

Machine Learning on Simulated and Real Farm Data based on an Ontology-Controlled Data Infrastructure

Alexander Muenzberg¹, Christian Troost², Daniel Martini³, Francisco Mendoza², Rajiv Srivastava², Thomas Berger², Liv Seuring³, Nils Reinosch³, Thilo Streck² and Ansgar Bernardi¹

¹German Research Centre for Artificial Intelligence (DFKI), Trippstadter Str. 122, 67663 Kaiserslautern, Germany

²University of Hohenheim, Schloss Hohenheim 1, 70599 Stuttgart, Germany

³Association for Technology and Structures in Agriculture (KTBL), Bartningstr. 49, 64289 Darmstadt, Germany

Abstract

The SimLearn project uses semantic information processing and machine learning to combine process-model simulation data with real-world farm data in order to support decision making in agriculture. This paper describes a system architecture for high-performance extraction, storage, processing and machine learning of data extracted from simulation models (Big Data) or collected from farmers. Ontologies are used to link data to expert knowledge. Neural networks are pre-trained on simulation output to be later employed in prediction of decision-relevant outcomes (e.g. crop growth, soil mineralization) after transfer learning from observed farm data.

Keywords

Farmer decision support, artificial neural networks, ontology-controlled systems, machine learning architecture, big data, agent-based bioeconomic modelling

1. Introduction


Machine learning can be used as a digital tool for handling big datasets and such learning methods are becoming increasingly important in agriculture to minimize the losses and enhance the profit of farmers by providing precise recommendations for crop management [1]. The potential for the application of machine learning models is, however, often limited by data for specific farm conditions, which is often fragmented and typically not comprehensive enough to support generalization towards practically relevant innovative crop management variation or expected climatic changes. At the same time, theory-based agricultural models incorporate generic structural domain knowledge that can support generalization beyond observed conditions, but parameterizing them to provide reliable and detailed predictions for the conditions of a specific farm remains challenging. As described by Berger et al. (2020) [2], the Project

In A. Martin, K. Hinkelmann, H.-G. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (Eds.), *Proceedings of the AAAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence (AAAI-MAKE 2022)*, Stanford University, Palo Alto, California, USA, March 21–23, 2022.

✉ alexander.muenzberg@dfki.de (A. Muenzberg); christian.troost@uni-hohenheim.de (C. Troost); d.martini@ktbl.de (D. Martini); f.mendoza@uni-hohenheim.de (F. Mendoza); rajiv.srivastava@uni-hohenheim.de (R. Srivastava); thomas.berger@uni-hohenheim.de (T. Berger); l.seuring@ktbl.de (L. Seuring); n.reinosch@ktbl.de (N. Reinosch); thilo.streck@uni-hohenheim.de (T. Streck); ansgar.bernardi@dfki.de (A. Bernardi)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

SimLearn, funded by the German Federal Ministry of Education and Research, explores new approaches to support the decision-making of farmers by combining machine learning and theory-based simulation modeling, in which neural networks are first trained on large sets of simulation outputs generated by process-based simulation models to implicitly learn the theoretical domain knowledge incorporated in these models. Subsequent transfer learning with more fragmented, locally-specific farm data is hypothesized to finally yield more reliable predictors that can anticipate the effects of innovative farm management strategies e.g. for timing and intensity of fertilization or tillage [2]. In effect, the use of simulation-generated data for initial training overcomes the lack of sufficiently manifold real world farm data, while model adaptation by transfer learning eases the fine-tuning towards the particularities of an individual farm.

This article describes the SimLearn system architecture and how it technically combines ontologies with knowledge from agricultural data, Big Data technologies, simulation model data, and machine learning processes. Our approach bears some similarities with the architecture of Klose (2020), who outlined how to combine ontologies of scientific data and Big Data technologies [3]: In their approach, agricultural raw data were semantically processed for the three representations geo-information, time series and semantic facts (an agricultural knowledge graph) which, due in part to its size, resided in the NoSQL Database Cassandra, in a semantically pre-processed form. Knowledge was made available via a specifically developed platform [3].

While the focus of Klose (2020) was on semantically processing the real data of farmers and sensor data and making them available [3], our approach is targeted at collecting agricultural simulation results and measurement data, semantically process it, and enable predictions using modern machine learning (ML) methods. Simulation output of theory-based models is integrated into a high performance storage infrastructure. Simulation metadata is linked to knowledge derived from data provided by the Association for Technology and Structures in Agriculture (KTBL) and represented using a knowledge graph in an ontology. An automatized machine learning workflow is established for efficient learning of time series of high dimensional multivariate agricultural system states.

