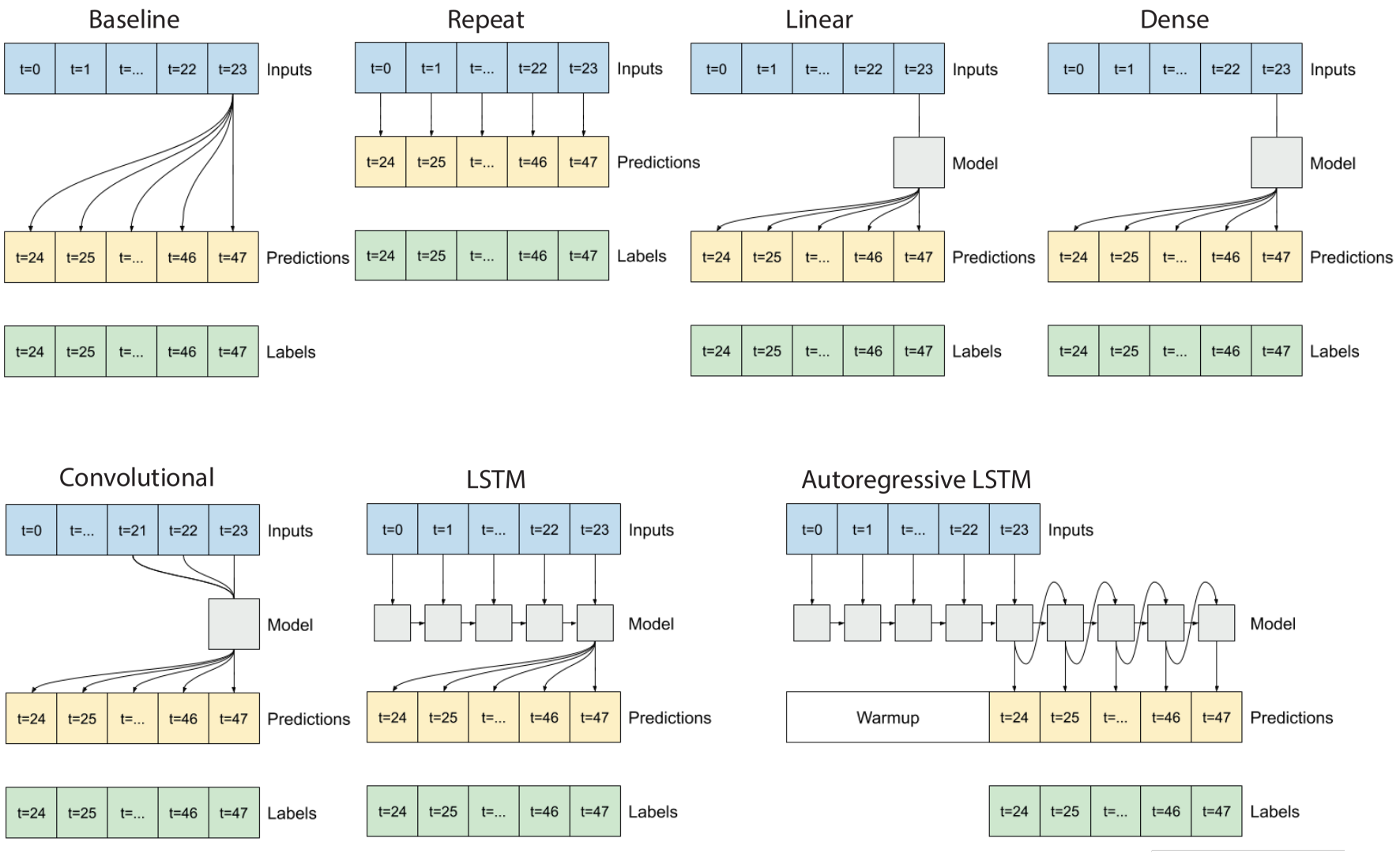



Figure A.2: General structure overview of all used model types [15]

