SustainML

Application Aware, Life-Cycle Oriented
Model-Hardware Co-Design Framework
for Sustainable, Energy Efficient ML Systems

Task modeling from user definition results (M6)

Deliverable D1.1

WP1 - Energy Consumption Oriented ML Task Modeling

This project has received funding from the European Union's Horizon Europe research and innovation programme (HORIZON-CL4-2021-HUMAN-01) under grant agreement No 101070408

Project

Title:	SustainML: Application Aware, Life-Cycle Oriented Model-Hardware Co-Design Framework for Sustainable, Energy Efficient ML Systems
Acronym:	SustainML
Coordinator:	eProsima
Grant agreement ID:	101070408
Call:	HORIZON-CL4-2021-HUMAN-01
Program:	Horizon Europe
Start:	01 October 2022
Duration:	36 months
Website:	https://sustainml.eu
E-mail:	sustainml@eprosima.com
Consortium:	**eProsima (EPROS)**, Spain **DFKI**, Germany **Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau (RPTU)**, Germany **University of Copenhagen (KU)**, Denmark **National Institute for Research in Digital Science and Technology (INRIA)**, France **IBM Research GmbH**, Switzerland **UPMEM**, France

Deliverable

Number:	**D1.1**
Title:	**Task modeling from user definition results (M6)**
Month:	6
Work Package:	WP1 - Energy Consumption Oriented ML Task Modeling
Work Package leader:	DFKI
Deliverable leader:	DFKI
Deliverable type:	R
Dissemination level:	Public (PU)
Date of submission:	2023-03-31
Version:	v1.0
Status:	FInished

Version history

Version	Date	Responsible	Author/Reviewer	Comments
v1.0	31-03-2323	DFKI	DFKI	

Executive summary

SustainML project aims to develop a design framework and an associated toolkit, so-called SustainML, that will foster energy efficiency throughout the whole life-cycle of Machine Learning (ML) applications: from the design and exploration phase that includes exploratory iterations of training, testing and optimizing different system versions through the final training of the production systems (which often involves huge amounts of data, computation and epochs) and (where appropriate) continuous online re-training during deployment for the inference process. The framework will optimize the ML solutions based on the application tasks, across levels from hardware to model architecture. It will also collect both previously scattered efficiency-oriented research, as well as novel Green-AI methods. Artificial Intelligence (AI) developers from all experience levels can make use of the framework through its emphasis on human-centric interactive transparent design and functional knowledge cores, instead of the common blackbox and fully automated optimization approaches.

This report corresponds to *Deliverable D1.1 - Task modeling from user definition results* of the SustainML project. This deliverable covers the results of T1.1, T1.2 and T1.3 from WP1. Although common approaches to similar taxonomy adopt a wide top-down approach, we consider those to be too abstract and lack of real application grounding and thus ignore the target users' perspective as discussed in section 1. We thus conduct our research in a combination of top-down abstraction, as well as building narrower yet generalisable application-oriented studies and prototypes from the bottom-up approach.

While still in the early stages of the project, several components of our work has already been published or in the review process of international academic venues:

1. A Knowledge Distillation framework for Multi-Organ Segmentation of Medaka Fish in Tomographic Image has been accepted for publication on IEEE International Symposium on Biomedical Imaging (ISBI http://2023.biomedicalimaging.org/)
2. Latent Inspector: An Interactive Tool for Probing Neural Network Behaviors Through Arbitrary Latent Activation has been submitted to the International Joint Conference on Artificia Intelligence (IJCAI https://ijcai-23.org)
3. FieldHAR: A Fully Integrated End-to-end RTL Framework for Human Activity Recognition with Neural Networks from Heterogeneous Sensors has been submitted to the IEEE International Conference on Application Specific Systems, Architectures and Processors (ASAP https://www.asap2023.org)

Contents

Executive Summary 3

Contents 4

Acronyms 5

1 Introduction 6
- 1.1 Related literature 6
- 1.2 Related toolkits 6

2 Progresses of WP1 8
- 2.1 Overall methodology 8
- 2.2 ML task modeling knowledge base 9
- 2.3 Interactive heterogeneous multimodal tool 11
- 2.4 HW resource aware ML design 11

3 Future works in the following period 12
- 3.1 Plans for WP1 12
- 3.2 Joint work with other WPs 13

References 14

Acronyms

AI	Artificial Intelligence.
CNN	convolutional neural networks.
DL	Deep Learning.
FPGA	field programmable gate arrays.
HAR	human activity recognition.
HW	Hardware.
LLM	Large Language Model.
MCU	micro-controller.
ML	Machine Learning.
RDF	Resource Description Framework.
RNN	recurrent neural networks.
RTL	register transfer level.
SOTA	State of the Art.

