SustainML

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

Task modeling from user definition results - Final

Deliverable D1.2

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

eProsima
The Middleware Experts

DFKI

Inria

upmem

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

IBM

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:	**D1.2**
Title:	**Task modeling from user definition results - Final**
Month:	24
Work Package:	WP1 - Energy Consumption Oriented ML Task Modeling
Work Package leader:	DFKI
Deliverable leader:	DFKI
Deliverable type:	R
Dissemination level:	PU
Date of submission:	2024-10-15
Version:	v1.1
Status:	Finished

Version history

Version	Date	Responsible	Author/Reviewer	Comments
v1.0	15-10-2024	DFKI	DFKI	
v1.1	15-10-2024	EPROS	EPROS	Review

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 D1.2 - Task modeling from user definition results - Final* 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.

Several components of our work have already been published within 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 accepted for publication in the International Joint Conference on Artificial 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 published on the IEEE International Conference on Application Specific Systems, Architectures and Processors (ASAP https://www.asap2023.org)
4. The Power of Training: How Different Neural Network Setups Influence the Energy Demand has been published on the 37th GI/ITG International Conference on Architecture of Computing Systems (ARCS https://arcs-conference.org/home)
5. TSAK: Two-Stage Semantic-Aware Knowledge Distillation for Efficient Wearable Modality and Model Optimization in Manufacturing Lines has been accepted at International COnference on Pattern Recognition 2024 (ICPR https://icpr2024.org/)

During the submission of this deliverable, the following works are submitted to international academic venues, but still within the reviewing process:

1. Talk2AIoD: Investigating an Adapted Retrieval-Augmented Generation Approach Towards Sustainable Web Agents has been submitted to CHI 2025 (https://chi2025.acm.org/)
2. Latent Boost: Leveraging Latent Space Distance Metrics to Augment Classification Performance has been submitted to the Thirteenth International Conference on Learning Representations (ICLR 2025 https://iclr.cc/Conferences/2025)
3. CoSS: Co-optimizing Sensor and Sampling Rate for Data-Efficient AI in Human Activity Recognition has been submitted to ICASSP 2025 (https://2025.ieeeicassp.org/)

Contents

Executive Summary **3**

Contents **4**

Acronyms **5**

1 Introduction **6**
1.1 Related literature 6
1.2 Related toolkits 7

2 Progresses of WP1 **7**
2.1 Overall methodology 7
2.2 ML task modeling knowledge base 7
2.3 Interactive heterogeneous multimodal tool 11
2.4 HW resource aware ML design 11
2.5 Embedding Space Exploration 12
2.6 Knowledge Beyond Black Box Optimization 14
2.7 Retrieval Augmented Generation for Sustainable Knowledge Utilization 15

3 Conclusion **17**

References **18**

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.
RAG	Retrieval Augmented Generation.
RDF	Resource Description Framework.
RNN	recurrent neural networks.
RTL	register transfer level.
SOTA	State of the Art.

Progress Since Last Report (M6)

Continuing the work from the previous report, we conducted research in multiple directions to thoroughly analyze the benefits and proper utilization of knowledge within the ML development cycle. As a main contribution of this deliverable, we evaluated how knowledge from model embedding spaces can be visualized and elevated to enhance the traditional black box setting of ML models. Advanced approaches in this direction can help to extend the knowledge base of the framework to prevent resource waste due to restricted or missing knowledge as early in the development pipeline as possible. Addressing the trends of LLMs, we added research towards Retrieval Augmented Generation (RAG), a novel approach to elevate the knowledge of Foundational LLMs with user-specified content from external sources. The further implementation of the work package regarding the knowledge base covering the landscape of ML, including samples of data structures and code, is presented in the following D1.2. Especially, we are addressing the continuous extension and retrieval of information within the targeted SustainML frameworks knowledge base from classic interaction modalities to advanced LLM supported RAG systems.