**Linear**

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda (Lambda)              (None, 1, 183)            0
_________________________________________________________________
dense (Dense)                (None, 1, 27648)          5087232
_________________________________________________________________
reshape (Reshape)            (None, 288, 96)           0
=================================================================
Total params: 5,087,232
Trainable params: 5,087,232
Non-trainable params: 0
_________________________________________________________________
```

**Dense**

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda (Lambda)              (None, 1, 183)            0
_________________________________________________________________
dense (Dense)                (None, 1, 1024)           188416
_________________________________________________________________
dense_1 (Dense)              (None, 1, 27648)          28339200
_________________________________________________________________
reshape (Reshape)            (None, 288, 96)           0
=================================================================
Total params: 28,527,616
Trainable params: 28,527,616
Non-trainable params: 0
_________________________________________________________________
```

**Convolutional**

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda (Lambda)              (None, 288, 183)          0
_________________________________________________________________
conv1d (Conv1D)              (None, 1, 512)            26984960
_________________________________________________________________
dense (Dense)                (None, 1, 27648)          14183424
_________________________________________________________________
reshape (Reshape)            (None, 288, 96)           0
=================================================================
Total params: 41,168,384
Trainable params: 41,168,384
Non-trainable params: 0
_________________________________________________________________
```

**LSTM**

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 128)               159744
_________________________________________________________________
dense (Dense)                (None, 27648)             3566592
_________________________________________________________________
reshape (Reshape)            (None, 288, 96)           0
=================================================================
Total params: 3,726,336
Trainable params: 3,726,336
Non-trainable params: 0
_________________________________________________________________
```