2. Data Basis

In SimLearn, machine learning models are trained on simulation output of the process and expert-based modeling system MPMAS_XN. In a second step, they will be retrained on actual farm data to achieve transfer learning. MPMAS_XN is a combination of the Mathematical Programming-based Multi-Agent Systems model (MPMAS), which calculates optimal farm production and investment plans and dynamically simulates their economic outcomes over time and the soil-plant simulator Expert-N (XN), which simulates crop growth and soil processes over time, including the actual yields obtained through the production plans selected by MPMAS [4, 5].

As external inputs, MPMAS_XN relies i.a. on a regularly updated KTBL expert dataset with technical characteristics of over 5000 field operations and 2400 machine and equipment types, daily time series of observed or simulated weather data, and multidimensional soil profile data.

MPMAS_XN generates different types of output including (A) farm-level multivariate data in relational data structures (e.g. optimal production and investment plans for a given farm) and (B) plot-level time series, i.e. multivariate daily-resolution time series of plant status (biomass, phenology, maturity, nitrogen contents, etc.), soil status (moisture, carbon stocks, nitrogen, etc.) adapted in-season management (field work dates, fertilization quantities) and final yields for a given plant management plan, initial soil status and exogenous weather time series (among others temperature, precipitation, humidity and wind speed).

For the initial development of the data and ML architecture described in this article about 20,000 individual plot-level time series (data type B), each with about 400 variables and daily time steps for more than 100 years were used.

3. Methods

3.1. Ontology

Semantic processing of the information provided by the data makes it easier for the machine to interpret the data. The semantics are represented in the form of a knowledge graph using the Resource Description Framework (RDF). The knowledge graph (hereafter also called ontology) describes the format and structure of the agricultural data. The RDF knowledge graph is stored as triple syntax (subject -> predicate -> object) in Turtle (Terse RDF Triple Language) files (in .ttl file format). The knowledge of the graph can be queried using the query language Sparql [6]. Relationships among the data can be queried through the ontology, such as fertilization steps related to a specific crop and soil type. Among others, this information can be used to explain parameters via a user interface. Additional technical information is annotated to classes and properties in the ontology [3]. These are, among others, properties of the data that define, for example, which data correlate with each other and thus serve as input and output for ML models. In addition to the ML model information, information is annotated to control the Extract, Transform, Load (ETL) processes. For example, if changes are made to parameters required for machine learning processes, the properties and connections within the ontology must be adjusted, but not the program code. This enables dynamic and automatic adaptation based on the data stored in the ontologies. The ontology is thus the knowledge base on which a controller in the system architecture (see section 4.1) makes decisions about which data parameters to use for decision making. The ontology includes parts of the knowledge base of the GeoBox project, which looks at agriculture from the perspective of a critical system infrastructure and pursues the goal of resilient system architecture, high security, data sovereignty and usability [7] and introduced the concept of a semantically annotated agricultural activity log for recording farm data.

3.2. Big Data Technology

In the course of database development, various solutions for data storage were assessed. Especially for the large amounts of time series data, different requirements for the temporal performance when storing and reading the data had to be considered. Most of the simulation data is delivered in comma separated or tab separated value format. For each simulated farm plot

there are separate data files and each line in these files represents a time series with parameters (described in section 2) related to the plot. Furthermore, the data is mapped with additional information that is contained in configuration files (e.g. .ini files). The files are read into virtual data tables (so-called data frames). Both the sequential writing of the large time series tables and the writing of complete data batches in conventional relational databases (SQL databases) would take an enormous amount of time.

There are special NOSQL (not only SQL) or NOSQL-like solutions for storing time series data. TimescaleDB has advantages here because it is based on the relational database technology PostgreSQL and maintains a clear relational structure with fewer redundancies. Furthermore, the comfortable and well-known query language SQL is used by many developers [8]. Nevertheless, the write performance has limits with very large data sets in the field of CPU and disk usage [9]. The time series database VictoriaMetrics offers very performant read and write capabilities for measurements, but is subject to the limitation of storing only current or future data streams [10]. Past simulated data cannot be loaded with this solution due to the limitations. Another option for using NOSQL databases for storing time series data sets is the wide-column storage Cassandra. This database offers very good read and write performance for single-data rows [11, 12]. Nevertheless, Cassandra has limitations in the case of storing large data batches [13]. Even reading such batches with different keys causes problems for the database. As a database solution for data streams with unique IDs Cassandra is an excellent and performant solution [11].

The potential solutions discussed above have the capabilities to store data distributed across multiple compute clusters to increase storage and compute resources available for processing. Furthermore, there are also so-called data lakes that store data in a semi-structured manner. The Hadoop Distributed File System (HDFS) is such a storage solution and also offers distributed storage across computing clusters, as well as a high read and write performance of timeseries data batches [14].