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 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

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

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

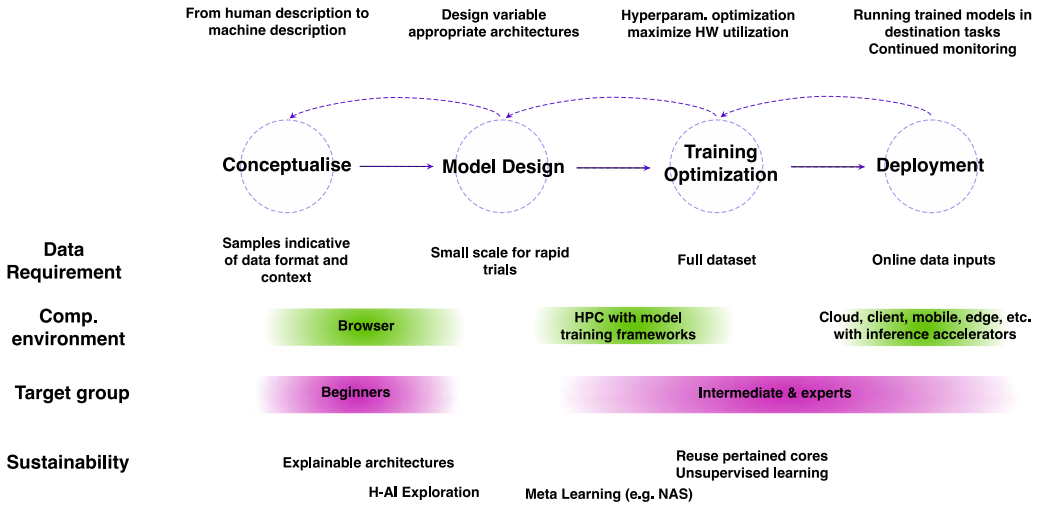

Figure 1: Development stages throughout the entire life cycle of ML model from initial conceptualisation phase to final deployment.

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.

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.

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

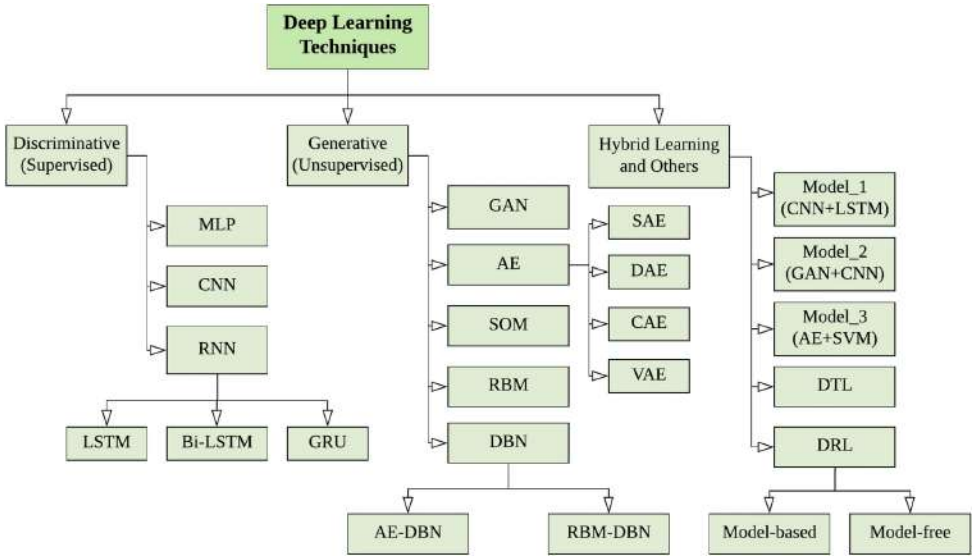

Figure 2: Taxonomy of DL techniques from [10].

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].

