Accepted for presentation in: 2020 IEEE International Conference on Communications (ICC)

LIMITS: Lightweight Machine Learning for IoT Systems with Resource Limitations

Benjamin Sliwa[1], **Nico Piatkowski**[2], **and Christian Wietfeld**[1]
[1]Communication Networks Institute, [2]Department of Computer Science, AI Group
TU Dortmund University, 44227 Dortmund, Germany
e-mail: {Benjamin.Sliwa, Nico.Piatkowski, Christian.Wietfeld}@tu-dortmund.de

***Abstract*—Exploiting big data knowledge on small devices will pave the way for building truly cognitive Internet of Things (IoT) systems. Although machine learning has led to great advancements for IoT-based data analytics, there remains a huge methodological gap for the deployment phase of trained machine learning models. For given resource-constrained platforms such as Microcontroller Units (MCUs), model choice and parametrization are typically performed based on heuristics or analytical models. However, these approaches are only able to provide rough estimates of the required system resources as they do not consider the interplay of hardware, compiler-specific optimizations, and code dependencies. In this paper, we present the novel open source framework LIghtweight Machine learning for IoT Systems (LIMITS), which applies a platform-in-the-loop approach explicitly considering the actual compilation toolchain of the target IoT platform. LIMITS focuses on high-level tasks such as experiment automation, platform-specific code generation, and sweet spot determination. The solid foundations of validated low-level model implementations are provided by the coupled well-established data analysis framework Waikato Environment for Knowledge Analysis (WEKA). We apply and validate LIMITS in two case studies focusing on cellular data rate prediction and radio-based vehicle classification, where we compare different learning models and real world IoT platforms with memory constraints from 16 kB to 4 MB and demonstrate its potential to catalyze the development of machine learning-enabled IoT systems.**

I. Introduction

Ubiquitously deployed intelligent IoT [1] systems are expected to revolutionize smart city applications such as cognitive energy management [2] as well as smart logistics and intelligent traffic control in Intelligent Transportation Systems (ITSs) [3]. Moreover, future 6G communication networks are envisioned to be massively exploiting data-driven methods [4] for network optimization in real time.

Although machine learning-enabled communication has been demonstrated to achieve massive performance improvements, e.g., through application of *anticipatory* communication methods [5], the deployment of such methods is not straightforward. State-of-the-art machine learning models are most often trained *offline* in domain-specific languages—e.g., `python`, `R`, and `MATLAB`—however, resource-constrained IoT platforms and MCUs typically operate on `C/C++` code. As automated processes for the transition from offline training to on-device prediction are missing, the deployment phase of the trained model can be rather cumbersome. Moreover, for highly memory-constrained platforms—especially for ultra low power MCUs—even model choice and correct parametrization are challenging, as the trained models might easily exceed the capabilities of the target platform. Pure analytical memory usage analyses of machine learning models are mostly *optimistic* as they only consider the overall variable size of the model itself. External dependencies (e.g., mathematical libraries for the activation functions of Artificial Neural Networks (ANNs)) as well as the implementation of the logical data flow are not considered. However, the estimates can also be *pessimistic* as they do not consider model size reduction techniques (e.g., pruning of Classification and Regression Trees (CARTs)) and compiler-specific optimization.

In this paper, we present the novel open source framework LIMITS, which aims to catalyze rapid prototyping of machine learning-enabled IoT systems. LIMITS provides an automated process for model comparison and parametrization with *sweet spot* determination as well as generation of platform-specific `C/C++` code. The proposed framework implements a hybrid approach, where the low-level training of the machine learning models is performed using the well-known WEKA framework. Considering the actual compilation toolchain of the target IoT platform, LIMITS is then able to perform a deep inspection of the trained model, including the actual memory consumption with all involved optimization steps. We illustrate typical data analytics tasks based on two case studies focusing on cellular data rate prediction and radio fingerprinting-based vehicle classification, both shown in Fig. 1.

The contributions provided by this paper can be summarized as follows:

- Presentation of the novel **open source**[1] LIMITS framework for automating machine learning tasks (performance evaluation, model selection, model visualization) in the IoT and wireless communication domains.
- **Automatic generation of platform-specific `C/C++` code**

[1] Source code is available at https://github.com/BenSliwa/LIMITS

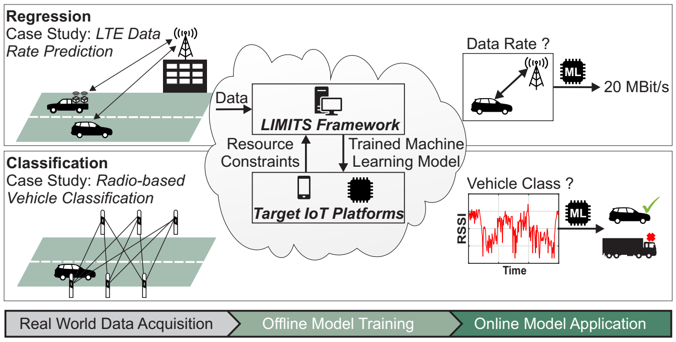


Fig. 1. Example application scenarios: Models are trained offline and deployed to a target IoT platform for performing online predictions.

for online application of the trained models on a given resource-constrained target platform and **platform-in-the-loop sweet spot** determination.
- **Proof-of-concept** study with real world data sets for Long Term Evolution (LTE) uplink data rate prediction and radio-based vehicle classification and **deployment on different resource-constrained IoT** platforms.