In order to achieve good performance both when storing time series as batches and when sequentially streaming single records, as well as real-time analysis of the data, the Lambda architecture described by Marz (2015) is used [15]. The Lambda architecture consists of a speed layer (for high-performance data processing), a serving layer for structured data provision and a batch layer for batch storage of large data [16]. For serving and batch layers, the Cassandra and HDFS technologies explained above are used. The Spark Framework from the Apache Foundation is used as the speed layer technology. The framework is used in the Python programming language (named PySpark). PySpark allows high-performance processing of data in the form of resilient distributed datasets (RDD) and data frames in the main memory of one or more distributed computer clusters [17]. PySpark helps with the ETL process and with the preparation of the data for the machine learning process.

3.3. Machine Learning Models

Using modern deep learning approaches, regressions are performed to predict time series related to the data in section 2. Convolutional Neural Networks (CNN) perform well for predicting large amounts of time series data. Through the convolutional layer, fixed time periods are sampled and dimensioned, which can ultimately be subjected to prediction using regression [18, 2].

The CNNs used here are built on simple deep neural networks (DNN), such as the multi-layer perceptron (MLP). The MLP consists of many perceptrons deeply nested in several layers. A perceptron is associated with several input values x_i and contains several weights w_i which are multiplied by the input values ($x_0w_0 + x_1w_1 + \dots + x_nw_n$). In the next step, an activation function f is applied to the multiplied values to calculate the output y of the perceptron. Possible activation functions include the rectifier linear unit function (ReLU) (1) or the sigmoid function (2) shown below.

$$f_{relu}(z) = \max(0, z) \quad (1)$$

$$f_{sigmoid}(z) = \frac{1}{1 + e^{-z}} \quad (2)$$

The calculation of the output y is described in the following (3).

$$y = f(z); z = \sum_{i=1}^n x_i w_i \quad (3)$$

Backpropagation is used to learn the weights during network training, in which their values are corrected step by step using an error calculation (e.g. the mean squared error). Here, the calculated errors are passed on from layer to layer. This is done by starting at the output and passing the error backwards layer by layer [19, 20].

In CNNs, three types of layers are applied. These are the convolutional layer, the pooling layer, and the fully-connected layer. The convolutional layer pools regions of input parameters that are connected to specific perceptrons (artificial neurons). Through the pooling layer, downsampling is performed along the spatial dimensionality of the layer input [18]. Thus, the number of input parameters is reduced. The fully-connected layer performs the same task as a layer in MLP and performs classification, or regression based on the layer input. These three types of layers are sequentially connected in the CNN [18].

In our approach, a CNN is used because the convolutional mechanisms can reduce the large dimensions or feature maps created by the large number of parameters over many years and the CNN is more performant in its prediction of Big Data sets than conventional neural networks for time series prediction [21, 22]. To predict the temporal development of soil and plant status (e.g., plant growth status) over a period of time, a multi-step prediction is performed by the CNN. Multi-step prediction (recursive prediction) means that the 5th day is predicted on the basis of start parameters and start conditions from day 1 to 4. Furthermore, the 6th day is predicted on the basis of the days 2 to 5 (whereby the prediction of the 5th day is included in the network input). Then from day 3 to 6 the 7th day is predicted, from day 4 to 7 the 8th day and so on. The predictions are performed recursively for as many days until the maximum number of prediction days specified by the user is reached. Figure 1 graphically illustrates the process of multi-step prediction (adapted from [23]).

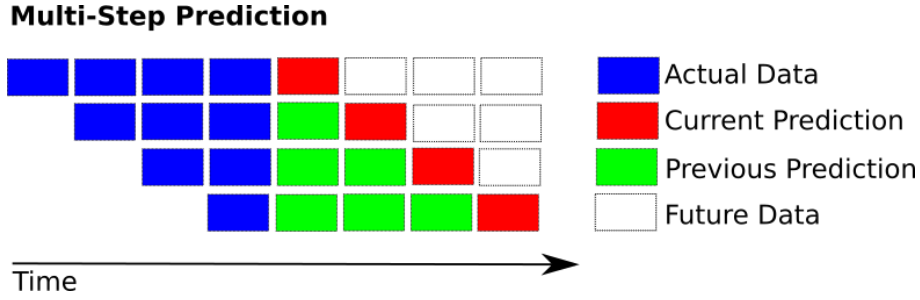


Figure 1: Process of 4-day ahead multi-step prediction (4 days respectively) [23]

4. Results

4.1. System Architecture Approach

The system architecture links a web view that enables human-machine interaction with controllers that are linked to the ontologies as a knowledge base to dynamically access ETL or data processes and ML processes. By adapting the ontologies, the ML processes can be dynamically modified and extended.