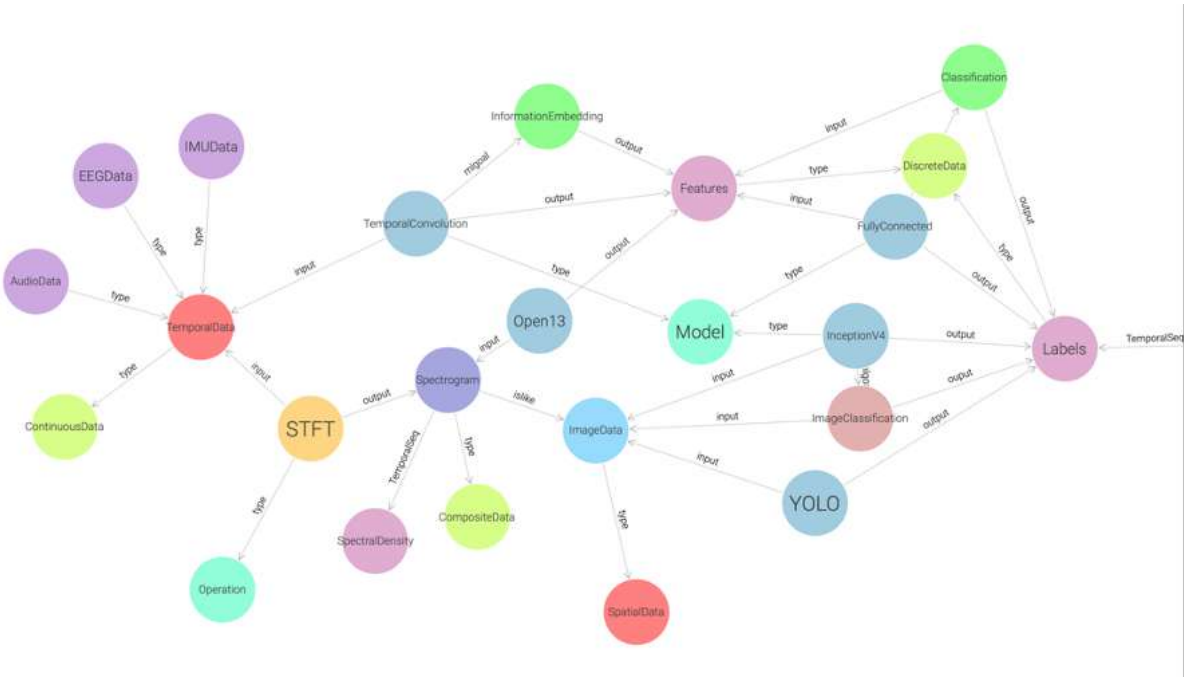

Figure 3: Visualization of part of the knowledge graph being constructed.

2.3 Interactive heterogeneous multimodal tool

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,

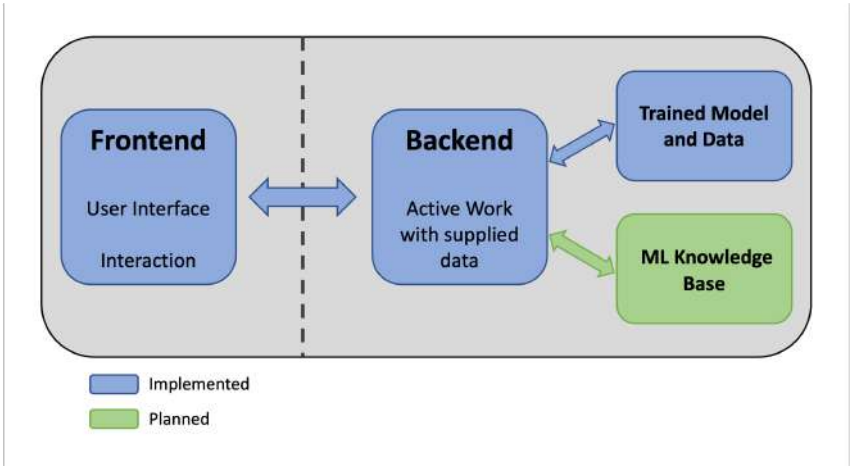

Figure 4: Modularity of the Interactive Latent Space Explorer architecture as an option to embed the ML knowledge base.

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.

