








HiMLEdge – Energy-Aware Optimization for Hierarchical Machine Learning

Julio Wissing¹ , Stephan Scheele¹ , Aliya Mohammed¹ ,
Dorothea Kolossa² , and Ute Schmid^{1,3} 

¹ Fraunhofer IIS, Fraunhofer Institute for Integrated Circuits IIS, Erlangen, Germany
{julio.wissing,stephan.scheele}@iis.fraunhofer.de

² TU-Berlin, Berlin, Germany

dorothea.kolossa@tu-berlin.de

³ University of Bamberg, Bamberg, Germany

ute.schmid@uni-bamberg.de

Abstract. Smart sensor systems are a key factor to ensure sustainable compute by enabling machine learning algorithms to be executed at the data source. This is particularly helpful when working with moving parts or in remote areas, where no tethered deployment is possible. However, including computations directly at the measurement device places an increased load on the power budget. Therefore, we introduce the Hierarchical Machine Learning framework “HiMLEdge” which enables highly specialized models that are tuned using an energy-aware multi-criteria optimization. We evaluate our framework with prognostic health management in a three-part feasibility study: First, we apply an exhaustive search to find hierarchical taxonomies, which we benchmark against hand-tuned flat classifiers. This test shows a decrease in power consumption of up to 47.63% for the hierarchical approach. Second, the search strategy is improved with Reinforcement Learning. As a novel contribution, we include real measurements in the reward function, instead of using a surrogate metric. This inclusion leads to a different optimal policy in comparison to the literature, which shows the error that may be introduced by an approximation. Third, we conduct tests on the system level, including communication and system-off power draw. In this scenario, the optimized hierarchical model can perform four times as many readings per hour as a flat classifier while achieving the same five years of battery life with similar accuracy. In turn, this also means that the battery life can be increased by the same amount if the readings per hour are kept constant.

Keywords: Edge AI · Energy efficiency · Hierarchical machine learning

1 Introduction

The advancement of artificial intelligence (AI) might be one of the biggest impact factors when it comes to achieving the United Nations’ Sustainable Development

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

T. Guarda et al. (Eds.): ARTIS 2022, CCIS 1676, pp. 15–29, 2022.

https://doi.org/10.1007/978-3-031-20316-9_2

Goals (SDGs) [17]. Monitoring and managing natural disasters or explainable AI for health care [5], there are many possible applications for machine learning (ML) or AI to accelerate progress on the SDGs.

Previously, the dominant data processing method followed a centralized architecture, in which sensor data is transferred to cloud resources and processed remotely. However, this approach has the downside of a high energy consumption overhead due to the excessive communication of unfiltered sensor data. Particularly, considering that many applications include an event or anomaly detection, there is a waste of energy due to avoidable data transmissions. In recent years, the widespread adoption of machine learning in edge computing under the term “tinyML” enabled intelligent applications on resource-constrained IoT devices. Thereby, the on-device execution of machine learning models is becoming a considerable alternative to the centralized approach of data processing [12]. Nonetheless, a key requirement of such wireless systems is energy efficiency, a mandatory factor to ensure long-lasting battery life of months or even years for tiny embedded sensor systems. Achieving this goal is a challenge demanding highly efficient ML-Models, which are optimized with respect to energy consumption and accuracy. In current studies, hierarchical machine learning (HiML) has been explored as method to save energy. In its simplest form, it is already present in everyday devices like, e.g., keyword spotting based natural language processing pipeline. However, scaling this idea to multiple hierarchy levels and applying it to arbitrary problems is challenging.

Hence, we introduce our framework HiMLEdge that makes use of *automatically optimized* HiML models, which represent the decision task as a classification hierarchy. This allows for lazily triggered computations that only consume the energy needed. We see this technology as an enabling factor for many applications that are currently not practically solvable. Especially in remote areas with no connection to direct power (e.g., undeveloped areas, rain forests, or the sea), on device computing can be very beneficial [13]. Increasing the battery life or duty cycle of such system is therefore the ultimate goal and enables more TinyML devices to achieve progress on the SDGs.

1.1 Synopsis

This paper discusses the impact of HiML for energy efficient inference with the help of the newly introduced framework HiMLEdge. To do so, Sect. 2 introduces the works related to the topic. Next, Sect. 3 describes the structure, background, and general functionalities of HiMLEdge, which we test in a feasibility study in Sect. 4. We structure the study in four steps. First, we describe the measurement setup. Second, we conduct an initial experiment on optimization with an exhaustive search strategy, which we improve with reinforcement learning as a third step. Fourth, we take a system-wide view of energy consumption with HiML, including communication and leakage power. Finally, we conclude our findings and discuss possible future directions in Sect. 5.