The architecture combines the described Lambda architecture, a Model-View-Controller architecture (MVC) and microservices for ETL and ML processes. The encapsulation of data model, web view, controller and the deployment as microservices allows an independent and dynamic implementation of the processes. Parts of the architecture can be changed and only have to comply with interface specifications without affecting other parts of the architecture. Furthermore, independent implementation by respective experts (database developers, software developers, ML developers and Web View developers) is possible. Figure 2 represents an abstraction of the system architecture graphically.

In summary, the architecture connects different modern approaches to encapsulation, distributed development and best practices in the area of big data. These approaches are combined with knowledge-based controlling of the processes. The human component influences the knowledge base, represented as an ontology, and the triggering of the processes via API control.

4.2. RDF Knowledge Graph (Ontology)

In the ontology files used, relationships between individual properties of plots, soil conditions, crop type, weather, climate, production and management processes, among others, are evident. Figure 3 shows an exemplary excerpt of the ontology in a slightly modified and simplified VOWL notation [24]. Classes are shown as ellipses, connecting *owl : ObjectProperties* inbetween as boxes. Boxes filled in grey denote datatypes of *owl : DatatypeProperties*. It shows the relation of soil clay, silt and sand proportions to the class representing soil texture as well as the linkage of the latter to the soil as a whole and to the plot. The soil itself can have further properties. As an example, *organicCarbonContent* is shown. Likewise for the plot: the figure illustrates the plot size as one of several additional properties of that class. We use a common naming convention in this ontology, where instances of a class can be linked by properties of

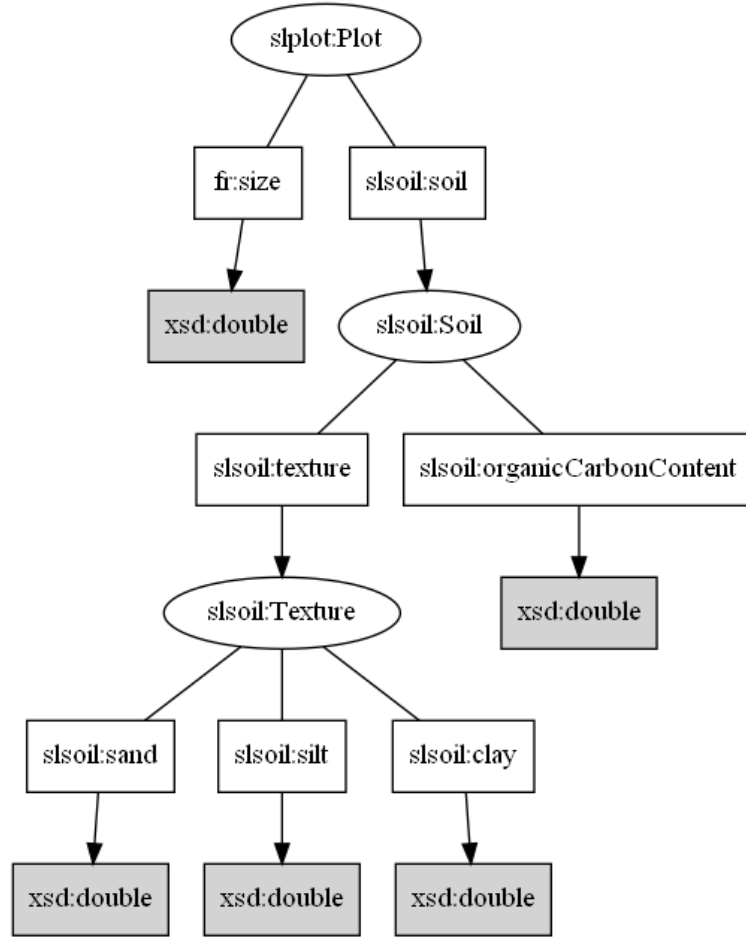


Figure 3: Excerpt of the SimLearn Ontology representing soil texture and several soil and plot properties

cnnTimeseriesPredictionValue is thus a technical property that is inherited from relevant parameter properties. The SimLearn architecture contains such and similar properties for controlling ML processes.

4.3. Convolutional Neural Network (CNN)

Using a CNN, we trained data sets of 320 randomly selected simulated plots (from a pool of 5000 simulated plots) in a multi-step procedure. The simulated plots contain time series data sets over 100 years each (from 1980 to 2080). Subsequently, 256 plots were predicted. For the prediction, the first 4 simulation days were entered as network input and then in sequential steps, in which the predicted data sets were repeatedly included, a total of 6 years (from 2041 to 2043 and from 2061 to 2063) were predicted. Trained and predicted plots contain a total of 5 different soil characteristics (differing in soil texture, i.e. in clay, silt and sand content but also in other variables) and 2 different crop rotations (maize and rotations between winter rape,