After discussing the related work in Sec. II, we present the LIMITS framework in Sec. III. The considered IoT platforms and performance indicators are introduced in Sec. IV and the case studies are discussed in Sec. V.

II. Related Work

Machine learning has significantly stimulated research in cognitive optimization of wireless communication systems. Comprehensive introductions with a domain-specific perspective are provided by [6] and [7]. In this work, we focus on *supervised* learning models for *classification* and *regression*—such as Random Forest (RF) [8], M5 Regression Tree (M5) [9], Support Vector Machine (SVM) [10], and ANN [11]. Recent advances in intelligent communication networks have demonstrated the potential benefits from using machine learning-based channel assessment for increasing the end-to-end data rate in vehicular cellular networks [12], [13], for reinforcement learning-based opportunistic data transfer [14] as well as for optimizing power consumption of mobile User Equipments (UEs) [15], [16], and Data-driven Network Simulation (DDNS) [17]. Although the considered regression case study in Sec. IV focuses on client-based data rate prediction, first analyses have pointed out the potentials of cooperative prediction methods which might be realized in future 6G networks [18].

Impressive results of machine learning—in particular with *deep ANNs*—have been demonstrated, mostly in the computer vision and speech domains. However, in the communication networks domain, different principles apply, which influences the choice of analysis methods and models. Compared to aforementioned domains, the amount of training data is typically relatively low due to the highly application-centric nature of the analysis tasks. Data sets are usually acquired manually based on real world experiments. As a consequence, deep learning methods are often outperformed by simpler approaches, e.g., CART-based models, which require less data to reach a satisfying performance level [13]. A positive side effect—which is exploited for sweet spot determination in Sec. III-C—of the small data sets is the comparably short training time, which allows to train and compare a multitude of different models and parameterizations for a given task.

Resource-constrained IoT systems have raised a keen interest of the research community due to their sensing and communication capabilities as well as their cost efficiency [19], [20]. In [21], the authors compare the classification performance of different models for a positioning task in an industrial environment. Although a memory-constrained target platform is considered, the memory consumption of the machine learning models is only derived analytically based on model complexity considerations. However, as investigated in Sec. V, there is a significant gap between analytical estimations and the actual platform-specific resource occupation.

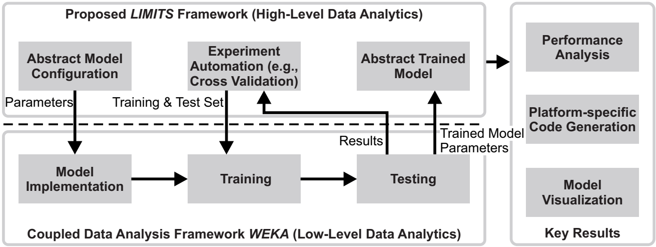

Proposed LIMITS Framework (High-Level Data Analytics)
Command Line Interface
Abstract Model Configuration
Experiment Automation (e.g., Cross Validation)
Abstract Trained Model
Parameters
Training & Test Set
Results
Trained Model Parameters
Model Implementation
Training
Testing
Coupled Data Analysis Framework WEKA (Low-Level Data Analytics)
Performance Analysis
Platform-specific Code Generation
Model Visualization
Key Results

Fig. 2. Overall system architecture of the proposed LIMITS data analysis framework.

As an alternative approach for deploying generic machine learning models on IoT platforms, other approaches aim to optimize the memory efficiency of the algorithms themselves. E.g., model compression techniques [22], [23] stem from the machine learning community and do not take the actual hardware into account. In [24], the *Bonsai* model is presented which consists of a sparse tree model that is learned in a low-dimensional feature space through projection. The model is deployed to a `ATmega328P` platform—which is also considered in this paper—with 2 KB Random Access Memory (RAM) and 32 KB program memory. In comparison to cloud-based offloading (e.g., as proposed by [25]) the local execution consumes 47-497 times less energy. Another optimization approach is the avoidance of floating point arithmetic. In [26], the authors propose integer undirected models for performing probabilistic inference on resource-constrained systems.

Data analysis tools: In addition to well-established mathematical tool such as `MATLAB` and `R`, a wide range of different machine learning tools has emerged. RapidMiner [27] is a commercial and graphical data analysis framework focusing on business analytics. Although a free version is provided for academic usage, it is limited in the number of data values. Konstanz Information Miner (KNIME) [28] provides similar functions and an open source licensing model. WEKA [29] is a `Java`-based open source framework for machine learning with limited graphical features but a rich Command Line Interface (CLI), which makes it a powerful tool for automation.

Recently, `python`-based frameworks focusing on deep learning such as Scicit-learn [30], PyTorch [31], and TensorFlow [32] enjoy a great popularity. Although the latter toolkit has made initial attempts to support data analysis on embedded devices with *TensorFlow Lite*, the available models and supported platforms are limited and focus on typical ANN tasks such as image recognition.