1 Introduction

1.1 Related literature

Defining an ML taxonomy has always been an important research topic to keep an informative summary of the large and diverse mass of ML problems, applications, and techniques. Traditionally it is common, not only in the area of AI, to structure contents within umbrella terms to clarify the perception and understanding of the proposed contents. However, there is no clear guideline on how to organize and group the ML landscape in order to maintain usability and explainability due to the ever-increasing amount of different ML areas to cover. [1]

Most of the available ML taxonomies from current research only address a specific ML topic. In [2], a hierarchical taxonomy of ML algorithms is provided by clustering well-known algorithms by its learning strategy, for instance, Principal Component Analysis and K-Means within Unsupervised Learning. The resulting graph comprises the major umbrella terms of ML topics together with their related algorithms in form of a hierarchical tree structure. More detailed ML taxonomies are presented in [3] through two structured tables. The first one introduces similar content to [2] by segmenting ML algorithms into three hierarchical partitions, starting with the ML paradigm to the actual algorithm and a description containing some algorithm-specific in-depth information at last. The second table targets the applications of ML by listing the application with a comprehensive description. This kind of taxonomy further supports the definition of ML problems and can therefore be used to support our approach of building a ML taxonomy from the users' perspective. Another application-specific taxonomy is presented in [4]. Based on multi-modal machine learning, challenges like for instance fusion or co-learning are linked to the appropriate applications.

In summary, the existing ML taxonomies are mostly kept too general and abstract. They cannot picture the whole versatility of the ML landscape with sufficient depth of detail. For this project, it is essential to merge all present information into a uniform, adaptable, and most importantly usable data structure. Therefore, we are building a knowledge base covering the whole ML task modeling from the user problem description to the technical implementation.

The existing ML taxonomies are typically based on relational database structures like tree structures or tables. Such structures tend to become unclear and are therefore unusable due to the increasing complexity of the content, requiring more progressive methods. A widespread method to resolve this issue is semantic-oriented graph structures that can cover highly complex and interconnected content while preserving the ability to explore and query the structure [5]. From the technical side, the Resource Description Framework (RDF) has been established to store graph-oriented data structures with a universal syntax.

A more natural way of representing a knowledge base is presented in [6]. Instead of storing the data in a syntax-oriented structure like RDF, this approach resolves the need for schema engineering and forwards the knowledge into a language model. From the user's perspective, the knowledge base can not only be accessed with fixed commands, moreover, the information can be queried and also presented with natural language commands.

1.2 Related toolkits

Large Language Model (LLM) like ChatGPT has recently made a huge global impact with generative AI. It is also capable of outputting code. However, the fundamental operating principle of large language models relies on generating results based on probability in the large training dataset. The limitation is apparent after some quick trials with specific expert queries. For example, we queried ChatGPT with two questions from different fields as shown in table 2. While looking at individual answers, the answer appears to provide a coherent solution. However, there lacks any reasoning as to how each listed

how do I make a python script for human activity recognition from IMU data using deep learning?
And the answer is:
To create a Python script for human activity recognition from IMU data using deep learning, you can follow the steps below:
Collect IMU data: Collect the Inertial Measurement Unit (IMU) data from the sensor. An IMU typically consists of an accelerometer, gyroscope, and magnetometer.
Preprocess the data: The IMU data may contain noise and bias that needs to be removed. You can apply signal processing techniques to preprocess the data, such as low-pass filtering, normalization, and feature extraction.
Prepare the dataset: The dataset should be split into training, validation, and test sets. The dataset should also be labeled with the corresponding human activity for each data point.
Build a deep learning model: Use a deep learning framework like TensorFlow or PyTorch to build a neural network for activity recognition. The model should be trained on the training set using the labeled data. You can use a convolutional neural networks (CNN) or a recurrent neural networks (RNN) for this task.
Evaluate the model: Evaluate the performance of the model on the validation set. You can use metrics such as accuracy, precision, and recall to evaluate the performance of the model.
Test the model: Once the model is trained and validated, you can test the model on the test set to check its real-world performance.
Here is a sample Python code snippet for activity recognition using CNN:

```
import tensorflow as tf
from tensorflow.keras.layers import Conv1D, MaxPooling1D,
                            Dropout, Flatten, Dense


# Build the CNN model
model = tf.keras.Sequential([
    Conv1D(64, kernel_size=3, activation='relu',
           input_shape=(timesteps, features)),
    Conv1D(64, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),
    Dropout(0.5),
    Conv1D(128, kernel_size=3, activation='relu'),
    Conv1D(128, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),
    Dropout(0.5),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

# Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, batch_size=64, epochs=50,
          validation_data=(X_val, y_val))

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print("Accuracy: {:.2f}%".format(accuracy * 100))
```

