AutoML for Log File Analysis (ALFA) in a Production Line System of Systems pointed towards Predictive Maintenance

Matthias Maurer, Andreas Festl, Bor Bricelj, Germar Schneider, and Michael Schmeja

Abstract—**Automated machine learning and predictive maintenance have both become prominent terms in recent years. Combining these two fields of research by conducting log analysis using automated machine learning techniques to fuel predictive maintenance algorithms holds multiple advantages, especially when applied in a production line setting. This approach can be used for multiple applications in the industry, e.g., in semiconductor, automotive, metal, and many other industrial applications to improve the maintenance and production costs and quality. In this paper, we investigate the possibility to create a predictive maintenance framework using only easily available log data based on a neural network framework for predictive maintenance tasks. We outline the advantages of the ALFA (AutoML for Log File Analysis) approach, which are high efficiency in combination with a low entry border for novices, among others. In a production line setting, one would also be able to cope with concept drift and even with data of a new quality in a gradual manner. In the presented production line context, we also show the superior performance of multiple neural networks over a comprehensive neural network in practice. The proposed software architecture allows not only for the automated adaption to concept drift and even data of new quality but also gives access to the current performance of the used neural networks.**

Index Terms—**Arrowhead Tools, AutoML, Log Analysis, Neuronal Architecture Search, Predictive Maintenance Framework**

I. INTRODUCTION

Predictive Maintenance (PdM), which roots can be traced back to 1940, gained more and more attention with the rise of automated data acquisition and data processing in decision making [30]. By monitoring system parameters, such as performance, vibrations, temperature development, oil conditions, noise generation, or the like, a useful purpose should be derived. The promises associated with using PdM are diverse, starting with management control, reduction of overtimes, reduction of downtimes, higher quality output, higher user support, etc. [27]. As diverse as the desired benefits of PdM are, the systems' type, under which predictive maintenance is applied, might even be more diverse. This type, besides general principles of PdM, needs to be considered when designing an appropriate PdM approach.

One of the regarded system types, under which PdM is applied, includes an ever-changing production line in a System of Systems (SoSs). An SoS, as a construct of systems, where each was designed and can be used for a main purpose other than being part of this SoS [2, 25], usually brings along non harmonized log messages and uncoordinated behaviour. When combining these different systems with an everchanging environment, e.g., due to a replacement of certain subsystems or because of a changing system load, a highly untransparent and difficult to predict SoS is created. Depending on the concrete setup of the system, different PdM approaches can be applied, among these are log analysis approaches.

Log analysis is a rather easy approach to apply to predictive maintenance [31] since in most cases log data, as a basis for the predictive maintenance intervention, is produced automatically and no further adjustments to the system are needed. They get collected anyway and, hence, only need processing to yield useful predictive maintenance findings. Log analysis approaches can be found for software [1, 12, 13, 15, 24, 28] or hardware SoSs [32], following different PdM objectives with different means. Among these, one can find visual tree representation [15], prediction heuristics, such as the so-called Dispersion Frame Technique [24], or machine learning methods [4, 13].

This work was submitted for review on the 14th of June 2021.

This research work has been funded by the European Commission, through the European H2020 research and innovation programme, ECSEL Joint Undertaking, and National Funding Authorities from 18 involved countries under the research project Arrowhead Tools with Grant Agreement no. 826452. The publication was written at Virtual Vehicle Research GmbH in Graz and partially funded within COMET Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Climate Action, the Austrian Federal Ministry for Digital and Economic Affairs, the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the programme management.

Matthias Maurer is with the 'Contextual Information Systems & Operational Insights' group at the Virtual Vehicle Research GmbH, Graz, Austria (email: matthias.maurer@v2c2.at).

Andreas Festl is Senior Researcher for Data Science at Virtual Vehicle Research GmbH, Graz, Austria.

Bor Bricelj is a Senior Researcher and Data Scientist, working with the "Information Network Extraction Systems" group at the Virtual Vehicle Research GmbH, Graz, Austria. His work is focused on domains of computer vision and data enrichment.

Germar Schneider is Senior Expert for Factory Integration at Infineon Technologies Dresden GmbH & Co. KG and work package leader in the Arrowhead Tools project, Dresden Germany.

Michael Schmeja is the Area Manager for Safety and Security at the Virtual Vehicle Research GmbH, Graz, Austria.

DOI: 10.36244/ICJ.2021.3.8

Machine learning, as one approach to log analysis, is itself a well-researched academic area with application in nearly every imaginable area, especially in autonomous driving, health care, finance, manufacturing, and energy harvesting [3]. It is generally divided into supervised, unsupervised, and reinforcement learning [19]. Supervised learning uses features to predict labels, unsupervised learning uses features to get an insight about their statistical properties, and reinforcement learning uses a reward system for the current model behaviour to optimize the model's behaviour. Depending on the maintenance task in mind, an appropriate approach needs to be identified. In terms of identifying situations that need maintenance actions in advance, one can use the system's logs as features and the maintenance situations as labels, which are expressed by certain log entries. This would suggest supervised learning as a suitable method approach. Supervised learning methods include logistic regression, decision trees, support vector machines, and neural networks, among others [4].