III. Machine Learning with LIMITS

In this section, we introduce the proposed LIMITS framework and highlight different capabilities focusing on automating data analytics and deployment of trained models to IoT platforms. The general process for performing machine learning-based data analysis with LIMITS is illustrated in Fig. 2. LIMITS provides a convenient `python`-based interface for performing high-level machine learning tasks, ready to analyze the performance of multiple models or configurations. It uses an abstract model configuration, for which the parameters are close to the analytical model descriptions. The actual low-level machine learning processes are performed by WEKA, which provides a solid foundation of validated model implementations. Based on the reported result parametrization of the trained model (e.g., weight matrices for ANNs, abstract trees for CARTs), LIMITS derives an abstract intermediate representation of the trained model, which can then be used to generate platform-specific code.

As LIMITS only considers abstract models and the whole interaction with the coupled WEKA is performed through dedicated interfaces, it is highly extensible and can be extended to support additional low-level frameworks in the future.

```
$ ./cli.py -r ../examples/mnoA.csv -m rf,m5,ann

r2               mae               rmse
0.78+/-0.01      2.95+/-0.24       4.01+/-0.21
0.78+/-0.02      2.76+/-0.12       3.99+/-0.18
0.79+/-0.01      3.34+/-0.33       4.35+/-0.32
```

Lst 1. Example for CLI-based machine learning for rapid result analysis.

Lst. 1 shows an example usage of the CLI where three different models (RF, M5, and ANN) are applied to perform a regression task on the data set `mnoA.csv`. It can be seen that the CLI provides a lightweight interface for rapidly assessing the performance of different models. More detailed model configurations and parameter definitions can be performed based on the integrated `python` interface.

A. Machine Learning Models

In its current version, the proposed LIMITS framework supports data analysis and code generation for different well-known *supervised* machine learning models.

Artificial Neural Networks (ANNs) [11] consist of different layers of interconnected nodes which are referred to as *neurons* in analogy to the cognitive systems of living creatures. Recently, *Deep Neural Networks*, which are ANNs with more than one hidden layer, have received great attention due to their superior performance in tasks such as image classification. Those models can be implemented as a series of matrix vector multiplications followed by component-wise neuron activation functions. The resulting memory occupation is mainly related to the weight matrices and threshold vectors. For a fully connected ANN with a layer configuration L of N layers, the resulting memory usage $\mathrm{M}_{\mathrm{ANN}}$ is estimated as

$$\mathrm{M}_{\mathrm{ANN}} = \sum_{i=2}^{N} L_i(1 + L_{i-1}) \tag{1}$$

Support Vector Machine (SVM) [10] learn a hyperplane that separates real-valued data points in a d dimensional hyperspace by minimizing a specific objective function. d corresponds to the dimensionality of the provided feature vector. LIMITS currently provides code generation for linear L2/L2 Support Vector Machine (SVM). For multi-class problems with n classes, the *one-vs-one* strategy is applied and the memory usage M_{SVM} can be estimated as

$$M_{\mathrm{SVM}} = \frac{n(n-1)}{2} d \tag{2}$$

M5 Regression Tree (M5) [9] is a regression tree method where each of the leaves contains a linear regression model. The resulting memory consumption of tree-based methods cannot be easily calculated proactively, as the tree structure is derived within the training process and its optimization steps.

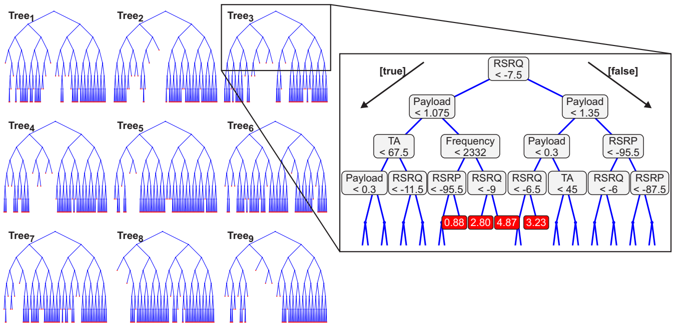


Fig. 3. Example model visualization for a trained Random Forest (RF) with nine trees and maximum depth 7. The overlay shows an excerpt of the actual decision making within one of the trees.

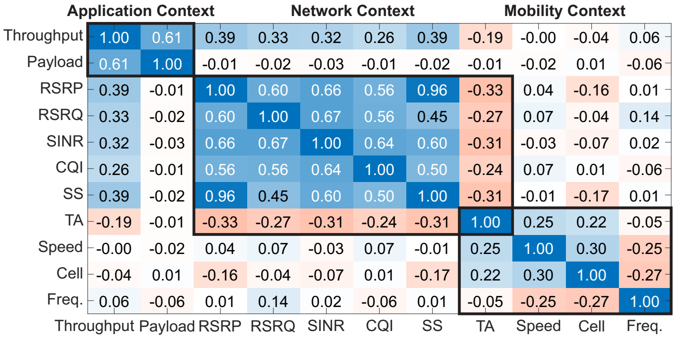


Fig. 4. Correlation-based feature analysis of the Android-based cellular network measurements. The different context classes can be clearly identified.

Although it is possible to derive worst case estimations for the tree size, the resulting value range is not meaningful enough for the considered MCU platforms which require fine-grained model size estimations. M5 can be applied for regression but not for classification.