how do I make a python script for speech enhancement in audio data for hearing aids?

To create a Python script for speech enhancement in audio data for hearing aids, you can follow the steps below:
Collect audio data: Collect audio data from the hearing aids. You can use a microphone to record the audio data.
Preprocess the data: Preprocess the audio data by removing noise and applying gain normalization.
Prepare the dataset: The dataset should be split into training, validation, and test sets. The dataset should also be labeled with the corresponding clean speech for each data point.
Build a deep learning model: Use a deep learning framework like TensorFlow or PyTorch to build a neural network for speech enhancement. The model should be trained on the training set using the labeled data. You can use a convolutional neural network (CNN) or a recurrent neural network (RNN) for this task.
Evaluate the model: Evaluate the performance of the model on the validation set. You can use metrics such as signal-to-noise ratio (SNR), perceptual evaluation of speech quality (PESQ), and mean opinion score (MOS) to evaluate the performance of the model.
Test the model: Once the model is trained and validated, you can test the model on the test set to check its real-world performance.
Here is a sample Python code snippet for speech enhancement using a denoising autoencoder:

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv1D,
                        MaxPooling1D, UpSampling1D
from tensorflow.keras.models import Model
# Build the denoising autoencoder model
input_audio = Input(shape=(timesteps, features))
x = Conv1D(32, 3, activation='relu', padding='same')
      (input_audio)
x = MaxPooling1D(2, padding='same')(x)
x = Conv1D(64, 3, activation='relu', padding='same')(x)
x = MaxPooling1D(2, padding='same')(x)
x = Conv1D(128, 3, activation='relu', padding='same')(x)
encoded = MaxPooling1D(2, padding='same')(x)

x = Conv1D(128, 3, activation='relu', padding='same')
      (encoded)
x = UpSampling1D(2)(x)
x = Conv1D(64, 3, activation='relu', padding='same')(x)
x = UpSampling1D(2)(x)
x = Conv1D(32, 3, activation='relu', padding='same')(x)
x = UpSampling1D(2)(x)
decoded = Conv1D(1, 3, activation='sigmoid',
             padding='same')(x)