A special fascination holds automated machine learning (AutoML) in this context since it could enable a PdM system to adapt to a changing environment. AutoML, which is mainly used in natural language processing (NLP) and computer vision (CV), aims at automating the entire pipeline of machine learning. Although there have been major achievements in NLP and CV, other areas are neglected [14]. This is also true for log analysis, which would benefit twice from such an approach. Firstly, such an automatization would provide access to this technology for a wider audience and, in general, support the creation of better ML systems. Secondly, besides these general advantages, this would allow the PdM system to update its outdated ML components automatically whenever it is necessary due to the changing environment.

The contributions of this paper are the following: a PdM framework for a steadily changing production line is introduced and different NN architectures are evaluated against each other within this framework using a proof-of-concept implementation. The practical relevance and automated nature of the approach allow for wide applicability, especially for novices in the area of machine learning.

We will now show in this paper the high potential of AutoML in the context of a production line system of systems. Therefore, we first discuss general AutoML techniques in Section 2, before we discuss the applications of AutoML in production systems of systems in Section 3. Section 4 shows the first implementation of the theoretical ideas of Section 3. Finally, Section 5 discusses the conclusions based on this work and possible further work in this context.

II. General AutoML techniques

An extensive review of the state-of-the-art regarding AutoML in the context of neuronal networks (NN) was done by Xin He, Kaiyong Zhao, and Xiaowen Chu [14]. They describe the AutoML pipeline consisting of four stages: data preparation, feature engineering, model generation, and model evaluation. Data preparation, as a means to obtain useful data, is composed of data collection, cleaning, and augmentation. Based on these steps, feature selection, extraction, and construction are used during feature engineering to obtain the features from the data, which are later used for model generation. This model generation can itself be divided into search space, where the ML model's structure is defined (e.g. support vector machine, k-nearest neighbours, neural networks) and optimization methods, which are concerned with optimizing hyperparameter and architecture of the previously defined model. Model evaluation, as the last described stage, is used to evaluate a model's performance. The AutoML pipeline is shown in Fig. 1.

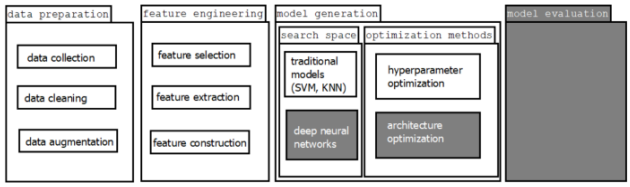

Fig. 1. AutoML pipeline as described by Xin He, Kaiyong Zhao, and Xiaowen Chu [14]. The elements shown in grey describe NAS elements.

Parts of this AutoML pipeline are referred to as neural architecture search (NAS), a sub-topic of AutoML gaining increased attention most recently [14]. NAS includes architecture optimization in the case of a neural network search space from the area of model generation in combination with model evaluation. The idea is to create a basic NN ML model based on the considered search space by applying architecture optimization to this structure and to create the final model by hyperparameter optimization. Evaluation during this procedure is inevitable. The search space describes how candidates for the model's basic structure are found and can be entire-structured, cell-based, hierarchical, and morphism-based. Entire-structured approaches create a structure by selecting layers and their order from a pool of layer candidates. Cell-based approaches use a fixed number of repeating cell structures, consisting of different blocks, which are concatenated afterward and consist itself of different layers combined at the end. One can tune the model by selecting the number of blocks, the operations of the layers in a block, and the combination method at the end of a block (e.g. addition, concatenation, etc.) and cell. An exemplary cell structure is shown in Fig. 2.

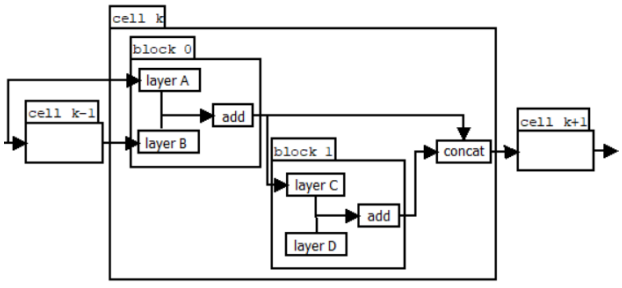

Fig. 2. Exemplary cell structure as shown by Xin He, Kaiyong Zhao, and Xiaowen Chu [14].

In comparison to cell-based approaches, hierarchical search also focuses on the network structure and not only on the cell structure. There are different approaches, all allowing for a fitting on a cell level and a network level [20, 21, 22]. Morphism-based search space uses already existing model structures to improve already existing networks and, hence, create new networks [33].

After defining the NN based on the search space, architecture optimization is used to find the best-performing architecture, which always includes the evaluation of different NNs. This search for the best architecture can be regarded as a search for a hyperparameter, where human expertise is needed. Different algorithms aim at automating this process, such as grid/random search, the evolutionary algorithm, reinforcement learning, gradient descent, surrogate model-based optimization, and hybrid methods. Grid and random search are two very basic optimization methods not considering any feedback from the current state of the architecture and might be considered a baseline approach for comparison. The evolutionary algorithm is a heuristic optimization algorithm, which uses an evaluation procedure until a stopping criterion is met. Starting with a set of NN, the evaluation procedure, inspired by biological evolution, selects a subset based on the NNs' performance, generates a new network from every two previously selected NNs, mutates the resulting NN a bit, and removes the worst-performing new NNs. Another recognized approach, reinforcement learning [37], usually uses a recurrent neural network (RNN) to incrementally improve the architecture by executing certain actions leading to a so-called reward influencing the next action and moving in that manner through the search space. This is shown in Fig. 3.