Random Forests (RFs) [8] in an ensemble method based on multiple randomized CART trees where each tree is learned from a random sub-set of the training data. The leaves contain the prediction results and the final result is obtained by averaging over all tree results. An example visualization of a trained RF is shown in Fig. 3. RFs are frequently used for network quality prediction (e.g., [13], [33], [34] and often outperform methods which require a larger amount of training data for reaching a similar level of prediction performance.

B. Automation of Data Analytics Tasks

LIMITS supports the automatic execution of different data analysis tasks:

- **Correlation analysis** is an intuitive way of feature selection for removing redundancies within a data set [35]. Fig. 4 shows an example data set of Android-based LTE performance indicator measurements (further details about the methodological aspects on the features are discussed in Sec. IV). It can be seen that Reference Signal Received Power (RSRP) and Signal Strength (SS) have a high cross-correlation. In fact, the SS reported by the Android operating system represents the Arbitrary Strength Unit (ASU) which is derived as $\mathrm{ASU} = \mathrm{RSRP} + 140$. Such redundancies should be removed from the data set in order to optimize the information gain or whenever machine learning methods assume independent features.

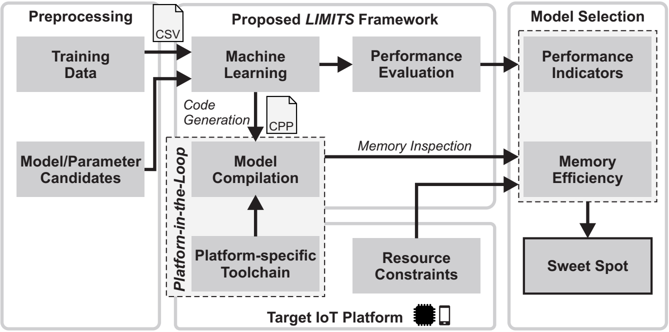


Fig. 5. Platform-in-the-loop model selection process which is performed with respect to the platform-specific compilation toolchain in order to determine the *sweet spot* of the model.

- **Cross validation** considers multiple training and test scenarios from one data set in order to achieve deep insights into a model's mean performance as well as its standard deviation. Relying on a single train/test split of the data could deliver highly over- or underconfident estimates of a model's performance. As LIMITS focuses on real world model deployments, a more robust assessment approach is therefore applied.
- **Experiments** are used to compare the performance of multiple models/parametrizations on a single data set.
- **Multi experiment** analyze the performance of a single model on N data sets. This method is applied to analyze if a machine learning model generalizes well on different data sets (e.g., different evaluation tracks, network data for multiple Mobile Network Operators (MNOs) [13]). The results are NxN matrices for each performance indicator.
- **Feature importance** analysis based on model-specific indicators, e.g., Mean Decrease Impurity (MDI) [36].

C. Platform-in-the-Loop Model Selection

The proposed platform-in-the-loop approach allows us to determine the Pareto optimal configuration—the *sweet spot*—with respect to quality and resource allocation of each model for a given platform. More precisely, the goal is to find the model parametrization with the highest prediction accuracy, which just fulfills the memory requirements of the target platform.

An overview about the provided automated model selection process is shown in Fig. 5. For each *candidate*, a model is trained on the supplied training data and the corresponding `C/C++` code for the model is generated automatically. The latter is compiled with the toolchain of the target platform, which allows to inspect the real resulting memory occupation. Finally, the sweet spot is determined based on the achieved model performance and its resource efficiency.

IV. Methodology

In the following, we discuss the case studies, performance indicators, and target IoT platforms.

A. Case Studies and Data Sets

As the major contribution of this work is of methodological nature, we utilize existing data sets, which have been acquired in previous work.

Regression is considered with a case study focusing on client-based LTE uplink data rate prediction in vehicular scenarios [13]. Measurements for the context indicators RSRP, Reference Signal Received Quality (RSRQ), Signal-to-interference-plus-noise Ratio (SINR), Channel Quality Indicator (CQI), Timing Advance (TA), velocity, cell id, carrier frequency, and payload size are utilized by the UE to forecast the achievable data rate for each to be performed Transmission Control Protocol (TCP) data transfer. The considered data set consists of 3907 transmissions, which were performed in the public cellular network and in four different scenarios (urban, suburban, highway, campus).

Classification is considered with a radio-based vehicle classification system [37]. Hereby, an installation of six communicating IEEE 802.15.4 nodes is installed with three nodes on each of the road sides (see the illustration in Fig. 1). The Reference Signal Strength Indicator (RSSI) level of the resulting nine different radio links Φ_i is monitored continuously. If a vehicle passes the installation, the resulting attenuation pattern results in a *radio fingerprint*, which is characteristic for the type of the vehicle. Seven different vehicle classes are considered: *Passenger car, Passenger car with trailer, Van, Truck, Truck with Trailer, Semitruck, Bus*. The considered data set consists of 2605 time series traces.