autoencoder = Model(input_audio, decoded)
autoencoder.compile(optimizer='adam',
                loss='binary_crossentropy')
# Train the model
autoencoder.fit(X_train, y_train, epochs=50,
    batch_size=128, shuffle=True,
    validation_data=(X_val, y_val))
# Evaluate the model
loss = autoencoder.evaluate(X_test, y_test, verbose=0)
print("Loss: {:.2f}".format(loss))
# Apply the model to new audio data
enhanced_audio = autoencoder.predict(audio_data)
```

Table 2: Example of using ChatGPT to generate solutions of specific ML problems

method (RNN, CNN, etc), being already very abstract, is different from each other. Also comparing both answers, an obvious common pattern can be extracted. Indeed this is a common workflow, however, the level of detail provided at each step is not sufficient for the users to directly solve their problems. And when details are queried, LLMs tend to generate hallucination answers [7]. Even the latest and most potent LLM GPT-4 has been found lacking planning in arithmetic and reasoning problems [8]. This then inspires us to develop a cataloged knowledge system by which clear explanations and logic pathways can be formed instead of uniform probabilistic language search. However, a knowledge base is not necessarily exclusive to LLMs as LLMs can be leveraged to help interact with the expert user to narrow down queries or suggest relevant information.

On the other hand, there are several community hubs as a centralized repository of ML solutions. HuggingFace (https://huggingface.co) for example hosts over 150,000 ML models as of the writing of this report. While they provide specific solutions to certain problems, the topics are largely scattered and heavily lean towards vision and language modalities. From the usability aspects, those hosted models are often difficult for ML specialists to graft and repurpose for their own projects, let alone domain experts with limited knowledge and ML code maneuver expertise.

2 Progresses of WP1

2.1 Overall methodology

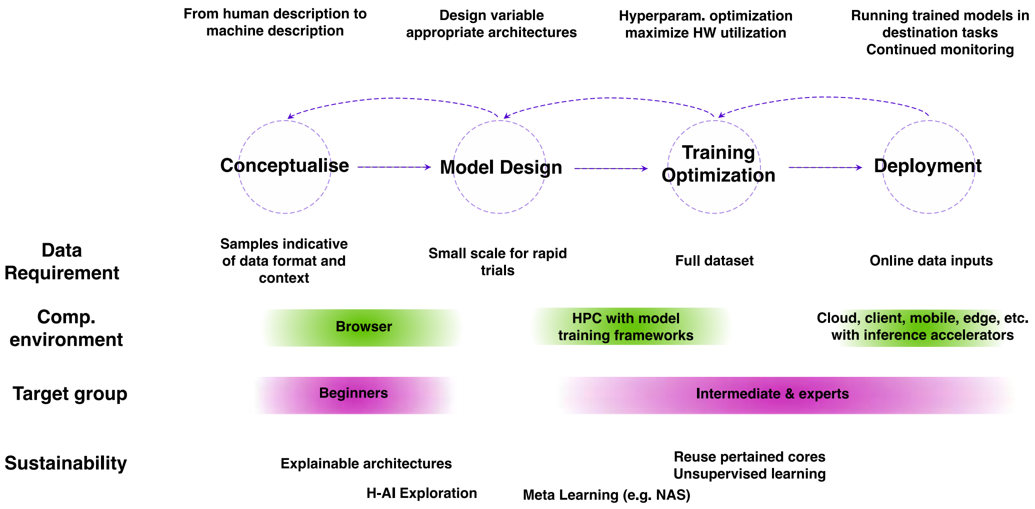
From human description to machine description
Design variable appropriate architectures
Hyperparam. optimization maximize HW utilization
Running trained models in destination tasks Continued monitoring
Conceptualise
Model Design
Training Optimization
Deployment
Data Requirement
Samples indicative of data format and context
Small scale for rapid trials
Full dataset
Online data inputs
Comp. environment
Browser
HPC with model training frameworks
Cloud, client, mobile, edge, etc. with inference accelerators
Target group
Beginners
Intermediate & experts
Sustainability
Explainable architectures
H-AI Exploration
Meta Learning (e.g. NAS)
Reuse pertained cores
Unsupervised learning

Figure 1: Development stages throughout the entire life cycle of ML model from initial conceptualisation phase to final deployment.

To provide a comprehensive solution for sustainable ML, the entire lifecycle of a ML model, which is outlined in fig. 1, needs to be taken into account. Basically, the lifecycle consists of four stages starting with the conceptualization of the model design to training and optimization of the model after which it results in the deployment phase to put the model into operation.

WP1 is mostly centralized on the first two phases, the conceptualize and model design phase, but will foresee the constraints from the downstream phases of the model lifecycle, especially the Hardware (HW) constraints. Therefore, the exchange between the project partners and work packages is essential in order to not only consider the individual stages but also the coherence throughout the entire lifecycle in terms of sustainability.