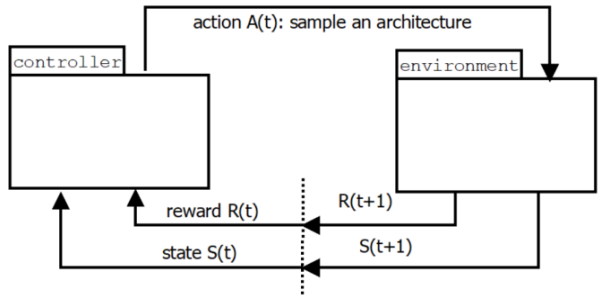

Fig. 3. Basic functionality of reinforcement learning as shown by Xin He, Kaiyong Zhao, and Xiaowen Chu [14].

In comparison to these already mentioned methods to search for the best-performing architecture, gradient descent is an approach allowing for a continuous search space [14, 23]. For that reason, it uses a continuous relaxation of the architecture representation, which is then used for optimization for the operations used in one node of the architecture's cell and leads to one architecture. Surrogate model-based optimization is a broadly used approach for architecture optimize by building a surrogate model to predict the best performing architecture [8, 17, 26]. Of course, all mentioned methods might be combined in a hybrid approach.

After deciding on the architecture to use, which is done with the same set of hyperparameters in most cases, one can turn to the optimization of the hyperparameters for the used architecture [14]. Therefore, different approaches are used, such as grid/random search, Bayesian optimization, and gradient-based optimization. Alternatively, hyperparameter and architecture optimization (HAO) can be used to optimize hyperparameter and architecture in combination [34].

All NAS approaches share one mutual problem, which is the need for frequent evaluation of architectures, leading to a high need for time and computing resources [14]. Different approaches, such as weight sharing [29], surrogate methods [9], early stopping [7], and low fidelity methods [18], aim at reducing this need for resources.

III. ADVANTAGES OF AUTOML IN A PRODUCTION SETTING

AutoML has proven its usefulness in the context of NLP [5, 16] and CV [11, 35], other areas have been neglected [14]. Especially in a production setting, applying AutoML approaches to log data might be useful, since they not only bring along the known advantages, such as easy access to NNs and improvement of ML models but also advantages specific to a production setting. Since there is a wide variation in production settings, we will now describe one rather generic production setting to show the usefulness of AutoML approaches when working with log data in this context.

One aspect of the production line setting is the usage of log data, which is produced by the subsystems and generally easily accessible - merely a central collection of this already existing and accessible data is required for the proposed utilization in a NN. Naturally, one can collect a unique ID for a specific log entry, a unique ID for a specific subsystem where the log entry originated, the time of occurrence, and possibly a duration. Depending on the actual setup further data might be recorded. In this setting, some log IDs might have an informing character, others might indicate a critical or interesting situation in the overall system. It is of utmost importance to prevent the cause of critical log entries from happening or, if this is not possible, to quickly react to the negative influences associated with such a critical log entry. In a predictive maintenance manner, a NN can be trained to predict upcoming log entries of interest-based on the observed log entries to allow for appropriate intervention.

Another aspect of the hereinafter regarded production line setting is its SoS nature, which determines its composition of distinct, changing subsystems working on a changing production load. Both addressed circumstances, a changing system and a changing production load are realistic due to continuous improvements of the production line and variations in the production demands, and they impact heavily on the log data. An altered production load influences the statistic properties of the log data and, hence, might render a trained NN unsuitable. These changing statistical properties, denoted as concept drift [10], are not necessarily happening incremental

but might happen suddenly without any further indication on how the change might unfold, due to an unforeseen change in production. An even greater influence is exhibited by a new subsystem, which might introduce a new quality of log data, which cannot be handled by a trained NN.

The described problems of a sudden concept drift change and the introduction of a new quality of log data, specific to the described production line setting, can be addressed by AutoML approaches. Concept drift, as the first addressed problem, is already a discussed topic in literature [10]. Forgetting mechanisms, for example, allow to incorporate data with different emphasis, depending on how recent they are. In combination with change detection based on sequential analysis, statistical process control, distribution comparison, or contextual approaches, the software can react to concept drift and an adjustment of the model can be initiated. This adjustment, or learning, can be divided into two approaches: retraining, where the current model is discarded, and incremental adaption, where the current model is updated. Hence, concept drift is a phenomenon one can cope with, whereas an introduction of a new quality of log data, as the second addressed problem, is not yet discussed on a wide basis. When working with an embedding layer to map the log IDs to n-dimensional vectors, for example, one would have to adjust the vocable size (number of different log IDs) as input to this layer. This would require the creation and training of a new model.

One alternative approach for introducing new log IDs to a NN, without adapting the NN, is feature hashing [6]. This would lead to classes of IDs, which are used in the model. These classes would be indistinguishable for the model, new IDs would be assigned to one of the already existing classes. Over the course of time, when new ids occur and old IDs vanish, this might lead to a situation where log entries of one ID are used to predict the occurrence of an entirely other log ID. Also, two IDs, which are very different in their behavior, might be bundled together in one class. Such a bundling would, hence, be problematic in certain cases.