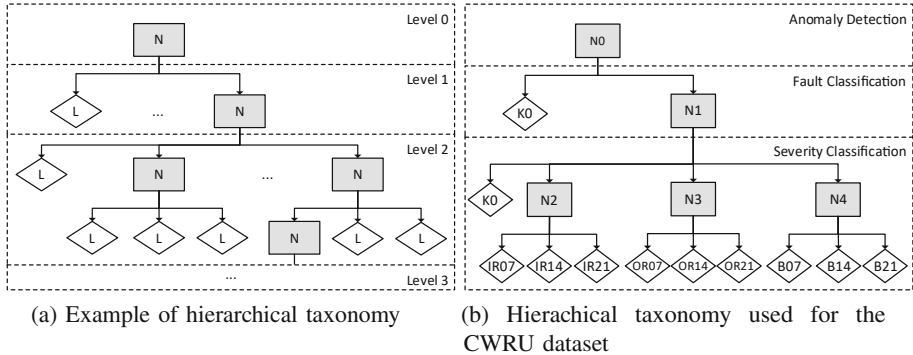


Fig. 1. Description of hierarchical taxonomies in a general example and applied to the CWRU dataset. For the applied case the labels no fault (K0), Inner Race fault (IR07-IR21), Outer Race fault (OR07-OR21) and Ball fault (B07-B21) are shown as leaves.

2 Related Works

Hierarchical classification has already been applied in several application scenarios [11, 18], often with the motivation to improve the quality of the classification results rather than to reduce energy consumption. However, previous work has shown that hierarchical classification can also be used as a partitioning method to increase the energy efficiency of a ML-model by using cascaded processing [7], modularizing its classification taxonomy [2] or by distributing its subcomponents and workload across multiple edge devices [15].

While the algorithm selection problem (ASP) for a non-hierarchical (flat) classifier is already a research field on its own [4], it becomes a more complex task when switching to a multi-model approach. The authors in [1] solve the ASP for hierarchical taxonomies with reinforcement learning by interpreting the selection of each classifier in the hierarchy as actions. The agent then collects a reward based on accuracy *and* theoretical computational complexity. Building up on this idea, we include energy measurements during the search to verify Adams et al. surrogate energy metric.

In contrast to previous works we also include the feature extraction in the search space to only compute necessary features at every step in the hierarchy. HiMLEdge allows us to streamline the process of training, optimizing and deploying HiML models towards embedded devices.

3 The HiMLEdge Framework

The HiMLEdge framework makes use of hierarchical machine learning to find an energy efficient pipeline. Based on a class hierarchy, the classification task is partitioned into a class taxonomy of decisions. The class taxonomy can be represented as a directed acyclic graph (DAG) or similarly a poset $(N, <)$, where

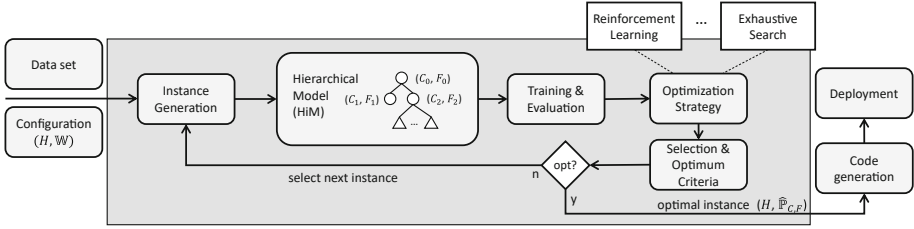


Fig. 2. Visualization of the HiMLEdge-Framework.

N is a finite set of nodes and a partial order \prec over N that is asymmetric, anti-reflexive and transitive. This structure can be simplified to a tree known from a classical hierarchy scheme, e.g., as shown in Fig. 1a, where levels or even single nodes can be represented by a unique classifier instance consisting of a feature extraction and a model solving stage. We will denote such hierarchical classifiers by *Hierarchical Model* (HiM) in the following.

Contrarily, a flat classification approach is represented by a single complex multilabel classifier, responsible for classifying all labels. The authors in [14] introduce an abstract description of such hierarchical structures, describing multiple approaches of which we use the *local classifier per parent node* variant to build a HiM for energy-efficient execution. In this approach, each parent node inside the HiM is modularised and can be represented by a different classifier instance to achieve the highest degree of flexibility. The modularity of hierarchical models has the advantages that (i) similarly to a divide-and-conquer algorithm, a complex multilabel classification problem is partitioned into simpler sub-problems that can be solved more efficiently; (ii) for each node we can select the best suitable and problem-specific classification algorithm; (iii) the node-wise modularization makes it possible to optimize individual parts of a hierarchical model, to adapt them more efficiently to changes and to scale the model to a distributed system architecture. This enables the combination of a broad variety of target platforms starting from specialized custom circuits such as neuromorphic hardware, embedded microcontrollers, or multi device networks. The HiMLEdge framework¹ as shown in Fig. 2 is able to handle the complete process of generating, training, selecting and transpiling to C-code of a HiM for an embedded platform. In the optimization process, the framework generates a HiM in a JSON-like representation out of a possible permutation of given feature and classifier sets for a specific class taxonomy. Any HiM formulated in this representation can then be read in, trained and evaluated by the framework, building the basis for the optimization process. Training and testing of a HiM is made possible with a DAG based recursive algorithm, following the taxonomy from top to bottom. Each node is modeled to have access to its own classifier as well as feature extraction and holds references to its following node(s). In that way, each node can call the train/predict function of the next node(s)

¹ <https://github.com/Fraunhofer-IIS/HiMLEdge>.

or terminate the recursion if a leaf node has been reached. For the ASP the user can choose an optimization strategy, which is able to include direct feedback of the energy consumption with real measurements of the HiM model in question, or use an approximation for the energy consumption to find an HiM without measurement hardware. Currently, the framework is able to utilize an exhaustive search approach or a reinforcement learning based optimization like shown in [1], but will be extended with further techniques like e.g. Evolutionary Algorithms [3] in the future.