Figure A.3: Model summaries for the environment controlled entry training models

**LSTM**

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 2800)              33420800
_________________________________________________________________
dense (Dense)                (None, 27648)             77442048
_________________________________________________________________
reshape (Reshape)            (None, 288, 96)           0
=================================================================
Total params: 110,862,848
Trainable params: 110,862,848
Non-trainable params: 0
_________________________________________________________________
```

Figure A.4: Model summary for the final LSTM environment controlled model

| Name | MAE LSTM | MAE Last | Δ |
|---|---|---|---|
| FEG PAR LIGHT 2 | 0.291 | 0.93 | 0.64 |
| FEG CO2 TARGET | 0.233 | 0.86 | 0.62 |
| L4-2R | 0.291 | 0.91 | 0.62 |
| R4-4R | 0.335 | 0.95 | 0.61 |
| R4-4L | 0.364 | 0.97 | 0.60 |
| FEG TEMPERATURE TARGET | 0.232 | 0.83 | 0.59 |
| SES TEMPERATURE TARGET | 0.220 | 0.79 | 0.57 |
| FEG PAR LIGHT 1 | 0.371 | 0.94 | 0.57 |
| L3-1L | 0.418 | 0.96 | 0.55 |
| R3-4R | 0.444 | 0.98 | 0.54 |
| FEG TEMPERATURE 2 | 0.374 | 0.90 | 0.53 |
| R1-2R | 0.422 | 0.94 | 0.52 |
| L3-2L | 0.379 | 0.87 | 0.49 |
| FEG TEMPERATURE 1 | 0.430 | 0.91 | 0.48 |
| R2-4L | 0.465 | 0.94 | 0.48 |
| R4-2L | 0.447 | 0.92 | 0.47 |
| R3-4L | 0.508 | 0.98 | 0.47 |
| L1-2L | 0.489 | 0.95 | 0.46 |
| R2-2L | 0.483 | 0.93 | 0.44 |
| R1-2L | 0.495 | 0.94 | 0.44 |
| L4-3R | 0.434 | 0.87 | 0.44 |
| L4-1R | 0.444 | 0.87 | 0.43 |
| R4-2R | 0.473 | 0.89 | 0.42 |
| L4-2L | 0.419 | 0.83 | 0.41 |
| FEG TEMPERATURE PHM 1 | 0.544 | 0.94 | 0.40 |
| L3-1R | 0.490 | 0.85 | 0.36 |
| R2-4R | 0.557 | 0.90 | 0.34 |
| R2-2R | 0.570 | 0.91 | 0.34 |
| R1-4L | 0.578 | 0.91 | 0.33 |
| R1-4R | 0.585 | 0.92 | 0.33 |
| L2-1R | 0.483 | 0.81 | 0.33 |
| L4-1L | 0.505 | 0.82 | 0.31 |
| L1-4L | 0.592 | 0.90 | 0.30 |
| L3-2R | 0.524 | 0.82 | 0.30 |
| L2-4R | 0.533 | 0.83 | 0.29 |
| L3-4L | 0.536 | 0.83 | 0.29 |
| L1-2R | 0.630 | 0.92 | 0.29 |
| L4-4R | 0.525 | 0.82 | 0.29 |
| L2-2R | 0.512 | 0.80 | 0.29 |
| L4-3L | 0.537 | 0.82 | 0.28 |
| L2-1L | 0.540 | 0.82 | 0.28 |
| L2-4L | 0.552 | 0.82 | 0.26 |
| L2-2L | 0.533 | 0.78 | 0.25 |
| TEMP 1 TANK 2 | 0.341 | 0.57 | 0.23 |
| L2-3R | 0.613 | 0.84 | 0.22 |
| TEMP 2 TANK 2 | 0.345 | 0.57 | 0.22 |
| L2-4 R | 0.672 | 0.89 | 0.22 |
| L3-4R | 0.540 | 0.76 | 0.22 |

| Name | MAE LSTM | MAE Last | Δ |
|---|---|---|---|
| NDS-OZONE GENERATOR | 0.518 | 0.73 | 0.21 |
| L4-4L | 0.578 | 0.78 | 0.20 |
| FEG TEMPERATURE AMS 1 | 0.697 | 0.86 | 0.16 |
| VOLUME TANK 2 | 0.628 | 0.78 | 0.16 |
| LEVEL SENSOR TANK 2 | 0.631 | 0.79 | 0.16 |
| L2-3L | 0.637 | 0.78 | 0.15 |
| FEG CALCULATED VPD 1 | 0.663 | 0.77 | 0.10 |
| SES TEMPERATURE 1 | 0.520 | 0.60 | 0.08 |
| TEMP 2 TANK 1 | 0.561 | 0.62 | 0.05 |
| SES PAR LIGHT 1 | 0.228 | 0.27 | 0.04 |
| TEMP 1 TANK 1 | 0.570 | 0.61 | 0.04 |
| FEG HUMIDITY AMS 2 | 0.868 | 0.90 | 0.03 |
| LEVEL SENSOR TANK 1 | 0.806 | 0.83 | 0.02 |
| VOLUME TANK 1 | 0.806 | 0.83 | 0.02 |
| AMS-FEG-HEATER AIR | 0.733 | 0.75 | 0.02 |
| L3-3L | 0.817 | 0.83 | 0.02 |
| L3-3R | 0.758 | 0.77 | 0.01 |
| SES CALCULATED VPD 1 | 0.447 | 0.43 | -0.01 |
| FEG CALCULATED VPD 2 | 0.702 | 0.68 | -0.02 |
| NDS-A DOSING PUMP | 0.210 | 0.14 | -0.07 |
| NDS-B DOSING PUMP | 0.210 | 0.14 | -0.07 |
| EC 1 TANK 2 | 0.440 | 0.37 | -0.08 |
| pH 1 TANK 2 | 0.499 | 0.42 | -0.08 |
| SUBFLOOR CPO 1 | 0.312 | 0.20 | -0.11 |
| TCS TEMP OUT WINDOW | 0.555 | 0.45 | -0.11 |
| SES HUMIDITY 1 | 0.576 | 0.46 | -0.12 |
| NDS-C DOSING PUMP | 0.327 | 0.21 | -0.12 |
| NDS-D DOSING PUMP | 0.327 | 0.21 | -0.12 |
| NDS-SOLENOID FW TANK 2 | 0.275 | 0.14 | -0.13 |
| NDS-SOLENOID FW TANK 1 | 0.251 | 0.11 | -0.14 |
| pH 2 TANK 2 | 0.545 | 0.40 | -0.14 |
| EC 2 TANK 2 | 0.479 | 0.32 | -0.16 |
| TCS TEMP OUT EXIT | 0.622 | 0.42 | -0.20 |
| FEG HUMIDITY 1 | 0.754 | 0.54 | -0.22 |
| FEG HUMIDITY PHM 1 | 0.858 | 0.64 | -0.22 |
| FEG CO2 1 | 0.937 | 0.71 | -0.23 |
| FEG CO2 2 | 0.942 | 0.70 | -0.24 |
| FEG TEMPERATURE AMS 2 | 1.098 | 0.85 | -0.24 |
| SES DOOR STATUS | 0.426 | 0.18 | -0.25 |
| SES CO2 1 | 0.976 | 0.73 | -0.25 |
| NDS-ACID DOSING PUMP | 0.369 | 0.10 | -0.27 |
| FEG HUMIDITY 2 | 0.813 | 0.51 | -0.30 |
| FEG OXYGEN | 0.734 | 0.40 | -0.33 |
| EC 1 TANK 1 | 0.817 | 0.33 | -0.48 |
| FEG HUMIDITY AMS 1 | 1.072 | 0.57 | -0.50 |
| EC 2 TANK 1 | 0.860 | 0.29 | -0.57 |
| pH 2 TANK 1 | 2.156 | 0.27 | -1.89 |
| pH 1 TANK 1 | 2.276 | 0.25 | -2.03 |

Table A.7: Performance of the LSTM environment controlled model: sorted descending on improvement Δ over the Last model

All files can be found in the Git-Repository: https://unconsciou5.github.io/EDEN_ISS_NN/

`build_dataset.py`

Load every subsystem with european notations and parse date-time.