By introducing NNs for each log ID of interest, one would be able to create an overall ML model capable to adapt to concept drift concerning individual log IDs and it would be able to introduce a NN based on a new log ID as soon as enough data has been recorded to train such a NN. This allows for an incremental adaption of the overall model, even if new log IDs are introduced. However, each NN would need to accept unknown log IDs as feature values in a residual category. As soon as the respective NN gets retrained, this alarm ID does not receive a separate appearance in the NN. In the background, the active NNs need to be evaluated, replacement candidates are trained using AutoML methods and compared to the active NNs. Whenever a replacement candidate outperforms an active NN, the replacement candidate is incorporated into the overall model. The described workflow can be found in Fig. 4.

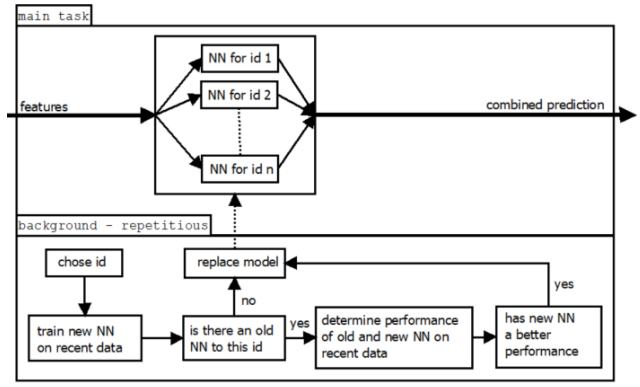

Fig. 4. Overall concept of a prediction model based on multiple NNs, each concerned with predicting one ID.

Another advantage of using log data in the described setup is that labels can be calculated directly based on the features. Input features of the NNs are log IDs, device IDs, times, and so on, output labels are log IDs of interest. This means that the true labels are received sometime after the prediction, which allows for an evaluation of the current NNs and training of replacement candidates.

IV. Analysis of Proof-of-Concept Implementation

Automated decision-making by predictive diagnosis and machine learning is a main topic within the Arrowhead Tools project, a Horizon 2020 project aiming for digitalization and automation solutions for the European industry. In this project different partners from the industry develop new tools to improve the European industry by creating many different new tools e.g., the Arrowhead Framework, but also tools based on new algorithms or neuronal networks which are used in different use cases. One important use case for complex maintenance tasks is the work on equipment data in the semiconductor industry. We worked together with the company Infineon Technologies Dresden on a use case using neuronal networks for a better understanding of the failure in a highly complex wafer transportation system to create predictive maintenance solutions saving time and high personal efforts. Another goal is, that this setup could be used in many different other industrial applications showing high potentials for interoperability. A first implementation of the theoretical ideas discussed in the previous section is implemented. We will first explain the available data, continue with the chosen NN structure, and finish with the used software design.

Available Data Quantities

The log entries in this AHT use case exhibit unique IDs, a start timestamp, an end timestamp, and a spatial location expressed as a segment of spatially close positions. This information is available for all entries, except for the segment, which is available only for around 50% of the data. Therefore, an additional category was introduced, expressing that no location information is available.

Based on these available log data quantities, different derived quantities can be constructed, such as observed log IDs, time since log occurrence, the active state of a log entry, and the segment of occurrence. For a given point in time, these quantities can be fed into a NN – the last log entries, each expressed as ID, time since the occurrence, the information, if it is still active, and the segment of occurrence. This can either be done for a fixed number of past log entries or, more accurately, for a fixed time in the past. To ensure a fixed-size input length to the NN in the latter case, such a fixed number of considered log entries need to be defined. If there are more log entries falling in the designated timeframe, they are ignored and if there are fewer log entries falling in this timeframe, placeholder values need to be introduced. These placeholder values can be zero for the log ID, expressing that no log entry occurred. For the time since log occurrence, it can be one, when the time since occurrence is expressed as a number between zero and one – zero standing for right now and one stands for before or at the beginning of the designated timeframe. When decoding the still active state, it can be zero for not active and for the segment, the no-location-available value is used. A graphical representation of the data quantities is shown in Fig. 5.

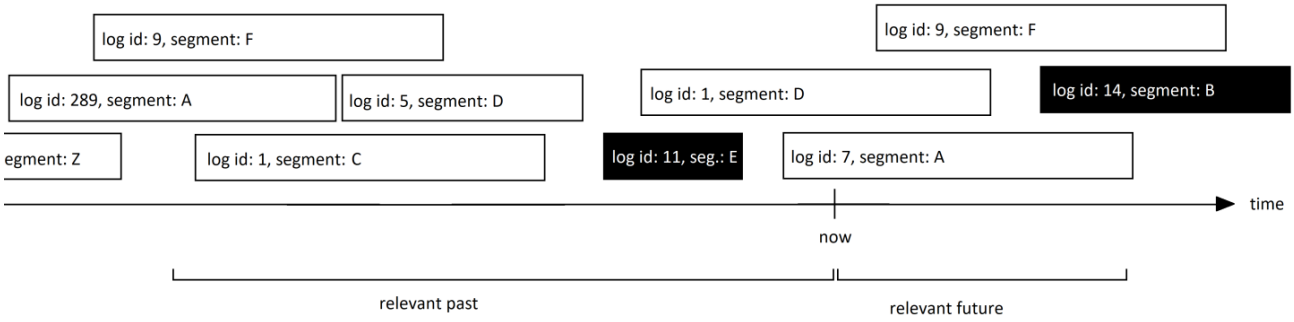

Fig. 5. Visualization of the AHT data. It contains a start and an end timestamp, a log ID, and a segment. Log IDs of interest are shown in a black rectangle. By introducing a relevant past, a relevant future, one can create input vectors for NNs.