B. Performance Indicators

The performance of the data rate prediction models is assessed by means of the *coefficient of determination* R^2 (a.k.a. amount of explained variance), which is a statistical metric and allows to compare the achieved results with other related approaches that consider the same indicators [33], [34]. It is calculated as

$$R^2 = 1 - \frac{\sum_{i=1}^{N} \left(\tilde{y}_i - y_i\right)^2}{\sum_{i=1}^{N} \left(\bar{y} - y_i\right)^2} \tag{3}$$

with $\tilde{y}_i$ being the current prediction, y_i being the current measurement, and $\bar{y}$ being the mean measurement value. Further performance metrics for the validation of the code generator are Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

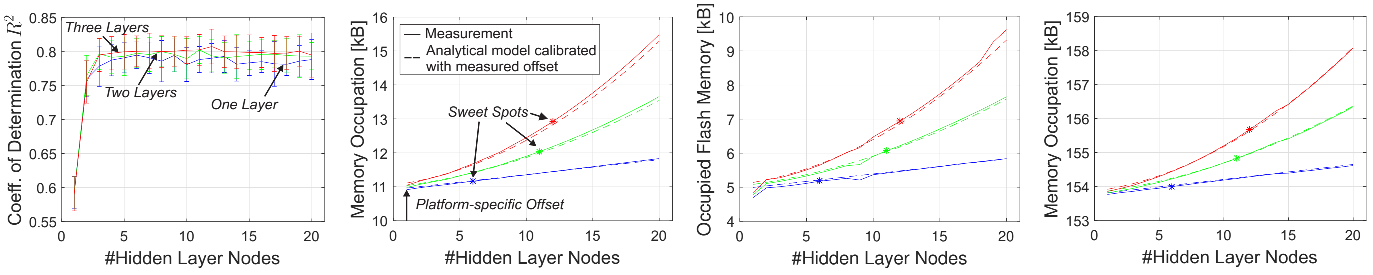
For the vehicle classification task, we mainly focus on analyzing *accuracy* (which is also referred to as *classification success ratio* in related work). For a given data set $\mathcal{D}$, a model f is trained to make predictions on unlabeled data $\mathbf{x}$ such that $\tilde{y} = f(\mathbf{x})$. The accuracy ACC is then derived as

$$\mathrm{ACC}(f; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(y,\mathbf{x}) \in \mathcal{D}} 1_{\{y=f(\mathbf{x})\}} \tag{4}$$

with $|\mathcal{D}|$ being the cardinality of $\mathcal{D}$ and $1_{\{y=f(\mathbf{x})\}}$ being the indicator function that only evaluates to 1 if $f(\mathbf{x})$ outputs the correct class y and is 0 otherwise. For the validation of the code generator, we further consider precision, recall and F$_1$-score.

C. Target IoT Platforms for the Performance Evaluation

For the performance evaluation in Sec. V, we evaluate three popular target IoT platforms with different resource requirements and computation capabilities.



(a) Model Performance (b) `MSP430` (c) `ATmega328` (d) `ESP32`

Fig. 6. Sweet spot determination for the data rate prediction data set with ANN and different values for number of hidden layers and number of nodes per hidden layer. The error bars show the standard deviations of the 10-fold cross validation.

`MSP430` (Model G2553) is a 16 Bit ultra low power microcontroller with 16 MHz, 16.35 kB program memory and 512 Byte RAM, which is programmed with `Code Composer Studio`. The compiler is configured to apply memory-centric optimization with disabled *loop unrolling* features.

`ATmega328` (Model ATmega328P) is a 16 MHz MCU with 32 kB program memory 2 kB RAM. Compilation and deployment are performed with the popular `Arduino` toolkit.

`ESP32` (Model ESP-WROOM-32) is a 32 Bit MCU with a 240 MHz dual core, 4 MB program memory and 532 kB RAM with integrated bluetooth and WiFi support. Compilation and deployment tools are provided by `Espressif IoT Development Framework (ESP-IDF)`.

V. Results

In this section, models for a given IoT platform are selected and the generated model implementations are validated against the WEKA results. All data analysis evaluations are 10-fold cross validated.

A. Platform-specific Model Selection

Sweet spots are identified by optimizing the model specific hyper-parameters on each platform.

For the ANN, we analyze the impact of different amounts of hidden layers and the number of nodes on each hidden layer. Fig. 6 shows the resulting coefficient of determination R^2 and the occupied program memory resources for all platforms on the data rate prediction data set. Although it can be seen that the analytical model with 4 Byte `float` data types (see Eq. 1) provides a good estimate for the real memory occupation, it contains a *platform-specific offset* which is unknown if only the analytical model is considered. For the ESP32, it needs to be denoted that the memory footprint can be reduced by disabling communication capabilities from the compilation process. However, as we aim to mimic typical application scenarios, we kept the default configuration of the MCU.

For the RF model, we vary the number of random trees and the maximum tree depth. Fig. 7 shows the results for the data rate prediction data set. While the `ESP32` is able to consider the whole parameter space, `ATmega328` and `MSP430` are significantly impacted by memory limitations.

For each model, the results of the platform-specific sweet spot parametrizations are summarized in Tab. I for the regression task and in Tab. II for the classification task with sweet spot parameters for ANN and RF.