3.1 Optimization Problem

We use a local classifier per parent node approach, because each classifier and feature extraction in every node of the HiM can be chosen independently. Let $\mathcal{C} = \{c_1, \dots, c_j\}$ and $\mathcal{F} = \{F_1, \dots, F_k\}$ be finite sets of classifier and feature labels, respectively. The ASP can be described by the optimization problem to maximize the reward function

$$R(\hat{\mathbb{P}}_{C,F}) = \lambda \mathcal{A}(\hat{\mathbb{P}}_{C,F}) + \frac{1}{\lambda} \hat{E}(\hat{\mathbb{P}}_{C,F}), \quad (1)$$

where $\hat{\mathbb{P}}_{C,F}$ is a finite set indexed by N that contains for each node $n \in N$ a possibly optimal classifier-feature pair (\hat{C}_n, \hat{F}_n) , which combines a single classifier $\hat{C}_n \in \mathcal{C}$ with a set of feature labels $\hat{F}_n \in \mathcal{P}(\mathcal{F})$. The reward consists of a weighted sum of the accuracy \mathcal{A} and the approximated energy consumption score \hat{E} , which may be replaced by a real measurement. The weighting factor λ is a hyperparameter that can be used to shift the optimization towards energy consumption or classification performance. In our tests, $\lambda = 2$ showed a good trade-off between both criteria. A smaller value led to trivial classifiers and an increased λ to inefficient models. This is because for more extreme values of λ the optimization solely focuses on either accuracy or energy consumption. The optimization procedure picks from a configuration set

$$\mathbb{W} = \{(\mathfrak{c}_0, \mathfrak{f}_0), (\mathfrak{c}_1, \mathfrak{f}_1), \dots, (\mathfrak{c}_N, \mathfrak{f}_N)\}, \quad (2)$$

where each $\mathfrak{c}_n \in \mathcal{P}(\mathcal{C}) \setminus \emptyset$ and $\mathfrak{f}_n \subseteq \mathcal{P}(\mathcal{F})$.

3.2 Reinforcement Learning

As an alternative approach to the exhaustive search, we also test Reinforcement Learning (RL) to solve the ASP for a HiM. RL is a machine learning method that aims to learn the optimal sequence of actions called *policy* required to reach a specific goal. It consists of an *environment*, and an *agent* that exists in certain *states*. The agent is the main actor, that interacts with the environment by performing actions. The agent gets the feedback from the environment as a *reward* and a new *state*. The agent's goal is to maximize its reward over time. In [1], the authors propose a RL method for the ASP of hierarchical classification

Algorithm 1: Reinforcement Learning Algorithm for Hierarchical Classification

Input : $\varepsilon, \alpha, \gamma, \lambda, X, Y$
Output: The optimal policy π^*

```

1 Initialize Q-table, for episode = 0...Episodes do
2   for i = 0...I do
3     Select  $X_k$  and  $Y_k$  for hierarchy level  $k$ 
4     Select testing instance  $x_i, y_i$ 
5     Build training set:  $X_k \setminus x_i, Y_i \setminus y_i$ 
6     Select an action type  $a \in A$ 
7     Train classifier of type  $a$  using training set
8     Predict label for  $x_i$  using trained classifier
9     Move to next state  $s'$  based on the prediction
10    Update  $Q(s,a)$ 
11    if  $s'$  is a leaf node then
12      | Break;
13    else
14      | Return to line 3;
15    end
16    for  $s \in S$  do
17      |  $\pi^*(s) = \underset{a}{\operatorname{argmax}} Q(s,a)$ 
18    end
19  end
20 end

```

problems. We extend this method by testing it on an embedded device and performing real-life measurements. In this context, the set of states are considered to be the different levels in the hierarchy. The set of actions are the different classification algorithms. In every episode, the agent selects a classifier type and trains it, then conducts inference on a random test instance of the dataset. Based on the generated prediction, the agent moves to the new state in the next level and collects a reward. An episode is complete when the agent reaches a terminal state. The reward function is similarly defined as in Eq. 1, with $\hat{E}(\hat{\mathbb{P}}_{C,F})$ being replaced by the measurement

$$R(\hat{\mathbb{P}}_{C,F}) = \lambda \mathcal{A}(\hat{\mathbb{P}}_{C,F}) - (1 - \lambda) \hat{E}(\hat{\mathbb{P}}_{C,F}).$$