2.5 Embedding Space Exploration

The increasing demand for high-performance machine learning models, especially in large-scale applications, has highlighted a critical issue: the inefficiency of training processes in terms of both, training time and computational resources. As models grow in size and complexity, achieving optimal convergence while ensuring high-quality latent space representations becomes more challenging. Traditional training approaches often lack real-time oversight, which can lead to inefficiencies, such as slow convergence, poorly organized latent spaces, and suboptimal class separation—issues that might only become apparent in later training stages. This not only increases the computational load but also exacerbates environmental costs due to the higher energy consumption required for prolonged training.

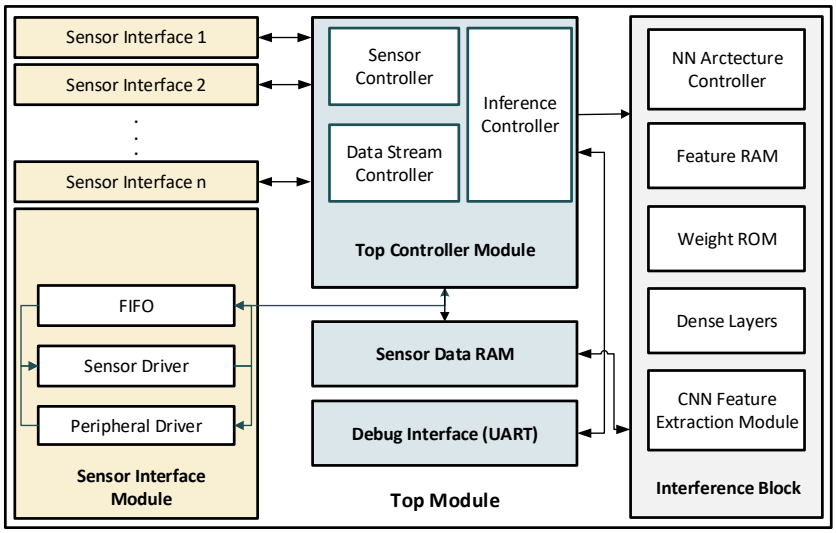

Figure 5: The FieldHAR RTL framework helping us explore the hardware constraints through iterative hardware-software co-optimization in a heterogeneous modality setting.

A human-in-the-loop (HITL) approach addresses this challenge by introducing an interactive inspection from latent layers in the training pipeline, enabling real-time adjustments to be made early in the training process. By allowing users to engage directly with the latent space representations of the model during training, it becomes possible to identify and correct issues such as poor class separation or non-coherent clustering at an early stage. This improves the overall training efficiency, optimizes convergence rates, and reduces unnecessary computational overhead. From a sustainability perspective, this approach can significantly minimize the carbon footprint of machine learning models by streamlining the training process and reducing the need for extended training cycles.

In Figure 6, we see an example of how this HITL approach is applied. The scatter plot as an example depicts the latent space representation of the test dataset being past till the inspected layer as extracted from a model during training. By presenting these latent spaces in an interactive, reduced-dimensional form (using various techniques t-SNE or PCA), users can observe how the model is organizing the data internally. This offers a unique opportunity to intervene in real time: users can spot misaligned or overlapping class clusters, make adjustments, and guide the latent space organization to improve class separation.

Given that, this projection is a dimensionality-reduced view of a much more complex space, users are constrained in their ability to rescale the data into its original dimensions. To address this, the HITL framework leverages distance metrics to quantify and calculate the effect of user modifications on the latent space. As the user adjusts the positions of points in the plot, bringing similar data points closer or pushing dissimilar points apart—distance metrics are computed to capture the degree and nature of these changes.

Finally, to ensure these user-driven refinements are effectively incorporated into the actual training process, we introduce a knowledge distillation technique. In this scenario, the modified latent space becomes the teacher, guiding the student, which is the actual model being trained. The model adjusts its learning to reflect the changes introduced by the user, thus ensuring that the latent space evolves in alignment with both the data and the user's insights. This process not only improves the efficiency of model training by targeting a more optimal latent space but also enhances the model's final performance, as the modifications are directly informed by human intuition and domain expertise.