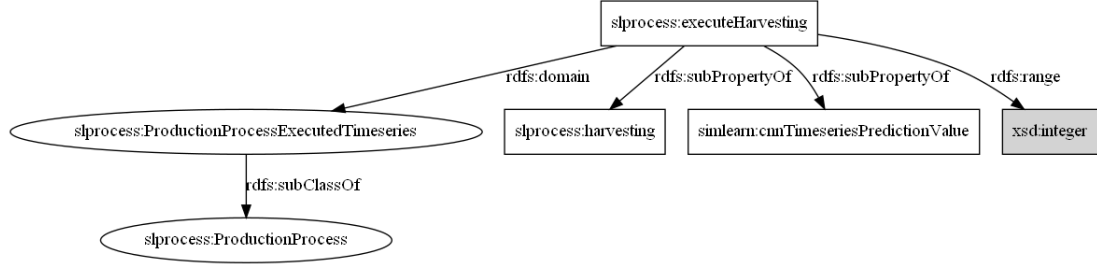


Figure 4: Section of the SimLearn Ontology representing connections of production process and ML

winter wheat, summer barley and winter barley). Furthermore, 32 different weather scenarios and 13 different management plans were included.

Figure 5 represents both the predicted curve and the actual simulated curve of the growth parameters biomass above ground, generative biomass and leaf area index (LAI). Other parameters shown are the nitrogen and water content parameters nitrogen uptake, nitrate (NO₃) content at 0-30 and 30-60 cm soil depth, water uptake, volumetric water content at 0-30 and 30-60 cm soil depth. The figure shows a section of 2 years and 10 months.

Overall, the focus was on predicting values for growth-related outcomes such as above ground biomass, generative biomass, the BBCH value (for the different development stages) and LAI; nutrient-related outcomes such as nitrogen uptake, NO₃ content at 0 to 30 and 30 to 60 cm and nitrogen concentration in fruit, leaf, and stem; water-related outcomes such as water uptake and volumetric water content at 0 to 30 and 30 to 60 cm. Figure 6 shows the root mean square errors (RMSE) between actual and predicted timeseries values for biomass (above ground), BBCH (development stage), and LAI (leaf area index) of 10 different plots. (4) shows the RMSE function with the actual timeseries values act_i and the predicted timeseries values $pred_i$.

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(act_i - pred_i)^2}{n}} \quad (4)$$

We calculated RMSE values for all the above outcomes. On the basis of the RMSE values of each of the 256 prediction plots, statistics were calculated for minimum, median, maximum, lower quartile and upper quartile. As an example, Table 1 shows the statistics of RMSE values of biomass above ground, generative biomass, BBCH, LAI nitrogen and water uptake. The different values of the variables were multiplied for the ML processes by a scaling factor which is listed in table 1.

Overall, it was determined that the plots containing single crop rotation maize data contained lower RMSE values and therefore better predictions were obtained (e.g. the 5 plots 0_5_4_0, 0_8_26_0, 0_6_18_0, 0_10_18_0 and 0_9_36_0 in figure 6 contain the crop rotation maize and have visibly the lowest RMSE values in the graphic). In addition, it was noticed that the majority of plots with high RMSE values contained a silty loam soil (Lu, Ut4) as soil texture. 29 of the 256 test plots contained significant outlier RMSE values and were therefore predicted worse than the remaining 227 plots.