TABLE I
Regression Model Performance and Program Memory Occupation for Different IoT Platforms

Model	`MSP430`		`ATmega328`		`ESP32`	
	$\mathbf{R^2}$	Memory	$\mathbf{R^2}$	Memory	$\mathbf{R^2}$	Memory
ANN {H, N}	**0.807** ±0.025	**12.62 kB** {3, 12}	**0.807** ±0.025	**6.77 kB** {3, 12}	0.807 ±0.025	152.03 kB {3, 12}
M5	0.772 ±0.03	5.15 kB	0.772 ±0.03	3.7 kB	0.772 ±0.03	149.80 kB
RF {T, D}	0.788 ±0.022	14.59 kB {5, 6}	0.801 ±0.022	24.76 kB {4, 8}	**0.829** ±0.013	**1307 kB** {30, 15}
SVM	0.551 ±0.03	3.72 kB	0.551 ±0.03	7.88 kB	0.551 ±0.03	148.70 kB

H: #Hidden layers, *N*: #Neurons on hidden layer, *T*: #Trees, *D*: Max. depth

TABLE II
Classification Model Performance and Program Memory Occupation for Different IoT Platforms

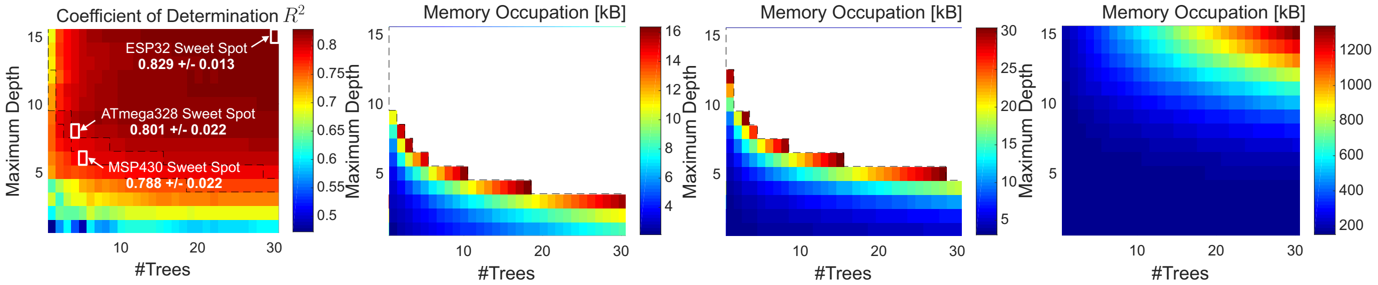
Model	`MSP430`		`ATmega328`		`ESP32`	
	Accuracy	Memory	Accuracy	Memory	Accuracy	Memory
ANN {H, N}	**93.79** ±1.15	**13.24 kB** {1, 24}	**93.79** ±1.15	**6.51 kB** {1, 24}	**93.79** ±1.15	**152.4 kB** {1, 24}
RF {T, D}	93.05 ±0.87	16.12 kB {5, 5}	93.48 ±1.32	28.54 kB {5, 13}	93.67 ±1.37	256.71 kB {12, 13}
SVM	92.48 ±0.62	5.45 kB	92.48 ±0.62	9.22 kB	92.48 ±0.62	151.04 kB

H: #Hidden layers, *N*: #Neurons on hidden layer, *T*: #Trees, *D*: Max. depth

B. Runtime Complexity

Enumerating the resource requirements of the full hyper-parameter space (as shown in Fig. 7) comes with an undeniable computational overhead. Models which allow for fast training are thus especially well suited for our proposed system. The training times of considered machine learning models are shown in Fig. 8: M5 and random forests clearly outperform ANN and SVM training, which implies that sweet spots of these method can be identified much faster in practice. Stochastic gradient methods are known to suffer from sub-linear convergence, which explains the inferior runtime of ANN training. SVM, on the other hand, is trained with Platt's sequential minimal optimization. While this algorithm can be very fast occasionally, it's worst-case behavior is quadratic in the number of data points—this is likely to happen whenever almost all data points will be support vectors of the underlying model.

While training time can be mitigated by using strong computational resources, prediction time of a model on the



(a) Model Performance (b) `MSP430` (c) `ATmega328` (d) `ESP32`

Fig. 7. Sweet spot determination for the data rate prediction data set with RF and different values for number of random trees and maximum depth.

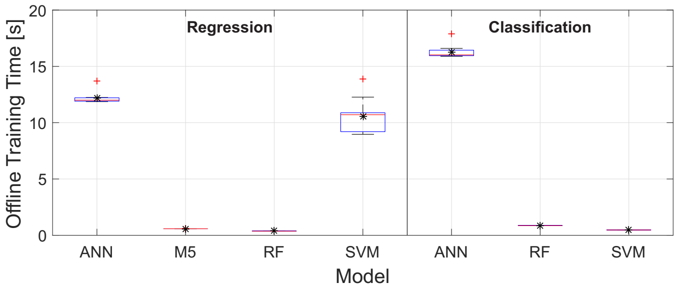


Fig. 8. Runtime of training and testing the machine learning models (offline).