2.2 ML task modeling knowledge base

The literature provides comprehensive machine learning taxonomy. For example, in [9], from the traditional ML paradigms of supervised, unsupervised, and reinforcement learning to the modern paradigm categorizations of

1. Multi-label learning;
2. Semi-supervised learning;
3. One-class classification;
4. Positive-unlabeled learning;
5. Transfer learning;
6. Multi-task learning;
7. Few/one-shot learning.

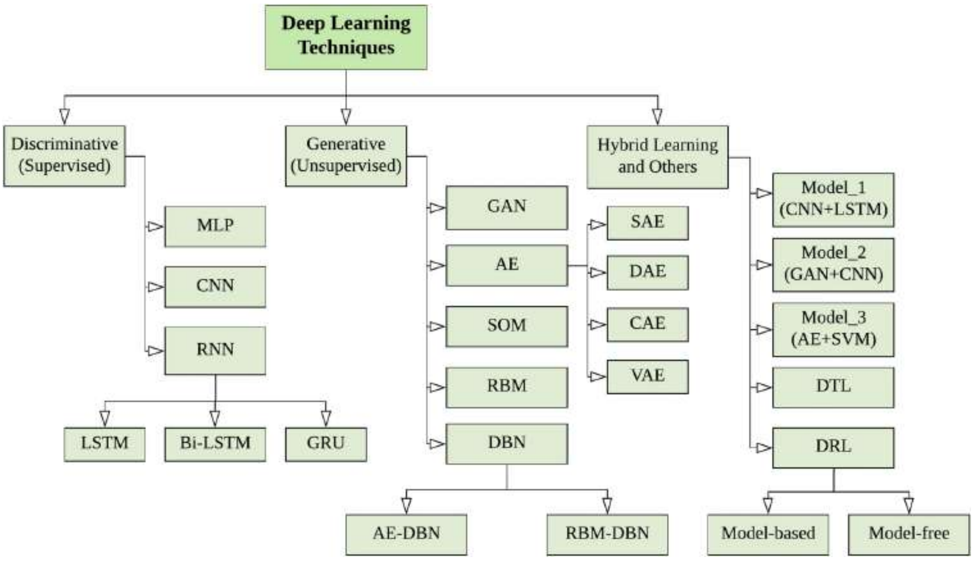

Figure 2: Taxonomy of DL techniques from [10].

A more technique-specific taxonomy is presented in [10] as shown in fig. 2. While those published taxonomy results provide clear guidance of Deep Learning (DL) technique categorization, none of the existing taxonomy can be directly used to model ML tasks, especially for target user groups who have limited knowledge of ML paradigms. We therefore will construct a scalable knowledge system to not only cover the multifaceted ML landscape but also be used to query and construct solution pathways for new application-specific problems.

Therefore, we use the RDF to create a comprehensive and easily expandable knowledge graph database. Compared to existing, technical and data-driven structures, rdf serves the purpose of modeling metadata in form of graph data. Together with the State of the Art (SOTA) Terse RDF Triple Language (Turtle) syntax, simple subject-predicate-object relationships can be defined and stored in the Knowledge Graph database in form of a .ttl file. Each relationship constitutes one connection between two nodes, whereon the whole graph structure is built on. The predicate defines the type of connection and can be further utilized to create hierarchies and flow directions within the graph.

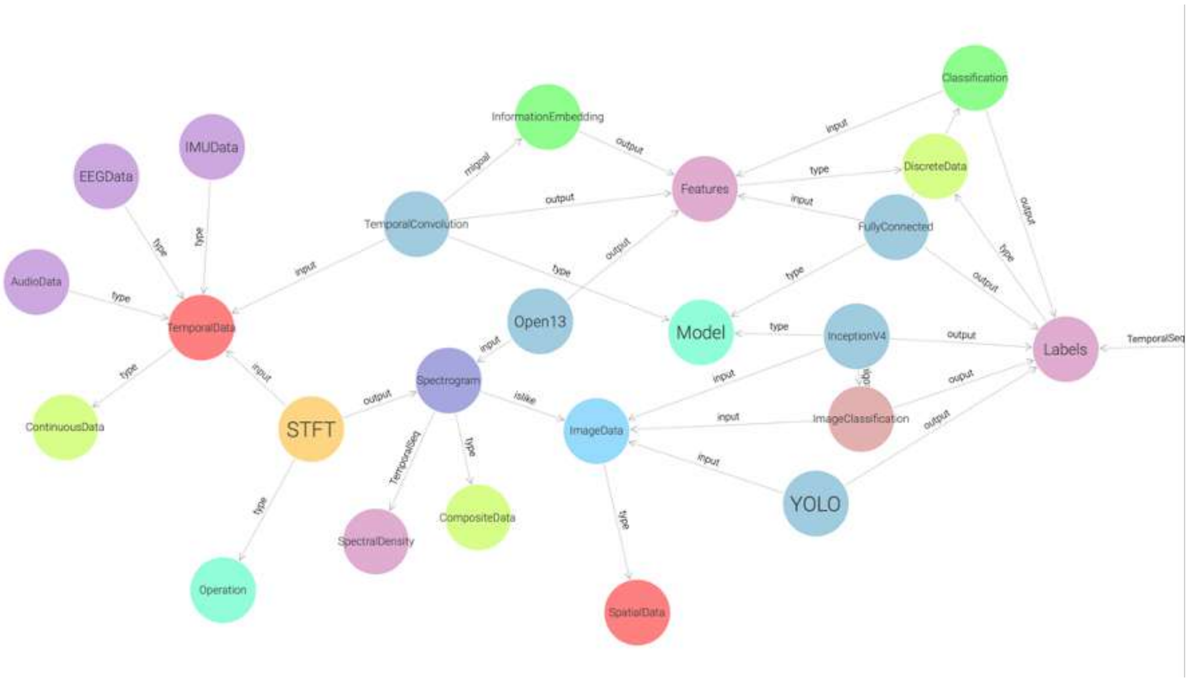

Figure 3: Visualization of part of the knowledge graph being constructed.

To cover the majority of ML problems, we use the existing literature from publications that present a taxonomy of ML problems and metadata from publication abstracts to extend those. After generating a basic graph containing hierarchical problem definitions and goals together with the related algorithms to solve, we gathered further knowledge from ML experts to refine the graph. To do so, various data types and sources together with more refined types of algorithms complete the knowledge graph.

To use the knowledge base in a useful manner, multiple ways are possible to visualize and iterate through the graph due to the generalized RDF syntax. To overall verify the correct creation of the knowledge graph, we utilize the tool Ontotext GraphDB to visually check the nodes and interconnections between them. In fig. 3, a part of the knowledge graph is visualized through GraphDB, following the hierarchical flow of connected nodes. Nevertheless, other commercial tools like Blazegraph or Virtuso may be used as well.

In order to not only visualize the knowledge base but also be able to use it actively, it must be possible to embed the uniform format in the best possible way. Rather, a system must be developed that can search the knowledge base and extract all related and relevant paths. Therefore we build a prototype based on the rdflib python library [11] for querying through RDF structures with help of the SPARQL query language. Unified semantics and syntax are defined through SPARQL to thoroughly inspect the knowledge graph based on search requests. We are embedding this prototype into an appropriate system to support the process of modeling ML tasks from user definitions and to make it more intuitive. As an example, we already have the Interactive Latent Space Explorer, which we are introducing in the following section, that can potentially be extended to serve as a basis.