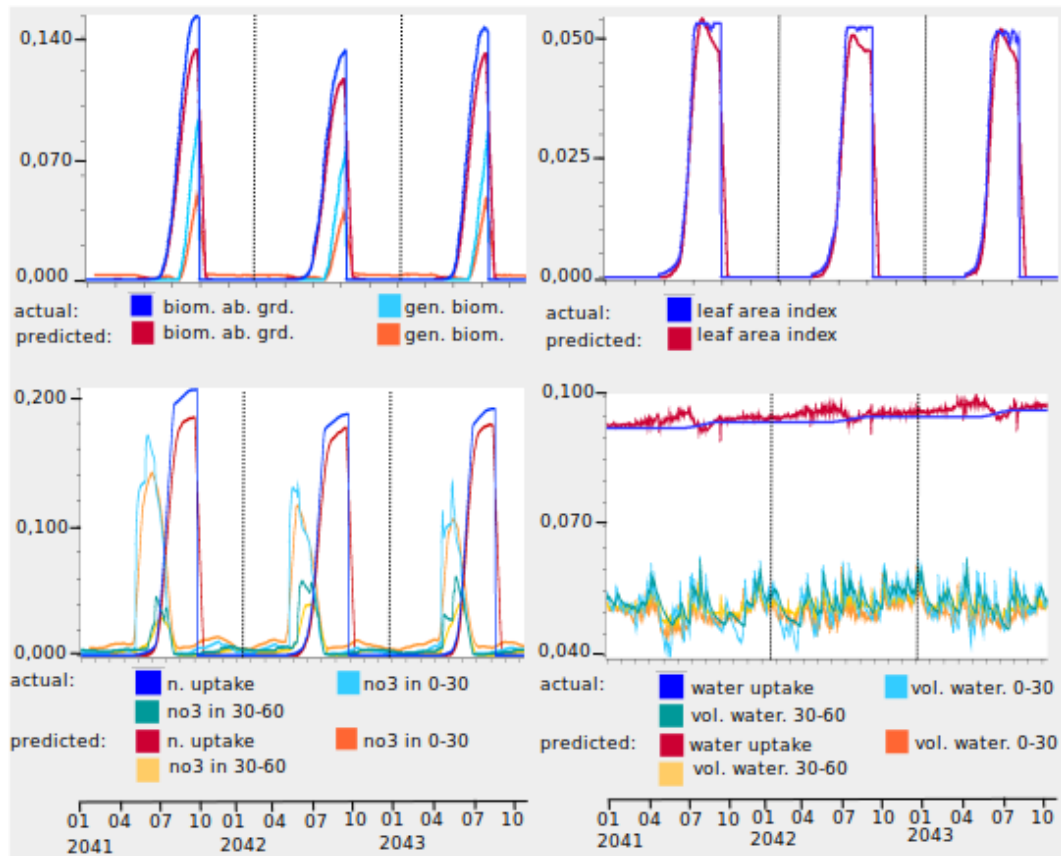


Figure 5: Predicted curve and the actual simulated curve of the growth parameters, water content parameters and nitrogen content parameters

Table 1

RMSE statistics

statistic	biom. ab. grd.	gen. biom.	BBCH	LAI	N upt.	water upt.
minimum	0.009	0.004	0.079	0.004	0.016	0.001
median	0.017	0.010	0.101	0.008	0.035	0.028
maximum	0.089	0.101	0.209	0.043	0.159	0.183
lower quartile	0.014	0.008	0.089	0.006	0.028	0.13
upper quartile	0.024	0.016	0.112	0.013	0.046	0.045
unit or value range	<i>t/ha</i>	<i>t/ha</i>	0-99	<i>m²/m²</i>	<i>kg/ha</i>	<i>m</i>
scale factor	0.01	0.01	0.01	0.01	0.01	0.01

4.4. Time performance of data processing

The currently stored number of 7,300,000 MPMAS_XN data rows including over 400 columns were stored in the databases (within the Lambda architecture). So far, satisfactory temporal performance of the data flow has been achieved by using the aforementioned Big Data tech-