Here we use a different weighting method, where λ is between 0 and 1 to keep comparability with [1]. The complete process of training the RL-Agent to select a classifier for each node can be seen in Algorithm 1. As shown in the literature, we apply the Monte-Carlo on-policy strategy, where the Q-table is updated after every episode. X and Y are the sets of data samples and their corresponding labels, respectively. The parameter ϵ balances exploitation and exploration in RL algorithms. The learning rate is denoted with α , and γ is the reduction factor. We implement an interface to handle communication from the host to the embedded device. The interface is able to obtain direct feedback from a

source measurement unit (SMU) about energy consumption and current inference results from the micro controller. First, the host conducts the action selection and the classifier training. It then sends the trained classifier’s parameters and a test instance to the embedded device that uses the sent parameters to perform inference. Simultaneously, the embedded device triggers a measurement of the energy consumption while inference is performed. Lastly, the Host receives the prediction from the embedded device and the energy consumption from the SMU. This information is used to update its Q-table and continues to perform a new episode with a new test instance. After several episodes the agent learns an optimal policy, which is a sequence of classifier types for every level in the hierarchy.

4 Feasibility Study

To show the real-world benefits of applying automated HiML, we conduct a feasibility study using the HiMLEdge framework with three experiments. We start by applying an exhaustive search algorithm, followed by reproducing the reinforcement learning based algorithm selection introduced by [1]. As a third step we conduct tests on a system level view of energy consumption by simulating an application scenario including communication, data retrieval and classification.

4.1 Dataset

For evaluation we chose condition monitoring as a target application, which is part of the SDG Goal 9 working towards sustainable industrialization. Therefore, we are using the well-studied CWRU-Bearing dataset [9], which was recorded using a 2HP motor connected to a dynamometer via a torque transducer/encoder. This also gives us the opportunity to see the applicability of a different dataset to hierarchical machine learning in comparison to [1]. The bearings used in this test are supporting the motor shaft and have been artificially damaged at different locations with fault depths ranging from 0.007” to 0.021”. The vibration data included in the dataset was collected using single-axis accelerometers with a sampling rate of 12 kS/s (fan-end). The data is separated into windows of 512 samples, with a split of 60% training, 20% validation, and 20% test data.

4.2 Measurement Setup

All classifiers used by the HiMLEdge framework are trained with the Scikit-Learn module in python. Before any data is evaluated, the input features are scaled to have zero mean and unit variance. For the features in the frequency space, a Fast fourier transform is used with an FFT size equal to the number of samples in a window. To describe the classifier and feature selection steps we will use the following abbreviations: DT = Decision Tree, LR = Logistic Regression, SVM = Support Vector Machine, RF = Random Forest, MLP =

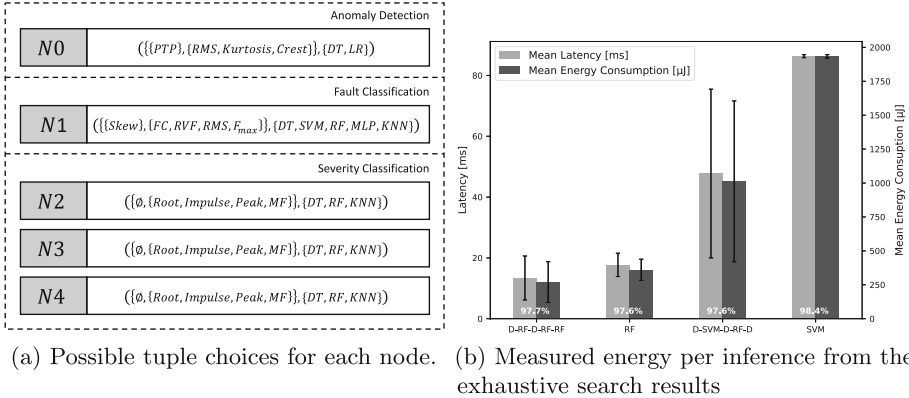


Fig. 3. Possible tuple choices for each note during exhaustive search (left) and search results with energy consumption (right).

Mult-Layer-Perceptron, KNN = k-Nearest Neighbours, PTP = Peak-To-Peak, RMS = Root Mean Square, FC = Frequency Centroid, RVF=Root Variance Frequency, F_{max} = Maximum Frequency, and MF = Mean Frequency. The calculation of the statistical features in both time and frequency domain can be found in [8]. The hyperparameters of each classifier are tuned by a grid search based approach on a validation set. After training, the selected classifiers are ported to plain C using the micromlgen² module and compiled using gcc with optimization (-O3) enabled. The ported classifiers are tested on an Arduino Nano 33 BLE Sense by using a subset of windows from the test-set. The energy consumption is measured with a source measurement unit (PXIe-4145). To reduce the influence of background systems and sensors, we first measure the base power consumption of the microcontroller during idle. This base power is subtracted from the power consumption measured during inference, resulting in the energy consumption in Joule per inference.

4.3 Exhaustive Search

In the first experiment, we use an exhaustive search algorithm to obtain a complete overview of the search space. In this way, we are able to construct a knowledge base, which can later be used to form heuristics for future search algorithms. We separate the classification task in three levels: Anomaly detection ($i = 0$), fault classification ($i = 1$) and severity detection ($i = 1, 2, 3$) (cf. Figure 1b). Due to the large search space we chose to approximate the energy consumption with the mean latency per inference τ measured on the host PC, which is normalized with respect to the highest measured latency during the search.

$$\hat{E}(\hat{\mathbb{P}}_{C,F}) = \left(1 - \frac{\tau(\hat{\mathbb{P}}_{C,F})}{\tau_{max}}\right). \quad (3)$$

² <https://github.com/eloquentarduino/micromlgen>.