Such an expert system will then query through the knowledge graph and locate pathways from the input node to the ML goals. There could result in several parallel pathways, which promotes exploring different solutions. The 'islike' keyword also provides knowledge transfer on similar modalities. For example, for a temporal data classification problem as shown in fig. 3, the shortest pathway is TemporalData-

TemporalConvolution-Features-FullyConnected-Labels, which essentially reflects the standard answer given by ChatGPT in table 2. However, there are also other possible pathways like TemporalData-STFT(short time Fourier transform)-Spectrogram-Open13-Features-FullyConnected-Labels. This suggests an alternative solution of using frequency analysis ML models (Open13 in this case) for feature extraction. Furthermore, after the Spectrogram node, there is an 'islike' link to ImageData, and then use image classification models. This type of transfer learning also exists in the domain-expert literature such as [12].

2.3 Interactive heterogeneous multimodal tool

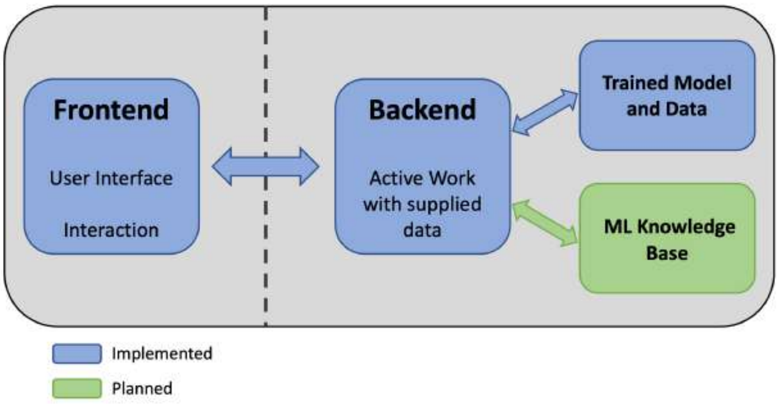

Figure 4: Modularity of the Interactive Latent Space Explorer architecture as an option to embed the ML knowledge base.

A project related to the topic of the ML knowledge base is the Interactive Latent Space Explorer. Based on the approach to start from an already existing and trained model, the Interactive Explorer constitutes an active software instrument allowing deep learning architects to interactively inspect neural network models' output behavior from user-manipulated values in any latent layer. The focus of the tool is on the technical aspects of the ML model, as it mainly analyzes the depths of the individual layers to dissolve the black box behavior of the model's architecture. Together with SOTA dimension reduction techniques, valuable representations of the model's latent space are visualized through interactive plots.

Beyond the scope of latent space, the interactive explorer provides a software foundation where user interactions can trigger ML model codes in the back-end, which can be further developed for other interactions such as linking the back-end to the knowledge base. As outlined in fig. 4, the interactive explorer obtains a common front-end and back-end architecture which is currently used to activate the inspected models in the back end through user interface interactions in the front end. The system implements a modular and versatile approach not only to accept a large variety of custom models for its initial use case, moreover it inherits the possibility to be extended with other use case scenarios like the knowledge base. To realize this, the back end needs to be connected with the knowledge base to perform search queries based on the guidance and interactions within the front end's user interface.

2.4 HW resource aware ML design

A major limitation of existing ML taxonomy or solution generation tools are that they stay on the pure theoretical level and do not consider the actual hardware platform. While at the deployment stage, many ML models will run on a different hardware environment than the training stage for a long period of time until their decommission. These inference hardware like microprocessors often have different (often inferior) algorithmic capabilities than AI training accelerators. For example limited bit precision, multiplier, memory for storing the model, etc. To optimize trained complex models for long time inference on those hardware causes repeated work and creates wasted resources retrospectively during the training phase, as the training phase may be considered over-complicating unpractical solutions. Therefore for generating end-to-end solutions, it is important to consider the HW aspect during the early phases of the AI lifecycle. This can also fully leverage the vision of the SustainML framework as the domain expert would have expectations of the end usecase, which with the help of the framework we can narrow down solution pipelines with HW aware optimization.