In essence, this HITL framework offers a powerful solution for improving sustainability in machine learning. By introducing early intervention and optimizing the latent space, it accelerates convergence, reduces resource consumption, and ensures that models achieve higher quality in a shorter timeframe.

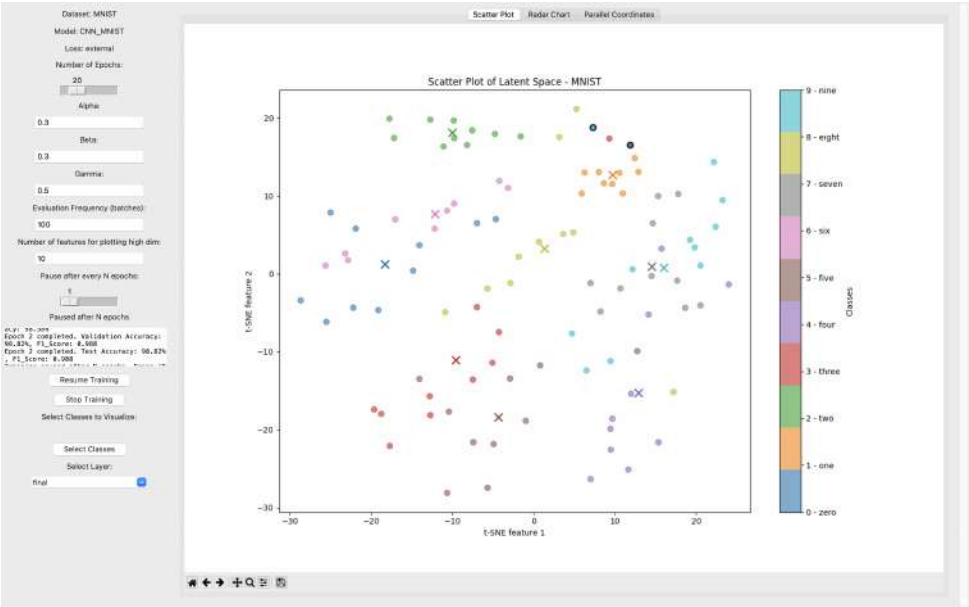

Figure 6

2.6 Knowledge Beyond Black Box Optimization

In traditional ML approaches, training typically optimizes models by minimizing a probabilistic loss function based on discrete data points. However, this method neglects the rich structural information embedded within the model's latent representations, which can be a critical source of knowledge. The primary focus of our work is on utilizing latent space knowledge during machine learning training to significantly improve classification performance. The so called Latent Boost work aims to explicitly leverage this latent knowledge to guide the learning process.

The latent space in a neural network holds complex, high-dimensional representations of input data, capturing the relationships between different classes and features. However, these representations are often left underutilized, with conventional training methods treating them as a byproduct rather than an essential component of the learning process. The key innovation of Latent Boost is to incorporate distance metrics derived from this latent space into the training itself. By doing so, the model is not only optimized to classify individual data points but is also driven to organize its internal latent space in a way that reflects the true relationships between data points.

Latent Boost leverages this latent knowledge by enforcing a structured clustering of the data in the hidden layers of the model. This means that semantically similar data points are drawn closer together in latent space, while dissimilar points are pushed further apart. The advantage of this approach is that it allows the model to utilize the latent knowledge of how data points are distributed and related to each other, improving both classification accuracy and the overall generalization of the model.