The two best performing HiMs are transpiled (source-to-source compiled) to plain C and deployed to the embedded device. As baseline models, two flat classifiers (SVM and RF) are compared to the optimized HiMs in terms of energy consumption and accuracy. For the hierarchy in Fig. 1b, we let the optimization pick out of the sets shown in Fig. 3a, which use the following configuration $\mathbb{W} = \{(\mathfrak{c}_0, \mathfrak{f}_0), (\mathfrak{c}_1, \mathfrak{f}_1), \dots, (\mathfrak{c}_4, \mathfrak{f}_4)\}$ with

$$\begin{aligned} \mathfrak{c}_0 &= \{\text{DT, LR}\}, \\ \mathfrak{f}_0 &= \{\{\text{PTP}\}, \{\text{RMS, Kurtosis, Crest}\}\}, \\ \mathfrak{c}_1 &= \{\text{DT, SVM, RF, MLP, KNN}\}, \\ \mathfrak{f}_1 &= \{\{\text{Skew}\}, \{\text{FC, RVF, RMS, } F_{max}\}\}, \\ \mathfrak{c}_2 &= \mathfrak{c}_3 = \mathfrak{c}_4 = \{\text{DT, RF, KNN}\}, \\ \mathfrak{f}_2 &= \mathfrak{f}_3 = \mathfrak{f}_4 = \{\emptyset, \{\text{Root, Impulse, Peak, MF}\}\}, \end{aligned}$$

leading to a total of 12964 permutations.

The results shows improvements in energy consumption for both HiMs over the baseline (cf. Figure 3b). A significantly influencing factor for the hierarchical model is the fault-detection, which is presumably responsible for most of the computations in the model. This is because of the inclusion of an FFT in the feature extraction process and the more complicated classification task. Therefore, we will continue comparing the hierarchical model with a RF at its core with the RF baseline and the SVM core with the SVM baseline.

Comparing the RF-based models, a decrease in both mean energy consumption and latency of 24.96% can be achieved using the hierarchical approach while obtaining a slight increase in accuracy of 0.1%. This improvement is most likely connected to the distribution of the dataset, where 25% of the cases are non-faulty. Here, the use of a DT with only time-domain features is enough for the decision to stop the computation. The additional overhead introduced by the hierarchical structure might be compensated by the severity detection, where in the case of the found model for some fault classes no additional features are needed.

When comparing the two SVM-based approaches, the gap between the flat classifier and the hierarchical model increases further with a decrease of 47.63% in energy consumption, but with a slightly worse accuracy of 97.6% (SVM-baseline 98.4%). This increase in energy efficiency for the hierarchical model can be explained with the complexity of a SVM, which in the worst case can be $O(n^3)$ [10]. With the higher number of input features as well as possible classes, the cubic complexity becomes a problem in this scenario. The decrease in accuracy can be explained by error propagation through the model. If a classification error is made in any node, it is forwarded through the complete hierarchy and therefore influences the final decision.

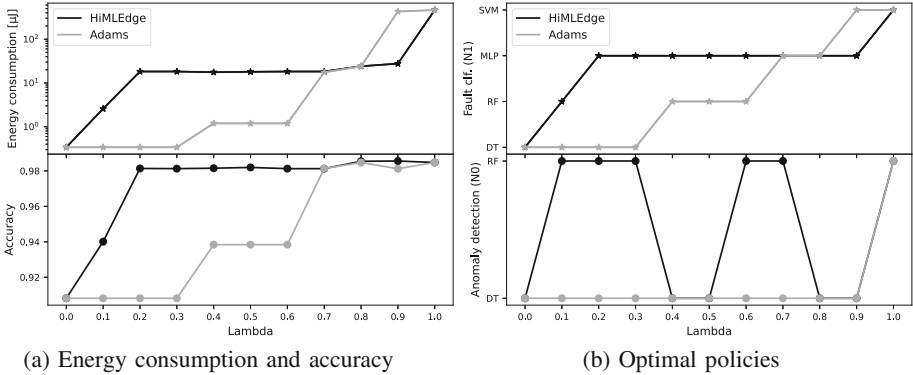


Fig. 4. Influence of λ on energy consumption, Accuracy and model selection.

4.4 Reinforcement Learning

To test the capabilities of the RL-based selection approach, we narrow down the search space to stay comparable to [1] by only optimizing the classifiers. For all nodes ($n \in \{0, 1, \dots, 4\}$) the agent can pick out of the set $\mathcal{C}_n = \{\text{DT}, \text{RF}, \text{MLP}, \text{SVM}\}$. Logistic regression and KNN classifiers were left out in this experiment, as they did not show adequate results during the exhaustive search. Additionally, the RFs have been limited to a maximum of ten estimators due to higher memory constraints introduced by the runtime in the background. The same feature sets found in Sect. 4.3 are used.