Comprehensive NN VS. Multiple NNs

The theoretical advantages of using multiple NNs, one for each log ID of interest, were already discussed at the end of section III and can be extended by practical advantages. Besides the ability to gradually adapt the overall model to the concept drift and new log IDs, the possibility to create a well-performing overall model seems more easily achievable. Although a comprehensive NN predicting all log IDs of interest is theoretically equivalent to a combination of multiple NNs predicting a certain log ID, the computational effort can be reduced by following a divide-and-conquer approach and splitting the comprehensive NN into multiple NNs. This hard to quantify assumption was also observed during our experimentation.

We created a comparison between a comprehensive NN to multiple NNs in our production line setting. Therefore, we aimed to predict log IDs of special interest (based on domain experts' rating) which occur with a relative frequency of at least 0.11%. Furthermore, we used a weighted version of the cross-entropy loss [36] to account for the unbalanced nature of the data, where the weights are inversely proportional to the relative frequency of the corresponding log ID. The used architecture was the same for both cases and is shown in Fig. 6. The features were constructed from the last 100 log entries within the last 45 minutes, the labels were created based on the next 15 minutes. Each embedding layer is of dimension 8, each hidden dense layer consists out of 32 neurons, the drop layer features a dropout rate of 50%.

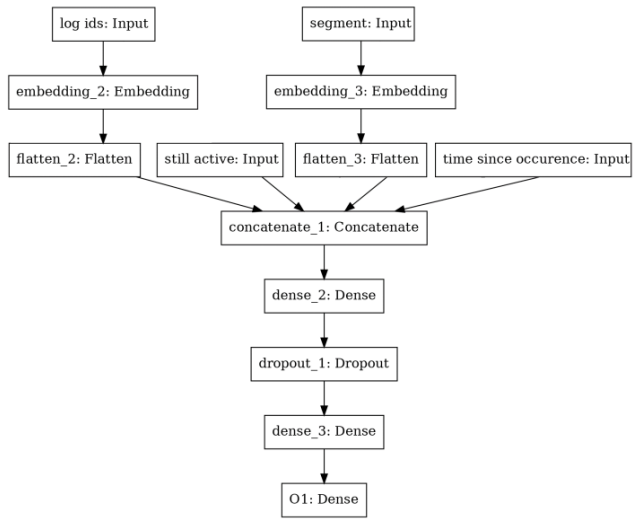

Fig. 6. NN architecture used for comparing a comprehensive NN to multiple NNs.

To compare the performance of the NNs, two widely used performance indicators are introduced – the positive predictive value (PPV) and the sensitivity. Both indicators are empirical probabilities. The indicator PPV can be calculated as the number of correct predictions of a log ID divided by the total number of predictions of this ID. The higher this value is, the more reliable is a gained prediction of this log ID. The indicator sensitivity can be calculated as the number of correct predictions of a log ID divided by the total number of

occurrences of this log ID. The higher this indicator is, the fewer occurrences of this log ID are 'overlooked' by the NN. A reliable prediction system requires both values to be high.

Comparing the performance of a comprehensive NN to multiple NNs in our production line setting speaks for the usage of multiple NNs over a comprehensive NN. Table I shows the two introduced performance indicators, observed when predicted and predicted when observed, for both discussed cases. Except for log ID 4 and 8, both indicators speak consistently for using multiple NNs over a comprehensive NN. Although the log IDs 4 and 8 lead to contradicting indicators to some extent, the overall results speak clearly for using multiple NNs over a comprehensive NN.

TABLE I
PERFORMANCE COMPARISON BETWEEN A COMPREHENSIVE NN (COMNN) AND MULTIPLE NNS (MNNS) IN THE DESCRIBED PRODUCTION LINE SETTING WITH PERCENTAGE VALUES.

LOG ID	PPV		Sensitivity	
	ComNN	MNNs	ComNN	MNNs
1	79	**92**	1	**63**
2	34	**92**	46	**70**
3	65	**71**	32	**78**
4	58	**79**	**68**	66
5	54	**97**	13	**95**
6	47	**80**	50	**93**
7	71	**76**	16	**39**
8	**70**	44	68	**89**
9	65	**70**	8	**32**

The superior practical performance of multiple NNs over a comprehensive NN might be based on different circumstances. One such circumstance is that, instead of working with only one set of hyperparameters, each NN predicting only one log ID of interest allows for its own set of hyperparameters, such as relevant past (see Fig. 5) or the number of neurons in a certain layer. Another contributing factor might be the more complex structure of the comprehensive NN, which might induce a worse performance of the optimization algorithm used for the training procedure, leading to a suboptimal trained NN. Another factor is the design of the used labels, which allow for only one next log ID of interest to be predicted. This might distort the NN's performance, since one log ID of interest might be concealed by another log ID of interest, leading to inferior performance.

To set the performance of the NNs from the multiple NNs approach into relation, one can compare it to trivial prediction models. A suitable base model can be obtained by always predicting the most common observed class. Due to the extremely unbalanced situation, we are facing with the AHT dataset, this is always clearly the prediction, that there will not be an ID of interest in the upcoming timeframe. The accuracy is calculated as the ratio of correct predictions to all predictions. For the no information rate model, this value is always the ratio of no-error predictions, which ranges for our situation between 89% to 98.6%. Although these values are quite high, the obtained NNs outperform this bae-line accuracy in nearly all cases with accuracies between 94.4% and 99.9%, as shown in Table II.