```
dataset_GMTF = pd.read_csv(url_GMTF, dayfirst=True,
    parse_dates=[['Date', 'Time']], index_col="Date_Time", na_values='NOTVerified',
    comment='\t', sep=';', skipinitialspace=True, low_memory=False, decimal=',' , thousands='.')
```

Round the date-time to flat 5 min steps and merge every subsystem dataset to an empty time-frame.

```
datetime_GMTF = pd.to_datetime(dataset_GMTF.index)
datetime_GMTF = pd.DatetimeIndex(((datetime_GMTF.asi8/(1e10*30)).round()*1e10*30).astype(np.int64))  # 5 min steps
dataset_GMTF.index = datetime_GMTF

dataset_GMTF = dataset_GMTF[~dataset_GMTF.index.duplicated(keep='first')]  # remove duplicate indices
del dataset_GMTF['SampleNumber'] # delete SampleNumber

dataset_GMTF = pd.concat([timeframe, dataset_GMTF], axis=1, sort=False, join='outer')
```

Interpolate data holes linear and remove the head and tail that cant get filled.

```
dataset_GMTF.astype('float64')
dataset_GMTF.interpolate(method='linear', limit_direction='forward', axis=0, inplace=True)
dataset_GMTF = dataset_GMTF.dropna()
```

Combine all subsystems with the inner set and cut the timespan.

```
dataset = pd.concat([dataset_GMTF, dataset_AMS, dataset_ILS, dataset_NDS], axis=1, join='inner')

from_datetime = pd.to_datetime('01.03.2018',dayfirst=True)
to_datetime = pd.to_datetime('13.08.2018',dayfirst=True)

dataset = dataset.loc[from_datetime:to_datetime]
```

Remove constants and corrupted values which got identified at A.1 (not displayed here due to size).

Filter the dataset according to A.2 and save them individually into .csv files.

```
ILS_time_controlled = ['L1-2L BLUE','L1-2R BLUE','L1-4L BLUE','L1-4R BLUE','R4-4R BLUE','R4-4L BLUE','L2-1L BLUE','L2-1
identified_columns = ['SES TEMPERATURE TARGET','FEG CO2 TARGET','FEG TEMPERATURE TARGET','NDS-OZONE GENERATOR','FEG PAR
time_controlled_columns = ILS_time_controlled + identified_columns

dataset_time_controlled = dataset[time_controlled_columns]
dataset_environment_controlled = dataset.drop(time_controlled_columns, axis=1)

dataset.to_csv('datasets\dataset_full.csv', sep=';', encoding='utf-8')
dataset_time_controlled.to_csv('datasets\dataset_time_controlled.csv', sep=';', encoding='utf-8')
dataset_environment_controlled.to_csv('datasets\dataset_environment_controlled.csv', sep=';', encoding='utf-8')
```

`EDEN_ISS_NN.py`

Load the full dataset, created by build_dataset.py and set date-time as index.