With feedback from measurements the experiment resulted in the optimal policy $\{\text{DT}, \text{MLP}, \text{RF}, \text{MLP}, \text{RF}\}$, which differs compared to the policy found when using Adams et al.’s approximation ($\{\text{DT}, \text{RF}, \text{DT}, \text{DT}, \text{DT}\}$). During testing we explored that even though an MLP had a much higher complexity measure in comparison to a RF, the measurement resulted only in a slight increase in energy consumption. Because of the higher accuracy of the MLP, the agent probably decided to lean towards picking the MLP over the RF. Additionally, the training process converged after only a few hours of training on a single desktop PC, while the exhaustive search was run in the course of multiple days. This shows the importance of a suitable algorithm selection techniques that does not only save energy for the inference, but also needs less power during optimization and training. Without a proper approach, the ASP becomes unsolvable for hierarchical classification with an exhaustive search in increasing search spaces and the positive impact on sustainability becomes questionable.

Influence of Trade-off Parameter λ . The model selection process is highly influenced by the trade-off parameter λ , which shifts the selection from maximum energy awareness ($\lambda = 0$) to maximum accuracy ($\lambda = 1$). While both Adams et al.’s surrogate metric and our framework behave as expected with an increase in energy consumption and accuracy for rising values of λ (cf. Figure 4a), the two approaches differ at the point this increase starts. Our approach reaches near

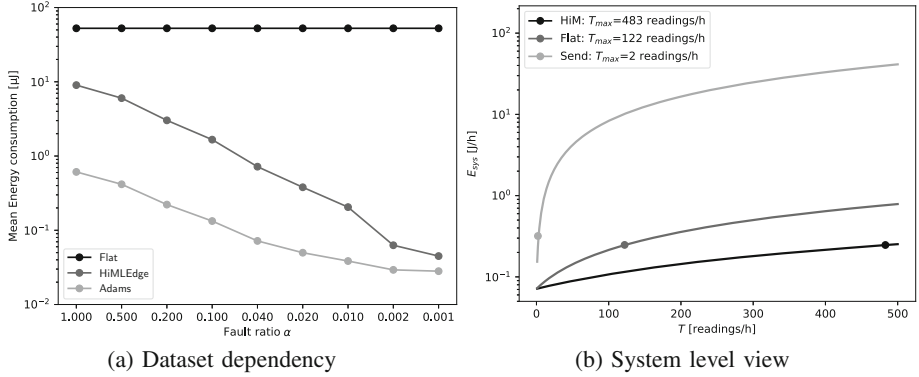


Fig. 5. Dataset dependency for different ratios between faulty and normal cases (left) and system level view (right). The maximum possible duty cycle for 5 years of run time are marked

peak accuracy much earlier ($\lambda = 0.2$) than Adams et al.’s approach, but the energy draw only reaches a modest value of 18.28 $\mu\text{J}/\text{Inference}$. This behavior is also reflected in the classifier choice of both approaches shown in Fig. 4b. The optimal policies for fault detection (N1) seem to be highly correlated with the energy consumption. This verifies the assumption made in Sect. 4.3 that the fault classification is most influential for the energy consumption of the complete HiM. In general, our approach seems to favor an MLP outside of extreme values for λ , while the agent collecting the reward from a surrogate metric sticks to tree-based algorithms (RF, DT) most of the time. This difference can be explained with a not optimal approximation of the energy for an MLP. While in theory, the MLP should be much more computational intensive in comparison to e.g. RF, the regular matrix computations used in a Neural Network are easier to optimize for a compiler. This leads to only a modest increase in energy consumption, but with the benefit of a much more precise model. Additionally, the Cortex M4 can utilize DSP functionalities for the optimal execution of matrix operations. Therefore, the search process should not only include theoretical compute complexity, but also measurements or hardware-aware surrogate models.

Dataset Dependency. Due to the nature of hierarchical classification, the energy consumption is inherently dataset dependent. This is especially the case if the hierarchy includes a lightweight anomaly detection like in our case. Here, the ratio between faulty and non faulty events

$$\alpha = \frac{\#FaultyEvents}{\#NormalEvents} = \frac{\alpha_f}{\alpha_n},$$

has the highest impact on the energy needed. As discussed before, with λ set to 0.5, our approach needs more energy compared to Adams et al.’s approach. However, it benefits from a higher classification performance. While this view

is true for the standard dataset distribution ($\alpha \approx 4$), in the field α is usually much smaller. In this case, both approaches converge to the same mean energy consumption

$$\overline{E} = \frac{\alpha_n E_{N0} + \alpha_f \overline{E}_f}{\alpha_n + \alpha_f} = \frac{\alpha_n E_{N0} + \alpha \alpha_n \overline{E}_f}{\alpha_n + \alpha \alpha_n} = \frac{E_{N0} + \alpha \overline{E}_f}{1 + \alpha} \underset{\alpha \rightarrow 0}{=} E_{N0}$$

if the energy draw of the anomaly detection E_{N0} is equal for both approaches (same policy for N0 cf. Figure 4b). For α nearing zero, the mean energy consumption of the remaining nodes (N1-N4) \overline{E}_f becomes negligible, which can also be observed in Fig. 5a.