To this end we have conducted a usecase implementation with heterogenous sensors and CNN-based sensor fusion human activity recognition (HAR) tasks - FieldHAR. FieldHAR is developed as a scalable register transfer level (RTL) framework as a result of iterative hardware-software co-optimization. FieldHAR aims to address the lack of apparatus to transform complex HAR methodologies often limited to offline evaluation to efficient run-time edge applications. The framework uses parallel sensor interfaces and integer-based multi-branch CNNs to support flexible modality extensions with synchronous sampling at the maximum rate of each sensor. To validate the framework, we used a sensor-rich kitchen scenario HAR application which was demonstrated in a previous offline study. Through resource-aware optimizations, with FieldHAR the entire RTL solution was created from data acquisition to ANN inference taking as low as 25% logic elements and 2% memory bits of a low-end Cyclone IV field programmable gate arrays (FPGA) and less than 1% accuracy loss from the original $FP32$ precision offline study. The RTL implementation also shows advantages over micro-controller (MCU) based solutions, including superior data acquisition performance and virtually eliminating ANN inference bottleneck.

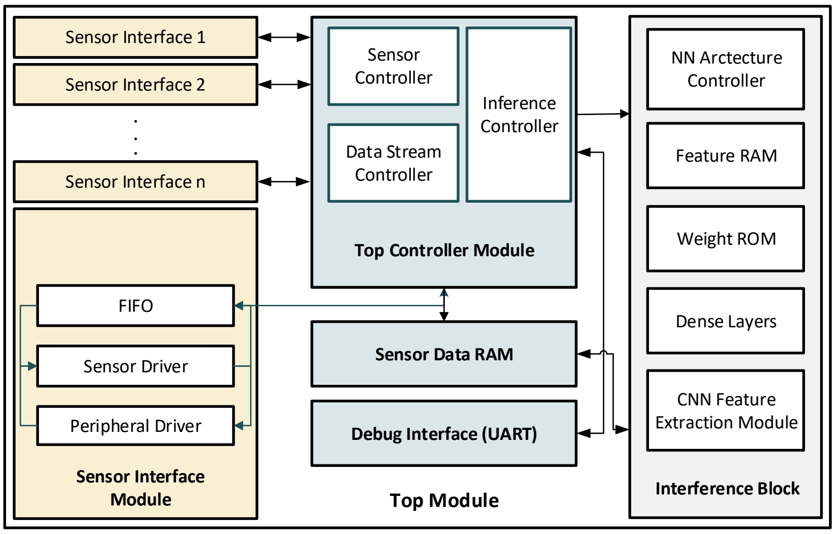

Figure 5: The FieldHAR RTL framework helping us explore the hardware constraints through iterative hardware-software co-optimization in a heterogeneous modality setting.

3 Future works in the following period

3.1 Plans for WP1

Referring again to figure 1, this deliverable mainly considers the first stage of the ML lifecycle to conceptualize a model from an abstract human description into a unified machine description. Our RDF-based knowledge base covers the ML landscape with a taxonomy from meta information like goals and problems to highly technical properties like datatypes, algorithms, and DL techniques. For the future, there is not only the need to further extend this knowledge base with more specific details, moreover, but we also aim to include it in more sophisticated systems that actually benefit from this large set of information. To ensure a sufficient user experience in this phase of conceptualizing the ML project, we will integrate the knowledge graph search into a user interface-based system to interactively explore the knowledge base.

Targeting the second state of the ML lifecycle, we are moving from a purely textual and graph-based representation of knowledge to an even more connected representation by linking search results of the graph to code examples from already existing projects. In terms of sustainability, we cannot only reduce the users' time and effort in conceptualizing the ML problem but more importantly reduce the carbon footprint when recycling already existing projects that are identical or at least similar to the user's problem statement. From there on, the process of gathering code from user descriptions can be further used by the other project partners working on optimizing the following stages of the ML lifecycle.