TABLE II
ACCURACY OF THE INDIVIDUAL MODELS FROM THE MULTIPLE NNS APPROACH COMPARED TO THE NO-INFORMATION-RATE MODEL (NEVER PREDICTING AN UPCOMING ID).

LOG ID	Accuracy	
	No information rate	MNNs
1	96.5	**98.6**
2	96.2	**98.6**
3	95.9	**97.8**
4	89.0	**94.2**
5	98.6	**99.9**
6	97.7	**99.3**
7	93.0	**94.9**
8	**95.4**	94.4
9	96.1	**96.9**

Proposed ALFA Software Design

To benefit from the advantages promoted in the previous sections, we propose the ALFA (AutoML for Log File Analysis) software design capable of handling the needed requirements. The software design contains two main components, the predictor and the model updater.

The predictor receives the log information as soon as they occur and creates a prediction based on this information. In the first step, the data – log ID, time, segment, and type (it is either the start or the end of a log event) is received and stored in a database. In a second step, the NNs are loaded from the model updater if they have not been loaded yet and enough data is available to do so. Finally, the NNs are used to predict the occurrence of log IDs of interest if enough data is available. Furthermore, the received log IDs can be used to calculate the performance of the previously made predictions, which guarantees for always present performance indicators for each NN. This component operates only, and as soon as new log data is received.

The model updater is used to store, load, and refit the NNs. It operates on an external trigger, either from the predictor when loading a NN, or a regular impulse, based on e.g., a certain time or a certain amount of received and relevant log data, to refit one or more NNs. When loading the NNs, the model updater first tries to load already existing NNs from the file system. If this is not possible, it loads the relevant log data to create and fit new NNs. In case that there is not enough data to fit NNs, nothing is done – until enough log data is present. Furthermore, on a regular basis, triggered e.g., by elapsed time or a certain amount of received data, the currently used NNs are reevaluated and possibly replaced, in other words, refitted. This includes loading of the relevant log data, the creation and fitting of new NNs, the comparison between these new NNs and the currently active NNs, and replacing the currently active NNs through the newly created and better performing NNs.

The software description is still lacking an essential piece of information, that is, what exactly we mean by creating and fitting a model. Recapitulating the discussions about NAS from Section II, we make use of architecture optimization in case of a neural network search space with subsequent hyperparameter optimization. The precedes tasks of data preparation and feature engineering, which are usually included into the AutoML pipeline are not required in the presented setting, since the data is already well formatted, and features have been predefined. The mode evaluation, on the other hand, is still a crucial aspect of the presented workflow and is used whenever a new NN is created and trained with the data. A schematic depiction of the described ALFA software architecture can be found in Fig. 7.

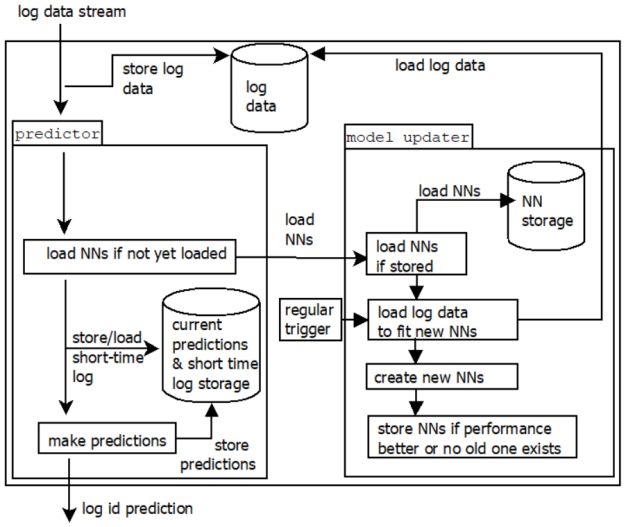

Fig. 7. ALFA software architecture proposed for the AHT use case, consisting of two main components, predictor and model updater.

This proposed software architecture is currently developed in the AHT project for industrial applications, especially for automated decision making by predictive diagnosis and machine learning in a semiconductor use case. The application is constantly enhanced. The search space includes the NN architecture shown in Fig. 6 and slight variations of it, the hyperparameter optimization uses a grid search approach. Possible extensions of this first implementation include the additional prediction of the occurrence segment and the extension of the currently used search space.

V. Conclusion

We have given an overview of the current state regarding log analysis and AutoML, furthermore, the advantages of combining these approaches were presented in theory and for the introduced AHT use case. In this context, the theoretical and practical predominance of using multiple NNs, each tuned to predict one log ID, over one comprehensive NN, predicting all log IDs, was shown. The invoked theoretical advantages are the possibility to gradually adapt the overall prediction model to the concept drift or even to a new quality of data, which are new log IDs in the AHT use case. The invoked practical advantage in the context at hand is the better results produced by using multiple NNs in the AHT use case, shown in Table I.

Based on these considerations, the ALFA software architecture comprised of multiple NNs was proposed, it is shown in Fig. 7. This architecture allows for the beforehand enumerated advantages and, beyond that, to also carry along the up-to-date performance of the used NNs. An automated update of the used NNs is done in the background, as soon as a better performing NN for a given log ID or even a new NN for a new log ID is found, it gets integrated. An expert in the field of NNs is not required to use this setup, opening the usage of it for a wide audience. The sole requirement is a suitable data format.