In general, this shows that the applicability of HiML for energy efficiency is highly dataset-dependent. If a problem can be solved by a classifier that can be partitioned into multiple stages with increasing degrees of complexity, and if the low-complexity nodes are executed more often, the hierarchical approach works well. However, in scenarios where complex computations are needed at every point in time, a HiM might even consume more energy due to the added overhead.

In conclusion, the tests conducted with RL show not only the dataset dependency and the influence of λ on the selection process. They also highlight the importance of an applicable search strategy. The exhaustive search discussed in Sect. 4.3 took multiple days on a cluster of CPUs to find a solution, while the RL agent was trained in hours on a single machine including measurement feedback. This training time decreases further with Adams et al.’s surrogate metric. In general, a combination of heuristics, search strategy, and hardware-aware surrogate models should be used to build a precise AutoML tool for HiML.

4.5 System Level Energy Consumption

As a last test we want to evaluate a system view to get insight in the power consumption in an application scenario. Before, we always focused on optimizing the energy consumption of the machine learning pipeline isolated from the total power draw of other components and communication. Therefore, in the following we will calculate a system level energy score per hour

$$E_{sys} = T[\alpha \overline{E}_f + (1 - \alpha)E_{N0}] + E_{off}, \quad (4)$$

where T denotes the readings per hour and E_{off} the system-off power consumption per hour. To simulate a real-life application scenario, we changed the dataset distribution to $\alpha_f = 1$ and $\alpha_n = 20$. Thus, a fault only occurs in 5% of the measurements. With this setting, we compare three cases: A flat classifier (RF), the best performing HiM from Sect. 4.3 and sending data only. In the case of classification, the result is sent via BLE if a fault has been detected, while in the sending data case the complete window holding 512 values is always sent to the host. The power-off energy is taken from the datasheet of the NRF58240 (71.28 mJ/h). The goal of this test is to find the maximum possible duty cycle T to achieve 5 years of battery life with a single CR2477 coin cell battery holding

1000 mAh (10800 J @ 3 V) of charge in the respective scenarios and thereby see the impact on energy consumption of hierarchical classification in an application scenario.

It is clearly visible in Fig. 5b that the automatically found HiM is able to achieve the best duty cycle and perform 3.95 times more classifications per hour in comparison to the flat model, coming very close to real-time monitoring of the machine in question. Additionally, the gap between the two classification scenarios and the wireless sending mode shows that filtering and classifying the sensor data should always be the preferred method if applicable. Considering that in other more extreme cases BLE might not be available due to its low range, different communication methods might draw even more energy. Therefore, an efficient filtering method should be especially useful in remote areas. Additionally, when looking at this test from a battery-life point of view, the more efficient algorithms could also be used to increase battery life, or decrease the battery capacity instead of applying a higher duty cycle. This perspective helps to improve sustainability in an industry 4.0 scenario and to enable further, new applications.

5 Conclusion

With the HiMLEdge framework, we were able to evaluate 12,964 hierarchical model architectures to find the best performing one w.r.t. accuracy *and* energy usage. The optimized HiMs achieve a reduction in energy consumption of up to 47.63% over the baseline in the standard distribution of the CWRU dataset (25% no-faults). These improvements are mainly linked to (i) the anomaly-detection as the first stage of evaluation and (ii) the modularized selection of features and classifiers. We explored RL as a selection method for HiMs, showing the potential behind more efficient optimization algorithms. We have demonstrated that the viability of an energy approximation measure needs to be validated before applying it to the optimization process. In our case, the unrealistic approximation lead to a different optimal policy in comparison to the optimal policy obtained by including real energy measurements. The system-level view on hierarchical classification exhibits the strength of efficient processing at the edge. It lead to a near real time execution of the monitoring while using only a coin cell battery for five years of run time. In a more realistic dataset distribution (95% no-fault), the HiM was able to perform 3.95 more readings per hour than a flat classifier. On the flip side this advantage in energy efficiency could also be transferred to longer battery life.

However, there are additional points that need to be investigated in future works. Further optimization techniques should be explored, e.g., evolutionary algorithms [3]. Even though RL showed promising results, it is unclear how it would handle an increased search space. The optimization process might also be positively influenced by a surrogate model that precisely estimates the energy consumption of a pipeline [6]. Additionally, further modularization of features

could improve the energy consumption by, e.g., utilizing a Multirate Filterbank [16] to decompose the input signals. Furthermore, the hierarchy is constructed with expert knowledge, so an automatic approach to learn hierarchical structures should be investigated as an alternative. Currently it is unclear how beneficial the approach would be in other scenarios. Even though we were able to show that in the case of condition monitoring, HiML can be very beneficial, other application areas might not benefit as much. Therefore, a study comparing the impact of HiML on energy efficiency in a broad range of contexts should be conducted in future work.