```
df = pd.read_csv('datasets/dataset_full.csv', parse_dates=['Date_Time'], index_col="Date_Time", sep=';')
```

Perform continuous data separation into 70% training, 20% validation and 10% test.

```
n = len(df)
train_df = df[0:int(n*0.7)]
val_df = df[int(n*0.7):int(n*0.9)]
test_df = df[int(n*0.9):]
```

Scaling the data by standardizing.

```
train_df = (train_df - train_df.mean()) / train_df.std()
val_df = (val_df - train_df.mean()) / train_df.std()
test_df = (test_df - train_df.mean()) / train_df.std()
```

Creating a window based dataset which is holding inputs and labels (true values to predict).

```
multi_window = WindowGenerator(input_width=IN_STEPS,          // WindowGenerator Class, make_dataset function
                               label_width=OUT_STEPS,         // & split_window function needed
                               shift=OUT_STEPS,
                               label_columns=OUT_FEATURES)    // IN_STEPS = 288 , OUT_STEPS = 288
```

Defining model instances for group testing.

```
class MultiStepLastBaseline(tf.keras.Model):                                                    // Last
  def call(self, inputs):
    return tf.tile(inputs[:, -1:, :], [1, OUT_STEPS, 1])
class RepeatBaseline(tf.keras.Model):                                                           // Repeat
  def call(self, inputs):
    return inputs
multi_linear_model = tf.keras.Sequential([                                                      // Linear
      tf.keras.layers.Lambda(lambda x: x[:, -1:, :]),
      tf.keras.layers.Dense(OUT_STEPS*num_output_features, kernel_initializer=tf.initializers.zeros),
      tf.keras.layers.Reshape([OUT_STEPS, num_output_features]) ])
multi_dense_model = tf.keras.Sequential([                                                       // Dense
      tf.keras.layers.Lambda(lambda x: x[:, -1:, :]),
      tf.keras.layers.Dense(1024, activation='relu'),
      tf.keras.layers.Dense(OUT_STEPS*num_output_features, kernel_initializer=tf.initializers.zeros),
      tf.keras.layers.Reshape([OUT_STEPS, num_output_features]) ])
CONV_WIDTH = IN_STEPS
multi_conv_model = tf.keras.Sequential([                                                        // Convolutional
    tf.keras.layers.Lambda(lambda x: x[:, -CONV_WIDTH:, :]),
    tf.keras.layers.Conv1D(512, activation='relu', kernel_size=(CONV_WIDTH)),
    tf.keras.layers.Dense(OUT_STEPS*num_output_features, kernel_initializer=tf.initializers.zeros),
    tf.keras.layers.Reshape([OUT_STEPS, num_output_features]) ])
multi_lstm_model = tf.keras.Sequential([                                                        // LSTM
      tf.keras.layers.LSTM(128, return_sequences=False),
      tf.keras.layers.Dense(OUT_STEPS*num_output_features, kernel_initializer=tf.initializers.zeros),
      tf.keras.layers.Reshape([OUT_STEPS, num_output_features]) ])
class FeedBack(tf.keras.Model):  ...                                                            // AR LSTM
```

Defining final environment controlled model instance.

```
multi_lstm_model = tf.keras.Sequential([
      tf.keras.layers.LSTM(2048, return_sequences=False, dropout=0.1, kernel_regularizer=regularizers.l2(0.00002)),
      tf.keras.layers.Dense(OUT_STEPS*num_output_features,
                            kernel_initializer=tf.initializers.zeros,kernel_regularizer=regularizers.l2(0.00001)),
      tf.keras.layers.Reshape([OUT_STEPS, num_output_features]) ])
```

Compile and fit function under early_stopping callback, Adam optimizer, MSE losses and MAE metrics.