Figure 7 illustrates how Latent Boost modifies the training process by introducing latent space distance metrics into the loss function. By balancing probabilistic loss with distance-based loss, the method forces the model to pay attention not just to the individual data points, but also to the structure of the latent

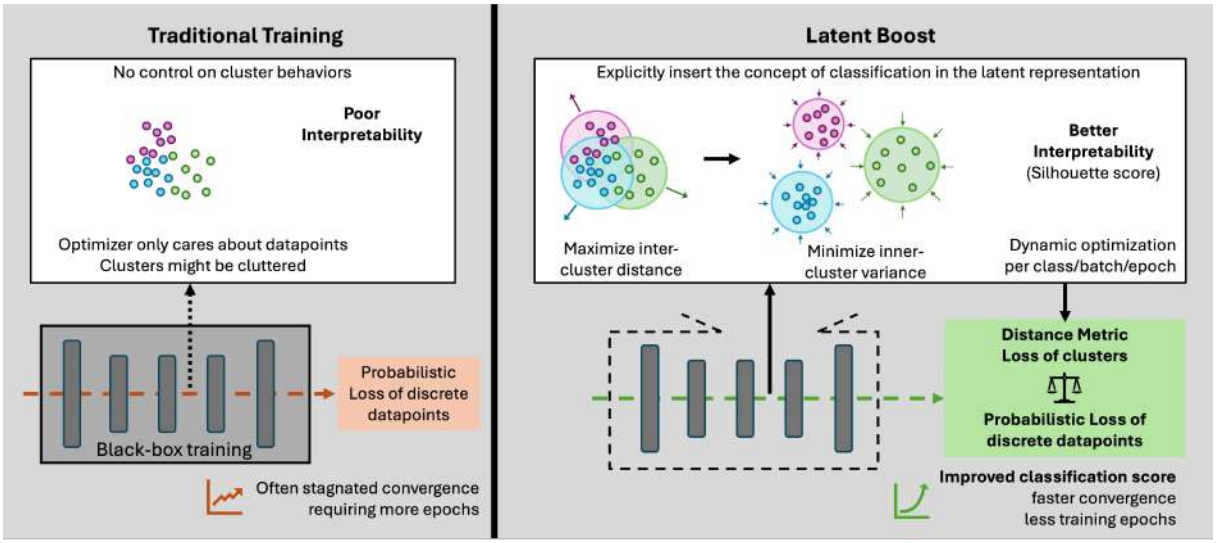

Figure 7: Oppose to traditional training, relying on probabilistic loss only, Latent Boost injects distance metric information, obtained from the model's hidden latent representations, as addition into the training through balanced weighted sum equations.

space. This utilization of latent knowledge ensures that the classes are well-separated and internally consistent in the hidden layers, providing a more robust framework for classification.

This approach enables more efficient learning as well. By utilizing latent space knowledge, Latent Boost accelerates convergence. The structured latent representations provide the model with clearer distinctions between classes, leading to faster learning and fewer required training epochs. This translates into better utilization of computational resources and faster overall training time, with the added benefit of improved interpretability. The clustering within the latent space provides a clearer view of how the model organizes its understanding of the data, offering insights into its decision-making process.

In summary, Latent Boost represents a shift from treating the latent space as incidental to actively utilizing its embedded knowledge to enhance the training process. Evalauted on multiple sota datasets coming from the vision community, Latent Boost proofed that extended knowledge about the training can significantly improve the convergence quality and speed. By integrating latent space distance metrics into the optimization process, the method ensures that the model is guided not only by the discrete data but also by the structure and relationships within its internal representations. This approach leads to better performance, faster convergence, and improved interpretability, all through more effective utilization of the latent knowledge present during training. We envision to add such approaches into the SustainML framework to use the best possible knowledge as early as possible in the development cycle to reduce the resulting energy footprint.

2.7 Retrieval Augmented Generation for Sustainable Knowledge Utilization

LLM based agents are able to crawl entire web pages during query execution, hence such querying further increase the computational demand as an addition to the already high demand from LLM inference. This approach, though capable of providing real-time and up-to-date information, incurs significant token costs and environmental impact when scaled.

The work of Talk2AIoD proposes an alternative approach that leverages RAG along with a vector database for efficient web search. The key idea is to store an intermediate vector-based representation of web

pages, allowing the system to retrieve only the most relevant information, reducing the need for full web crawls. This approach utilizes latent knowledge, extracted from previously crawled content, stored in a pre-generated vector database. By integrating this knowledge into the training and inference process, Talk2AIoD reduces the number of tokens processed per query by over 90% compared to state-of-the-art web agents.