Acknowledgement. This work was supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics - Data - Applications (ADACenter) within the framework of “BAYERN DIGITAL II” (20–3410-2–9-8).⁴

References

1. Adams, S., et al.: Hierarchical fault classification for resource constrained systems. *Mech. Syst. Signal Process.* **134**, 106266 (2019) <https://doi.org/10.1016/j.ymssp.2019.106266> <https://linkinghub.elsevier.com/retrieve/pii/S0888327019304819>
2. Akbari, A., Wu, J., Grimsley, R., Jafari, R.: Hierarchical Signal Segmentation and Classification for Accurate Activity Recognition. In: *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. pp. 1596–1605. *UbiComp '18*, Association for Computing Machinery, New York, NY, USA (2018). <https://doi.org/10.1145/3267305.3267528>
3. Aquino-Brítez, D., Ortiz, A., Ortega, J., León, J., Formoso, M., Gan, J.Q., Escobar, J.J.: Optimization of Deep Architectures for EEG Signal Classification: An AutoML Approach Using Evolutionary Algorithms. *Sensors* **21**(6), 2096 (2021) DOI: <https://doi.org/10.3390/s21062096>, <https://www.mdpi.com/1424-8220/21/6/2096>, number: 6 Publisher: Multidisciplinary Digital Publishing Institute
4. Bischl, B., Mersmann, O., Trautmann, H., Preuß, M.: Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In: *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. pp. 313–320. *GECCO '12*, Association for Computing Machinery, New York, NY, USA (2012). <https://doi.org/10.1145/2330163.2330209>
5. Bruckert, S., Finzel, B., Schmid, U.: The Next Generation of Medical Decision Support: A Roadmap Toward Transparent Expert Companions. *Front. Artif. Intell.* **3** (2020) <https://www.frontiersin.org/article/10.3389/frai.2020.507973>
6. García-Martín, E., Rodrigues, C.F., Riley, G., Grahn, H.: Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing* **134**, 75–88 (2019) <https://doi.org/10.1016/j.jpdc.2019.07.007>, <https://www.sciencedirect.com/science/article/pii/S0743731518308773>
7. Goetschalckx, K., Moons, B., Lauwereins, S., Andraud, M., Verhelst, M.: Optimized Hierarchical Cascaded Processing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **8**(4), 884–894 (2018). <https://doi.org/10.1109/JETCAS.2018.2839347>

8. Lei, Y.: 2 - Signal processing and feature extraction. In: Lei, Y. (ed.) *Intelligent Fault Diagnosis and Remaining Useful Life Prediction of Rotating Machinery*, pp. 17–66. Butterworth-Heinemann (Jan 2017). <https://doi.org/10.1016/B978-0-12-811534-3.00002-0>, <https://www.sciencedirect.com/science/article/pii/B9780128115343000020>
9. Neupane, D., Seok, J.: Bearing Fault Detection and Diagnosis Using Case Western Reserve University Dataset With Deep Learning Approaches: A Review. *IEEE Access* 8 (2020). <https://doi.org/10.1109/ACCESS.2020.2990528>
10. Ns, A.: Time complexity analysis of support vector machines (SVM) in LibSVM. *Int. J. Comput. Appl.* **128**(3), 957–8887 (2015). <https://doi.org/10.5120/ijca2015906480>
11. Pech, M., Vrchota, J., Bednář, J.: Predictive maintenance and intelligent sensors in smart factory: Review. *Sensors* 21(4) (2021). <https://doi.org/10.3390/s21041470>, <https://www.mdpi.com/1424-8220/21/4/1470>
12. Ren, H., Anicic, D., Runkler, T.A.: The synergy of complex event processing and tiny machine learning in industrial IoT. In: *Proceedings of the 15th ACM International Conference on Distributed and Event-based Systems*, pp. 126–135. DEBS '21, Association for Computing Machinery, New York, NY, USA (Jun 2021). <https://doi.org/10.1145/3465480.3466928>
13. Schwartz, D., Selman, J.M.G., Wrege, P., Paepcke, A.: Deployment of Embedded Edge-AI for Wildlife Monitoring in Remote Regions. In: *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1035–1042 (Dec 2021). <https://doi.org/10.1109/ICMLA52953.2021.00170>
14. Silla, C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* **22**(1), 31–72 (2011). <https://doi.org/10.1007/s10618-010-0175-9>
15. Thomas, A., Guo, Y., Kim, Y., Aksanli, B., Kumar, A., Rosing, T.S.: Hierarchical and distributed machine learning inference beyond the edge. In: *2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 18–23 (2019). <https://doi.org/10.1109/ICNSC.2019.8743164>
16. Vaidyanathan, P.: Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial. In: *Proceedings of the IEEE* **78**(1), 56–93 (1990). <https://doi.org/10.1109/5.52200>
17. Vinuesa, R., et al.: The role of artificial intelligence in achieving the Sustainable Development Goals. *Nature Commun.* **11**(1), 233 (2020). <https://doi.org/10.1038/s41467-019-14108-y>
18. Zhou, F., Gao, Y., Wen, C.: A Novel Multimode Fault Classification Method Based on Deep Learning. *J. Control Sci. Eng.* 2017 (2017). <https://doi.org/10.1155/2017/3583610>, <https://www.hindawi.com/journals/jcse/2017/3583610/>