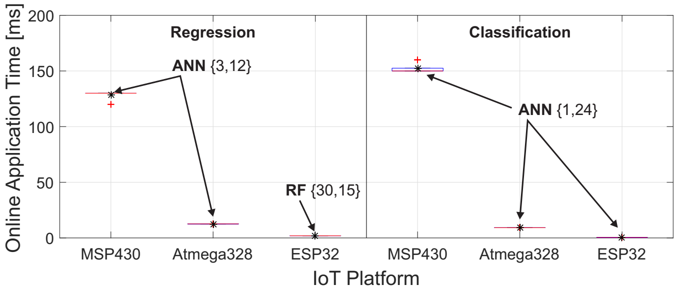


Fig. 9. Measured online execution time per prediction of the sweet spot models deployed on the considered target platforms. *ANN* {#Hidden layers, #Neurons on hidden layers}, *RF* {#Trees, #Max. depth}.

resource-constrained device can render a task impractical. Thus, we investigate how the program code that is generated by LIMITS performs on each platform. In the following, the best model is deployed for each target IoT platform and the execution time per single prediction is determined over 1000 online predictions. Fig. 9 shows the resulting measurement values for the regression and classification tasks.

Each test platform achieves prediction rates between 8Hz and 100Hz—sufficient for our example applications. Nevertheless, the `MSP430` which has by far the lowest energy requirements is outperformed by a large margin by both, the `ATmega328` and the `ESP32` platforms.

C. Code Generator Validation

In order to validate the code generator implementation, we replay all measurements with the generated `C/C++` models and compare the statistical properties to the WEKA results. The results are shown in Tab. III for the regression models and in Tab. IV for the classification models.

TABLE III
VALIDATION OF THE CODE GENERATOR FOR REGRESSION MODELS

Model	**WEKA**			**Generated Model**		
	$\mathbf{R^2}$	**MAE**	**RMSE**	$\mathbf{R^2}$	**MAE**	**RMSE**
ANN	0.807 ±0.026	2.66 ±0.127	3.731 ±0.234	0.807 ±0.026	2.66 ±0.157	3.731 ±0.234
M5	0.772 ±0.03	2.773 ±0.81	4.022 ±0.206	0.771 ±0.03	2.784 ±0.77	4.033 ±0.206
RF	0.829 ±0.018	2.457 ±0.85	3.485 ±0.133	0.826 ±0.017	2.475 ±0.90	3.514 ±0.137
SVM	0.552 ±0.03	4.351 ±0.147	5.666 ±0.192	0.551 ±0.03	4.36 ±0.149	5.68 ±0.19

TABLE IV
VALIDATION OF THE CODE GENERATOR FOR CLASSIFICATION MODELS

Model	**WEKA**				**Generated Model**			
	ACC	**PREC**	**REC**	$\mathbf{F_1}$	**ACC**	**PREC**	**REC**	$\mathbf{F_1}$
ANN	93.79 ±1.16	96.59 ±0.84	97.19 ±1.03	96.89 ±0.61	93.79 ±1.16	96.59 ±0.84	97.19 ±1.03	96.89 ±0.61
RF	93.67 ±1.58	96.15 ±1.26	97.31 ±0.7	96.72 ±0.89	93.52 ±1.51	95.97 ±1.19	97.29 ±0.74	96.62 ±0.81
SVM	92.48 ±0.62	94.97 ±0.79	97.56 ±0.55	96.24 ±0.43	92.48 ±0.66	95.05 ±0.85	97.56 ±0.56	96.29 ±0.49

ACC: Accuracy, *PREC*: Precision, *REC*: Recall, F_1: F_1 Score

It can be seen that the generated models achieve an accurate match with the ground truth provided by WEKA. However, minor deviations occur as the the latter exposes rounded parameters for some of the machine learning models.

VI. CONCLUSION

In this paper, we presented LIMITS—a novel open source machine learning framework for IoT applications, which provides automation features for high-level data analysis tasks and platform-specific code generation. In contrast to existing solution approaches, LIMITS explicitly integrates the platform-specific resource constraints and compilation toolchain of the target IoT platform into the model selection process. Its potential of catalyzing the development of machine learning-enabled IoT systems was demonstrated based on two case studies focusing on cellular data rate prediction in vehicular networks and radio-based vehicle classification. In future work, we will integrate further machine learning models into LIMITS. Furthermore, we consider integrating automatic static Worst-case Execution Time (WCET) analysis into the model selection process.

Acknowledgment

Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", projects B4. Parts of this research have been funded by the Federal Ministry of Education and Research of Germany as part of the competence center for machine learning ML2R (01|S18038A).

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb 2014.

[2] F. Terroso-Saenz, A. González-Vidal, A. P. Ramallo-González, and A. F. Skarmeta, "An open IoT platform for the management and analysis of energy data," *Future Generation Computer Systems*, vol. 92, pp. 1066 – 1079, 2019.

[3] B. Sliwa, T. Liebig, T. Vranken, M. Schreckenberg, and C. Wietfeld, "System-of-systems modeling, analysis and optimization of hybrid vehicular traffic," in *2019 Annual IEEE International Systems Conference (SysCon)*, Orlando, Florida, USA, Apr 2019.

[4] P. Yang, Y. Xiao, M. Xiao, and S. Li, "6G wireless communications: Vision and potential techniques," *IEEE Network*, vol. 33, no. 4, pp. 70–75, July 2019.

[5] N. Bui, M. Cesana, S. A. Hosseini, Q. Liao, I. Malanchini, and J. Widmer, "A survey of anticipatory mobile networking: Context-based classification, prediction methodologies, and optimization techniques," *IEEE Communications Surveys & Tutorials*, 2017.