```
def compile_and_fit(model, window, patience=None):
  early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=4,
                                                    mode='min', restore_best_weights=True)
  model.compile(loss=tf.losses.MeanSquaredError(),
                optimizer=tf.optimizers.Adam(),
                metrics=[tf.metrics.MeanAbsoluteError()])
  history = model.fit(window.train, epochs=MAX_EPOCHS, validation_data=window.val, callbacks=[early_stopping])
  return history
```

Call compile_and_fit, save the weights and evaluate the MAE on the validation- and test-data.

```
history = compile_and_fit(multi_lstm_model, multi_window)                              // for all models
multi_lstm_model.summary()
multi_lstm_model.save_weights('./models/' + model_target + model_type + '/LSTM')
multi_val_performance['LSTM'] = multi_lstm_model.evaluate(multi_window.val)
multi_performance['LSTM'] = multi_lstm_model.evaluate(multi_window.test)
```

Evaluate the individual MAE from an example batch of the test data frame.

```
def individual_MAE(self, model=None):
    inputs, labels = self.example
    predictions = model(inputs)
    multi_batch_MAE = pd.DataFrame()
    for x in range(predictions.shape[0]):
        single_batch_predictions = predictions[x,:,:]
        single_batch_labels = labels[x,:,:]
        single_batch_MAE = pd.DataFrame(sklearn.metrics.mean_absolute_error(
                            y_true=single_batch_labels, y_pred=single_batch_predictions, multioutput='raw_values'))
        single_batch_MAE = single_batch_MAE.transpose()
        multi_batch_MAE = multi_batch_MAE.append(single_batch_MAE, ignore_index=True)
    MAE = multi_batch_MAE.mean(axis=0)
    index_names = pd.DataFrame.from_dict(self.label_columns_indices, orient='index')
    MAE.index = index_names.index
    return MAE
individual_MAE_df = multi_window.individual_MAE(multi_lstm_model)
individual_MAE_df.to_csv('./models/' + model_target + model_type + '/MAE_LSTM.csv', sep=';', encoding='utf-8')
```

Plot the training history, the key features and all features with their inputs, labels and predictions.