The gained models, from a first, simple implementation show great potential. Compared to the no-information-rate model, always predicting the most likely class, good performance was achieved. On the very unbalanced AHT dataset, assessed by the accuracy, the obtained NNs outperform the no-information-rate model in nearly all cases on a very high level, as shown in Table II.

The ALFA software architecture is currently developed in the AHT project and is constantly evaluated in this context as a new tool that will be provided to the Eclipse Arrowhead project. Although the first results are promising, a long-time evaluation might hold crucial information for the further development of the proposed prediction model. This especially concerns the evaluate the software component's behavior when confronted with an unknown concept drift and the introduction of data of new quality. This, however, requires time for data in the given setup to be shifted in this direction.

A next step, besides the long-term evaluation, is the extension of the ALFA software architecture. There are two apparent extensions, the introduction of an additional label dimension with the segment of occurrence, additionally to the log ID, and the improvement of the used NAS approach. The used NAS approach can be enhanced by extending the search space to include more NN architectures and by refining the hyperparameter optimization.

References

[1] Bao, L., Li, Q., Lu, P., Lu, J., Ruan, T., & Zhang, K. (2018). Execution anomaly detection in large-scale systems through console log analysis. Journal of Systems and Software, 143, 172-186. DOI: 10.1016/j.jss.2018.05.016

[2] Boardman, J., & Sauser, B. (2006, April). System of Systems-the meaning of of. In 2006 IEEE/SMC International Conference on System of Systems Engineering (pp. 6-pp). IEEE. DOI: 10.1109/SYSOSE.2006.1652284

[3] Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., ... & Zdeborová, L. (2019). Machine learning and the physical sciences. Reviews of Modern Physics, 91(4), 045002. DOI: 10.1103/RevModPhys.91.045002

[4] Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. D. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. Computers & Industrial Engineering, 137, 106024. DOI: 10.1016/j.cie.2019.106024

[5] Chen, J., Chen, K., Chen, X., Qiu, X., & Huang, X. (2018). Exploring shared structures and hierarchies for multiple nlp tasks. arXiv preprint arXiv:1808.07658.

[6] Chen, W., Wilson, J., Tyree, S., Weinberger, K., & Chen, Y. (2015, June). Compressing neural networks with the hashing trick. In International conference on machine learning (pp. 2285-2294). PMLR.

[7] Domhan, T., Springenberg, J. T., & Hutter, F. (2015, June). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In Twenty-fourth international joint conference on artificial intelligence.

[8] Dikov, G., & Bayer, J. (2019, April). Bayesian learning of neural network architectures. In The 22nd International Conference on Artificial Intelligence and Statistics (pp. 730-738). PMLR.

[9] Eggensperger, K., Hutter, F., Hoos, H., & Leyton-Brown, K. (2015, February). Efficient benchmarking of hyperparameter optimizers via model-based surrogates. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 29, No. 1). DOI: 10.1007/s10994-017-5683-z

[10] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM computing surveys (CSUR), 46(4), 1-37. DOI: 10.1145/2523813

[11] Ghiasi, G., Lin, T. Y., & Le, Q. V. (2019). Nas-fpn: Learning scalable feature pyramid architecture for object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7036-7045). DOI: 10.1109/cvpr.2019.00720

[12] He, S., Lin, Q., Lou, J. G., Zhang, H., Lyu, M. R., & Zhang, D. (2018, October). Identifying impactful service system problems via log analysis. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 60-70). DOI: 10.1145/3236024.3236083

[13] He, S., Zhu, J., He, P., & Lyu, M. R. (2016, October). Experience report: System log analysis for anomaly detection. In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE) (pp. 207-218). IEEE. DOI: 10.1109/ISSRE.2016.21

[14] He, X., Zhao, K., & Chu, X. (2021). AutoML: A Survey of the State-of-the-Art. Knowledge-Based Systems, 212, 106622. DOI: 10.1016/j.knosys.2020.106622

[15] Jayathilake, D. (2012, May). Towards structured log analysis. In 2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE) (pp. 259-264). IEEE. DOI: 10.1109/jcsse.2012.6261962

[16] Jiang, Y., Hu, C., Xiao, T., Zhang, C., & Zhu, J. (2019, November). Improved differentiable architecture search for language modeling and named entity recognition. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 3576-3581). DOI: 10.18653/v1/D19-1367

[17] Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., & Xing, E. (2018). Neural architecture search with bayesian optimisation and optimal transport. arXiv preprint arXiv:1802.07191.

[18] Klein, A., Falkner, S., Bartels, S., Hennig, P., & Hutter, F. (2017, April). Fast bayesian optimization of machine learning hyperparameters on large datasets. In Artificial Intelligence and Statistics (pp. 528-536). PMLR.

[19] Lee, J. H., Shin, J., & Realff, M. J. (2018). Machine learning: Overview of the recent progresses and implications for the process systems engineering field. Computers & Chemical Engineering, 114, 111-121. DOI: 10.1016/j.compchemeng.2017.10.008

[20] Liu, C., Chen, L. C., Schroff, F., Adam, H., Hua, W., Yuille, A. L., & Fei-Fei, L. (2019). Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 82-92). DOI: 10.1109/CVPR.2019.00017

[21] Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L. J., ... & Murphy, K. (2018). Progressive neural architecture search. In Proceedings of the European conference on computer vision (ECCV) (pp. 19-34). DOI: 10.1007/978-3-030-01246-5_2

[22] Liu, H., Simonyan, K., Vinyals, O., Fernando, C., & Kavukcuoglu, K. (2017). Hierarchical representations for efficient architecture search. arXiv preprint arXiv:1711.00436.