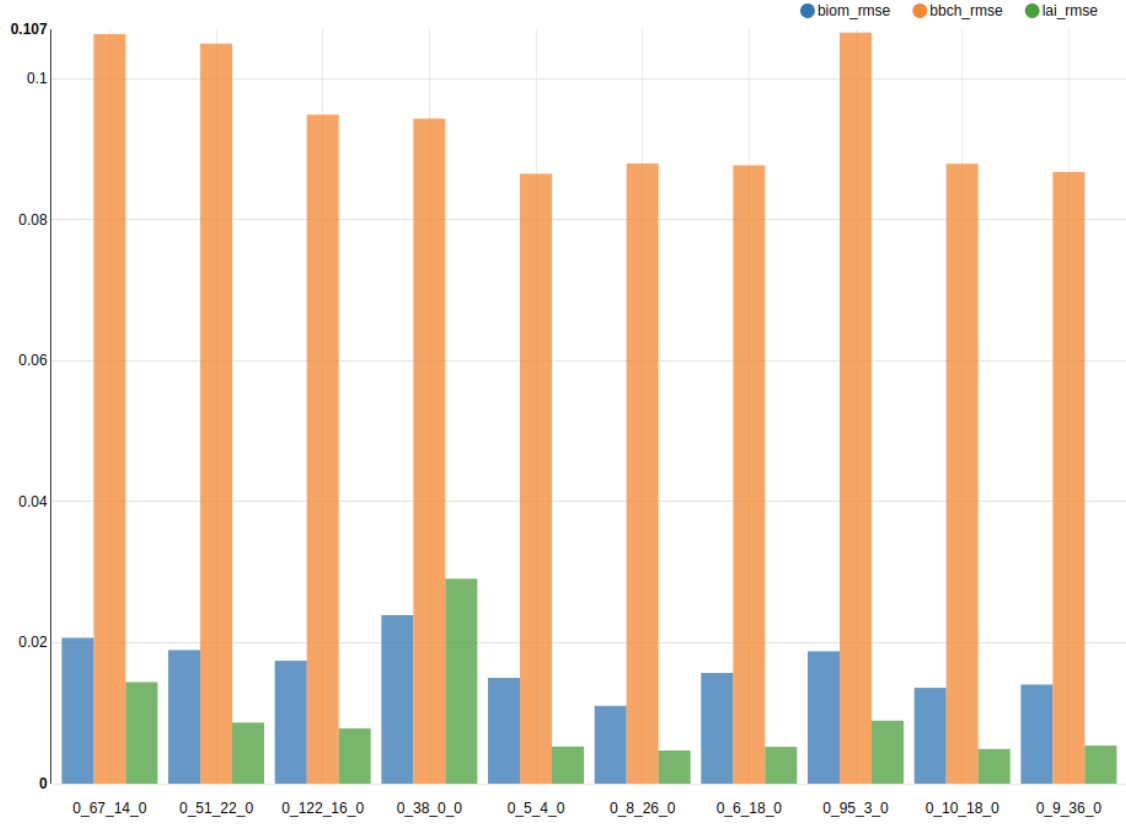


Figure 6: Bar chart of RMSE values between actual and predicted timeseries values of biomass above ground (scaled by factor 0.01 in $[t/ha]$), growth stage (BBCH value in range 0-99, scaled by factor 0.01) and leaf area index (scaled by factor 0.01 in $[m^2/m^2]$) on 10 different plots.

nologies. Maximum processing speeds of 270 seconds for the ETL process and loading into the databases and max. 25 seconds for reading out the large MPMAS_XN data sets were measured. The CNN-ML process took 140 seconds per plot and the prediction of one year is performed within 20 seconds. The tests ran on a cluster with Nvidia RTX A6000 GPU with 48 GB GPU memory, 8 GPUs and 12 CPUs per GPU.

5. Conclusion and Outlook

In this paper, we present an ontology-controlled ML architecture for agricultural decision support. The data flow is controlled by the user via an API. The processes presented in the architecture model are able to reproduce the results of the simulation models to a satisfactory degree. Satisfactory time performance of the data flow has been achieved by using a Lambda architecture as common in Big Data environments. The user-controlled data flow ran semi-automated (automated except for control interactions by the user) during tests. The ML net obtained required information via parameters from the ontologies. Upscaling of the systems for

more computing performance is possible.

Information on hyperparameter choices for the neural networks (training epochs, node density, etc.) is currently obtained via special configuration files. To extend the dynamic analysis capabilities, this information will be integrated into the SimLearn architecture ontology in the future. Additional information such as computational bases and metadata of the simulation data will be incorporated into the ontology. Future enhancements are, in addition to an extended evaluation of the simulation data by ML networks, the expansion of the ML infrastructure. Accompanying the optimization of infrastructure and ontologies, user interaction capabilities will be gradually adapted. In addition, real farmer data and machine sensor data will be incorporated into the data flow and ML networks. This is expected to lead to a significant improvement in network predictions in the project. For the system we want to collect and generate sufficient data about agricultural processes to support the farmer in decision-making in production management and yield optimization.

Acknowledgments

The research project SimLearn is funded by the German Federal Ministry of Education and Research under grant FKZ 01|S19073A. Simulations are performed using the high-performance computing resources bwHPC, funded by the Ministry of Science, Research and the Arts Baden-Württemberg, the Universities of the State of Baden-Württemberg, Germany and Deutsche Forschungsgemeinschaft (DFG) through grant INST 35/1134-1 FUGG.

References

- [1] Consultative Group on International Agricultural Research (CGIAR), Platform for big data in agriculture, 2021. URL: <https://bigdata.cgiar.org/>.
- [2] T. Berger, A. Bernardi, D. Martini, A. Münzberg, J. Parussis, T. Streck, C. Troost, Combining machine learning and simulation modelling for better predictions of crop yield and farmer income, In: A. van Griensven, J. Nossent, D. Ames (Eds.), Proceedings 10th International Congress on Environmental Modelling and Software, Brussels, Belgium (2020).
- [3] J. Klose, M. Schröder, S. Becker, A. Bernardi, A. Ruckelshausen, Datenaufbereitung in der Landwirtschaft durch automatisierte semantische Annotation - Basis für vielfältige smarte Dienste, In: 40. GIL-Jahrestagung, Campus Weißenstephan, Freising, 17.-18. Februar 2020 Referate der 40. GIL-Jahrestagung (2020).
- [4] C. Troost, X. Duan, S. Gayler, F. Heinlein, C. Klein, J. Aurbacher, M. S. Demyan, P. Högy, M. Laub, J. Ingwersen, P. Kremer, F. M. Tijerino, L. H. Otto, A. Poyda, K. Warrach-Sagi, T. K. D. Weber, E. Priesack, T. Streck, T. Berger, The bioeconomic modelling system mpmas_xn: Simulating short and long-term feedback between climate, crop growth, crop management and farm management, In: van Griensven, A., Nossent, J., Ames, D.P. (Eds.) 10th International Congress on Environmental Modelling and Software. Brussels, Belgium (2020).
- [5] C. Troost, T. Berger, Advances in probabilistic and parallel agent-based simulation: Modelling climate change adaptation in agriculture, In: Sauvage, S., Sánchez Pérez, J.-M.,