```
plotter = tfdocs.plots.HistoryPlotter(metric = 'mean_absolute_error')                  // history
plotter.plot(history)
plt.savefig('./models/' + model_target + model_type + '/fig/history_LSTM.svg')
for x in range(0,len(KEY_FEATURES)):                                                   // key features
      multi_window.plot(multi_lstm_model, plot_col=KEY_FEATURES[x])
      plt.savefig('./models/' + model_target + model_type + '/fig/LSTM_key_' + str(x) +'.svg')
for x in range(0,len(environment_controlled_features)):                                // all features
      multi_window.plot(multi_lstm_model, plot_col=environment_controlled_features[x])
      plt.savefig('./models/' + model_target + model_type + '/fig/all_LSTM/LSTM_' + str(x) +'.svg')
```

## References


[1] Paul Zabel, Matthew Bamsey, Conrad Zeidler, Vincent Vrakking, Daniel Schubert, Oliver Romberg, Giorgio Boscheri, and Tom Dueck, *The preliminary design of the eden iss mobile test facility: An antarctic greenhouse*, 46th International Conference on Environmental Systems, Ed., Vienna, Austria, July 2016. [Online]. Available: https://elib.dlr.de/105476/.

[2] Paul Zabel, Matthew Bamsey, Conrad Zeidler, Vincent Vrakking, Daniel Schubert, and Oliver Romberg, *Future exploration greenhouse design of the eden iss project*, 47th International Conference on Environmental Systems, Ed., Charleston, South Carolina, USA, July 2017. [Online]. Available: https://elib.dlr.de/114876/.

[3] Daniel Schubert, Matthew Bamsey, Paul Zabel, Conrad Zeidler, and Vincent Vrakking, *Status of the eden iss greenhouse after on-site installation in antarctica*, 48th International Conference on Environmental Systems, Ed., Albuquerque, New Mexico, USA, July 2018. [Online]. Available: https://elib.dlr.de/121867/1/ICES-2018-140.pdf.

[4] Paul Zabel, Conrad Zeidler, Vincent Vrakking, Markus Dorn, and Daniel Schubert, *Biomass production of the eden iss space greenhouse in antarctica during the 2018 experiment phase*, EDEN Research Group, Ed., May 2020. [Online]. Available: https://doi.org/10.3389/fpls.2020.00656.

[5] Vincent Vrakking, Conrad Zeidler, Paul Zabel, Markus Dorn, and Daniel Schubert, *Status and future of the eden iss mobile test facility*, 50th International Conference on Environmental Systems, Ed., July 2020. [Online]. Available: https://elib.dlr.de/135984/.

[6] B. Boss, S. Malakuti, S.-W. Lin, T. Usländer, E. Clauer, M. Hoffmeister, L. Stojanovic, and B. Flubacher, *Digital twin and asset administration shell concepts and application in the industrial internet and industrie 4.0: An industrial internet consortium and plattform industrie 4.0 joint whitepaper*, Industrial Internet Consortium, Ed., September 2020. [Online]. Available: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Digital-Twin-and-Asset-Administration-Shell-Concepts.html.

[7] F. Tao, F. Sui, A. Liu, Q. Qi, M. Zhang, B. Song, Z. Guo, S. C.-Y. Lu, and A. Y. C. Nee, "Digital twin-driven product design framework," *International Journal of Production Research*, February 2018. DOI: 10.1080/00207543.2018.1443229.

[8] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn, *Digital twin in manufacturing: A categorical literature review and classification*, August 2018. [Online]. Available: https://doi.org/10.1016/j.ifacol.2018.08.474.

[9] Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow, *Digital twin: Enabling technologies, challenges and open research*, IEEE Access, Ed., June 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2998358.

[10] M. Czupalla, A. Zhukov, J. Schnaitmann, C. Olthoff, M. Deiml, P. Plötner, and U. Walter, "The virtual habitat – a tool for dynamic life support system simulations," *Advances in Space Research*, June 2015, ISSN: 0273-1177. [Online]. Available: https://doi.org/10.1016/j.asr.2014.12.029.

[11] Michael Nielsen, *Neural networks and deep learning*, December 2019. [Online]. Available: http://neuralnetworksanddeeplearning.com/index.html.

[12] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, December 2014. [Online]. Available: https://arxiv.org/abs/1412.6980.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Dropout: A simple way to prevent neural networks fromoverfitting*, Journal of Machine Learning Research 15, Ed., January 2014. [Online]. Available: https://dl.acm.org/doi/10.5555/2627435.2670313.

[14] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *Design Automation Conference*, C. Brodley, Ed., New York, N.Y. and Piscataway, N.J.: Association for Computing Machinery and IEEE, July 2004, p. 78. [Online]. Available: https://doi.org/10.1145/1015330.1015435.

[15] Tensorflow Documentation, *Time series forecasting: Multi-step models*, February 2021. [Online]. Available: https://www.tensorflow.org/tutorials/structured_data/time_series#multi-step_models.

[16] Vincent Vrakking, Matthew Bamsey, Conrad Zeidler, Paul Zabel, Daniel Schubert, and Oliver Romberg, *Service section design of the eden iss project*, 47th International Conference on Environmental Systems, Ed., Charleston, South Carolina, USA, July 2017. [Online]. Available: https://elib.dlr.de/118462/.

[17] G.Bonzano and V.Vrakking, *D3.7–environmental control design document*, September 2017. [Online]. Available: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5c39e70ec&appId=PPGMS.

[18] Karin Dankis, *Lighting system design document*, September 2017. [Online]. Available: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5b56f99ec&appId=PPGMS.

[19] M. Stasiak and M. Dixon, *D3.5 – nds final design document*, September 2017. [Online]. Available: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5b56f8371&appId=PPGMS.

[20] K. G. Sheela and S. N. Deepa, "Review on methods to fix number of hidden neurons in neural networks," *Mathematical Problems in Engineering*, May 2013. [Online]. Available: https://doi.org/10.1155/2013/425740.

[21] Tensorflow Documentation, *Strategies to prevent overfitting*, February 2021. [Online]. Available: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit#combined_l2_dropout.