We will also work on integrating models hosted on HuggingFace with our knowledge database, so that curated models on HuggingFace will have corresponding nodes in the database which can be integrated in querried solutions.

3.2 Joint work with other WPs

The collaboration with WP2, WP3, and WP4 will further integrate the knowledge base into the ML lifecycle. For WP2, we target a better understanding of HW constraints and active optimization for better HW utilization. For WP3, we aim to measure the energy savings potential of the knowledge base. For WP4, a more extended user study will help in extending the knowledge base.

The following list states the detailed plans of the joint work with other WPs:

Joint with WP2:

1. Extend FieldHAR with the consortium HW expertiese
2. Pruning and quantization, especially quantization-aware training
3. Edge training and retraining for prolonged lifecycle with field data
4. Establish FPGA-based HW constraint including both NN accelerators and neuromorphic implementations. With such constraints, the SustainML expert system could provide more customized optimization if the inference devices are specified by the users.

Joint with WP3:

1. Literature review for ML model optimizations with a strong focus on sustainability and carbon footprint
2. Combination of Carbon Tracker [13] and the knowledge base to predict carbon footprint and possible savings throughout the ML lifecycle

Joint with WP4:

1. HCI-oriented domain expert user studies to validate and refine the knowledge base

References

[1] Rabah A Al-Zaidy and C Lee Giles. “Extracting semantic relations for scholarly knowledge base construction”. In: *2018 IEEE 12th international conference on semantic computing (ICSC)*. IEEE. 2018, pp. 56–63.

[2] Batta Mahesh. “Machine learning algorithms-a review”. In: *International Journal of Science and Research (IJSR).[Internet]* 9 (2020), pp. 381–386.

[3] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. “Machine learning from theory to algorithms: an overview”. In: *Journal of physics: conference series*. Vol. 1142. IOP Publishing. 2018, p. 012012.

[4] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. “Multimodal machine learning: A survey and taxonomy”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.2 (2018), pp. 423–443.

[5] Šejla Čebirić et al. “Summarizing semantic graphs: a survey”. In: *The VLDB journal* 28 (2019), pp. 295–327.

[6] Fabio Petroni et al. “Language models as knowledge bases?” In: *arXiv preprint arXiv:1909.01066* (2019).

[7] Ziwei Ji et al. “Survey of hallucination in natural language generation”. In: *ACM Computing Surveys* (2022).

[8] Sébastien Bubeck et al. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. 2023. arXiv: `2303.12712 [cs.CL]`.

[9] Frank Emmert-Streib and Matthias Dehmer. “Taxonomy of machine learning paradigms: A data-centric perspective”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 12.5 (2022), e1470.

[10] Iqbal H Sarker. “Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions”. In: *SN Computer Science* 2.6 (2021), p. 420.

[11] Carl Boettiger. *rdflib: A high level wrapper around the redland package for common rdf applications*. Zenodo, 2018. DOI: `10.5281/zenodo.1098478`. URL: `https://doi.org/10.5281/zenodo.1098478`.

[12] Kevin William Gunawan et al. “A transfer learning strategy for owl sound classification by using image classification model with audio spectrogram”. In: *international Journal on Electrical Engineering and informatics* 13.3 (2021), pp. 546–553.

[13] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. *Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models*. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems. arXiv:2007.03051. July 2020.