[6] L. Liang, H. Ye, and G. Y. Li, "Toward intelligent vehicular networks: A machine learning framework," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 124–135, Feb 2019.

[7] H. Ye, L. Liang, G. Y. Li, J. Kim, L. Lu, and M. Wu, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, June 2018.

[8] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] J. R. Quinlan, "Learning with continuous classes." World Scientific, 1992, pp. 343–348.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[12] B. Sliwa, T. Liebig, R. Falkenberg, J. Pillmann, and C. Wietfeld, "Efficient machine-type communication using multi-metric context-awareness for cars used as mobile sensors in upcoming 5G networks," in *2018 IEEE 87th Vehicular Technology Conference (VTC-Spring)*, Porto, Portugal, Jun 2018, Best Student Paper Award.

[13] B. Sliwa and C. Wietfeld, "Empirical analysis of client-based network quality prediction in vehicular multi-MNO networks," in *2019 IEEE 90th Vehicular Technology Conference (VTC-Fall)*, Honolulu, Hawaii, USA, Sep 2019.

[14] ——, "A reinforcement learning approach for efficient opportunistic vehicle-to-cloud data transfer," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, Seoul, South Korea, Apr 2020.

[15] B. Sliwa, R. Falkenberg, T. Liebig, N. Piatkowski, and C. Wietfeld, "Boosting vehicle-to-cloud communication by machine learning-enabled context prediction," *IEEE Transactions on Intelligent Transportation Systems*, Jul 2019.

[16] R. Falkenberg, B. Sliwa, N. Piatkowski, and C. Wietfeld, "Machine learning based uplink transmission power prediction for LTE and upcoming 5G networks using passive downlink indicators," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Chicago, USA, Aug 2018.

[17] B. Sliwa and C. Wietfeld, "Towards data-driven simulation of end-to-end network performance indicators," in *2019 IEEE 90th Vehicular Technology Conference (VTC-Fall)*, Honolulu, Hawaii, USA, Sep 2019.

[18] B. Sliwa, R. Falkenberg, and C. Wietfeld, "Towards cooperative data rate prediction for future mobile and vehicular 6g networks," in *2nd 6G Wireless Summit (6G SUMMIT)*. Levi, Finland: IEEE, Mar 2020.

[19] T. Park, N. Abuzainab, and W. Saad, "Learning how to communicate in the internet of things: Finite resources and heterogeneity," *IEEE Access*, vol. 4, pp. 7063–7073, 2016.

[20] S. Yao, Y. Zhao, H. Shao, S. Liu, D. Liu, L. Su, and T. Abdelzaher, "FastDeepIoT: Towards understanding and optimizing neural network execution time on mobile and embedded devices," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '18, New York, NY, USA, 2018, pp. 278–291.

[21] M. Masoudinejad, A. K. Ramachandran Venkatapathy, D. Tondorf, D. Heinrich, R. Falkenberg, and M. Buschhoff, "Machine learning based indoor localisation using environmental data in PhyNetLab warehouse," in *Smart SysTech 2018; European Conference on Smart Objects, Systems and Technologies*, June 2018, pp. 1–8.

[22] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, 2006, pp. 535–541.

[23] N. Piatkowski, S. Lee, and K. Morik, "Spatio-temporal random fields: compressible representation and distributed estimation," *Machine Learning*, vol. 93, no. 1, pp. 115–139, 2013.

[24] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 KB RAM for the internet of things," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 1935–1944.

[25] R. M. Shukla and A. Munir, "An efficient computation offloading architecture for the internet of things (iot) devices," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2017, pp. 728–731.

[26] N. Piatkowski, S. Lee, and K. Morik, "Integer undirected graphical models for resource-constrained systems," *Neurocomputing*, vol. 173, pp. 9 – 23, 2016.

[27] M. Hofmann and R. Klinkenberg, *RapidMiner: Data mining use cases and business analytics applications*. Chapman & Hall/CRC, 2013.

[28] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, "KNIME - the konstanz information miner: Version 2.0 and beyond," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 26–31, Nov. 2009.

[29] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," 2017.

[32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[33] F. Jomrich, A. Herzberger, T. Meuser, B. Richerzhagen, R. Steinmetz, and C. Wille, "Cellular bandwidth prediction for highly automated driving - Evaluation of machine learning approaches based on real-world data," in *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems 2018*, no. 4. SCITEPRESS, Mar 2018, pp. 121–131.

[34] A. Samba, Y. Busnel, A. Blanc, P. Dooze, and G. Simon, "Instantaneous throughput prediction in cellular networks: Which information is needed?" in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 624–627.

[35] K. Apajalahti, E. A. Walelgne, J. Manner, and E. Hyvönen, "Correlation-based feature mapping of crowdsourced LTE data," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1–7.

[36] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 431–439.

[37] B. Sliwa, N. Piatkowski, M. Haferkamp, D. Dorn, and C. Wietfeld, "Leveraging the channel as a sensor: Real-time vehicle classification using multidimensional radio-fingerprinting," in *2018 IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, Hawaii, USA, Nov 2018.