- Rizzoli, A. E. (Eds.): Proceedings of the 8th International Congress on Environmental Modelling and Software, July 10-14, Toulouse, France (2016).
- [6] W3C, Rdf 1.1 turtle, 2014. URL: <https://www.w3.org/TR/turtle/>.
 - [7] F. Kuntke, C. Reuter, W. Schneider, D. Eberz, A. Bernardi, Die geobox-vision: Resiliente interaktion und kooperation in der landwirtschaft durch dezentrale systeme, Mensch und Computer 2020 - Workshopband, C. Hansen, A. Nürnberger, B. Preim (Hrsg.), Magdeburg: Gesellschaft für Informatik e.V. (2020) 1–6. doi:10.18420/muc2020-ws117-407.
 - [8] Timescale, Postgresql + time-series, 2021. URL: <https://www.timescale.com/>.
 - [9] A. Valialkin, High-cardinality tsdb benchmarks: VictoriMetrics vs timescaledb vs influxdb. medium.com, 2018. URL: <https://valyala.medium.com/high-cardinality-tsdb-benchmarks-victoriametrics-vs-timescaledb-vs-influxdb-13e6ee64dd6b>.
 - [10] VictoriaMetrics, VictoriMetrics docs, 2021. URL: <https://docs.victoriametrics.com/Single-server-VictoriaMetrics.html>.
 - [11] S. Shirinbab, L. Lundberg, E. Casalicchio, Performance comparison between scaling of virtual machines and containers using cassandra nosql database, Cloud Computing, 2019, 103 (2019).
 - [12] Cassandra Documentation, Storage engine, 2021. URL: https://cassandra.apache.org/doc/latest/cassandra/architecture/storage_engine.html.
 - [13] Cassandra Documentation, Bulk loading, 2021. URL: https://cassandra.apache.org/doc/latest/cassandra/operating/bulk_loading.html.
 - [14] D. Borthakur, Hdfs architecture guide, Hadoop apache project, 53(1-13), 2 (2008).
 - [15] N. Marz, J. Warren, Big Data: Principles and best practices of scalable realtime data systems (1st ed.), Manning Publications, 2015.
 - [16] Z. Hasani, M. Kon-Popovska, G. Velinov, Lambda architecture for real time big data analytic, ICT Innovations, 133-143 (2014).
 - [17] K. Wang, M. M. H. Khan, Performance prediction for apache spark platform, 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, 2015 IEEE 12th International Conference on Embedded Software and Systems (pp. 166-173). IEEE (2015).
 - [18] K. O'Shea, R. Nash, An introduction to convolutional neural networks, arXiv preprint arXiv:1511.08458 (2015).
 - [19] F. Murtagh, Multilayer perceptrons for classification and regression, Neurocomputing, 2(5-6), 183-197 (1991).
 - [20] M. K. Alsmadi, K. B. Omar, S. A. Noah, I. Almarashdah, Performance comparison of multi-layer perceptron (back propagation, delta rule and perceptron) algorithms in neural networks, 2009 IEEE International Advance Computing Conference (pp. 296-299). IEEE. (2009).
 - [21] A. Brunel, J. Pasquet, J. Pasquet, N. Rodriguez, F. Comby, D. Fouchez, M. Chaumont, A cnn adapted to time series for the classification of supernovae, IST International Symposium on Electronic Imaging, 090, 1-4 (2019). doi:10.2352/ISSN.2470-1173.2019.14.COLOR-090.
 - [22] L. Sadouk, Cnn approaches for time series classification, In: C.-K. Ngan (edt.), Time Series Analysis - Data, Methods, and Applications. IntechOpen 2019 (2019). doi:10.5772/

intechopen.81170.

- [23] S. Suradhaniwar, S. Kar, S. S. Durbha, A. Jagarlapudi, Time series forecasting of univariate agrometeorological data: A comparative performance evaluation via one-step and multi-step ahead forecasting strategies, *Sensors* 2021, 21(7), 2430 (2021). doi:10.3390/s21072430.
- [24] S. Lohmann, S. Negru, F. Haag, T. Ertl, Visualizing ontologies with VOWL, *Semantic Web* 7 (2016) 399–419. URL: <http://dx.doi.org/10.3233/SW-150200>. doi:10.3233/SW-150200.