[23] Liu, H., Simonyan, K., & Y ang, Y . (2018). Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055.

[24] Lin, T. T., & Siewiorek, D. P. (1990). Error log analysis: statistical modeling and heuristic trend analysis. IEEE Transactions on reliability, 39(4), 419-432. DOI: 10.1016/0026-2714(92)90140-g

[25] Maier, M. W. (1998). Architecting principles for systems-of-systems. Systems Engineering: The Journal of the International Council on Systems Engineering, 1(4), 267-284. DOI: 10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D

[26] Mendoza, H., Klein, A., Feurer, M., Springenberg, J. T., & Hutter, F. (2016, December). Towards automatically-tuned neural networks. In Workshop on Automatic Machine Learning (pp. 58-65). PMLR. DOI: 10.1007/978-3-030-05318-5_7

[27] Mobley, R. K. (2002). An introduction to predictive maintenance. Elsevier. DOI: 10.1016/b978-0-7506-7531-4.x5000-3

[28] Oliner, A., Ganapathi, A., & Xu, W. (2012). Advances and challenges in log analysis. Communications of the ACM, 55(2), 55-61. DOI: 10.1145/2076796.2082137

[29] Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In International Conference on Machine Learning (pp. 4095-4104). PMLR.

[30] Selcuk, S. (2017). Predictive maintenance, its implementation and latest trends. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 231(9), 1670-1679. DOI: 10.1177/0954405415601640

[31] Sipos, R., Fradkin, D., Moerchen, F., & Wang, Z. (2014, August). Log-based predictive maintenance. In Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1867-1876). DOI: 10.1145/2623330.2623340

[32] Tan, W. N. (2019). SMT Machine Log File PDE Features Extraction and Analysis (Doctoral dissertation, Tunku Abdul Rahman University College).

[33] Wei, T., Wang, C., Rui, Y., & Chen, C. W. (2016, June). Network morphism. In International Conference on Machine Learning (pp. 564- 572). PMLR.

[34] Zela, A., Klein, A., Falkner, S., & Hutter, F. (2018). Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. arXiv preprint arXiv:1807.06906.

[35] Zhang, H., Li, Y., Chen, H., & Shen, C. (2019). Ir-nas: Neural architecture search for image restoration. arXiv preprint arXiv:1909.08228.

[36] Zhang, Z., & Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. arXiv preprint arXiv:1805.07836.

[37] Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.

Matthias Maurer is Senior Researcher for Contextual Information Systems and Operational Insights at Virtual Vehicle Research GmbH. His research interests include data analytics, machine learning, and cognitive science. Matthias has received his Dipl.-Ing. (equiv. MSc) from Graz University of Technology in Technical Mathematics and his Mag. rer. nat. from University of Graz in Education. His work is manly focused on the analysis of time-based production measurement and log data, focusing on predictive maintenance and root cause analysis..

Dipl.-Ing. **Andreas Festl**, BSc, is Senior Researcher for Contextual Information Systems and Management at Virtual Vehicle Research GmbH and affiliated lecturer for data and information science at FH Joanneum University of applied sciences. His research interests include data analytics, statistics and machine learning. Andreas has received his Dipl.-Ing. (equiv. MSc) from Graz University of Technology in Technical Mathematics. A large part of his work was and is focused on the analysis of time-based automotive measurement data, thereby answering questions about customer vehicle usage, driving behavior and various environmental conditions.

Mag. Bor Bricelj, CQRM holds a master's degree in economics and finance from University of Maribor's Faculty of Economics and Business, and a Certificate in Quantitative Risk Management from the International Institute of Professional Education and Research. He worked in the fields of financial services and higher education before transitioning to data science. He has more than five years of experience as a data scientist, implementing statistical and machine learning methods to analyse and solve various industry specific problems in different industry branches, ranging from automotive industry, heavy industry, to chemical industry. At Virtual Vehicle Research GmbH, he is employed as a Senior researcher / Data Scientist, working with the "Information Network Extraction Systems" group. His work and research are focused on domains of computer vision and data enrichment.

Dr. Germar Schneider (m) holds a Diploma and a PhD in chemistry. He joined the Siemens AG in Essonnes in France in 1995 as a process engineer in the wet department. In 1998 in Dresden, he became the section manager for the 200mm wet department. From 2004 to 2008, he built up a team that was important for factory automation. Between 2008 and 2012, as manager in the new wafer test department he was responsible for production & maintenance and equipment engineering. With 26 years of experience combining know-how of process engineering, production, maintenance, automation and the experience in digitalization projects he is main driver in various JU projects.

Dr. Michael Schmeja (55) has been working at the Virtual Vehicle Research Center in Graz since 2009 and is currently the responsible Area Manager for Safety & Security. After studying mathematics, Mr. Schmeja earned his doctorate at the Institute of Railway Engineering at the Graz University of Technology. From 1997 - 2009 he held various management positions at Siemens Mobility, where he was awarded Inventor of the Year in 2003. In 2009, he moved to Virtual Vehicle and, in addition to his function as Area Manager, he also manages numerous international research projects.