The system relies heavily on latent knowledge in the form of vector embeddings that represent website content. Instead of adding the entire webpage's HTML to the LLM's context (which increases token load), the agent retrieves and queries the vector database for the information relevant to the user's query. This reduces computational overhead by avoiding repeated crawling and minimizes the number of tokens needed to process the content. The vector embeddings act as condensed forms of latent knowledge, allowing the system to effectively pinpoint and retrieve only the most relevant portions of the data.

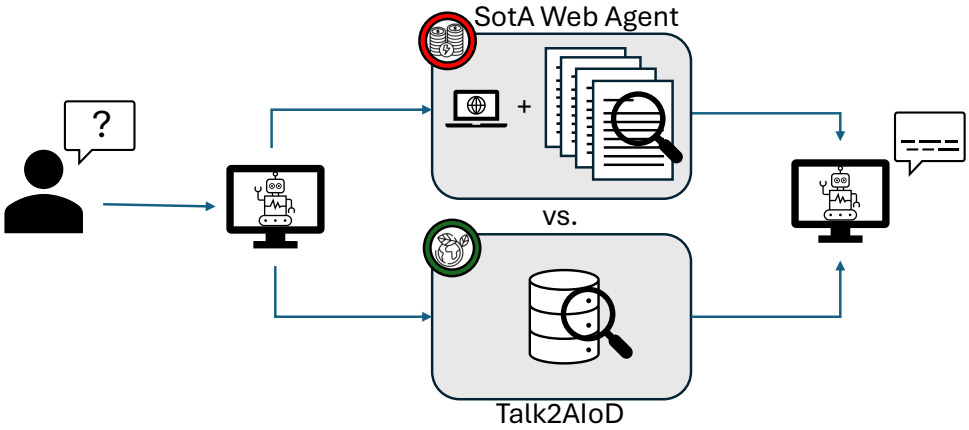

Figure 8: Talk2AIoD is an agent that uses significantly less computational resources when queried by the user. Instead of crawling multiple webpages (top row), it relies on a pre-generated vector database (bottom row) for efficient query execution.

Figure 8 presents this by contrasting the traditional web agent architecture, which requires crawling multiple pages and adding HTML to the LLM context, with Talk2AIoD's vector-based system, where web content is already stored in a vector database, enabling quicker and more efficient query resolution. This efficient utilization of pre-stored latent knowledge allows the model to handle complex queries with far fewer tokens and computational resources, contributing to the sustainability of web agents.

By leveraging latent knowledge embedded in the vector database, Talk2AIoD demonstrates how structured information from prior web crawls can be efficiently re-used during subsequent queries. This allows for faster and more sustainable query processing without the need to continuously retrieve large amounts of data from the web in real-time, thereby making web-based LLMs more scalable and energy-efficient. In the evaluation, Talk2AIoD significantly reduced token usage, requiring only 9.7% of the tokens needed by state-of-the-art web agents, leading to an 8.8x reduction in CO2 emissions. Additionally, user studies showed that Talk2AIoD improved task completion times and correctness by 20%, demonstrating both computational efficiency and enhanced usabilityIn the evaluation, Talk2AIoD significantly reduced token usage, requiring only 9.7% of the tokens needed by state-of-the-art web agents, leading to an 8.8x reduction in CO2 emissions. Additionally, user studies showed that Talk2AIoD improved task completion times and correctness by 20%, demonstrating both computational efficiency and enhanced usability.

In summary, the core motivation behind Talk2AIoD is to shift away from costly real-time web crawling towards a more sustainable, knowledge-centric approach. By utilizing latent space knowledge stored in a vector database, the system not only reduces computational and environmental costs but also maintains

high levels of accuracy and usability in web search. This marks a significant advancement in how LLMs can be trained and optimized for efficient information retrieval. For the SustainML framework, we aim to beenift form such approach. Instead of utilizing vector database as the foundation, we want to add a Graph-RAG approach as a basis to retrieve knowledge from the knowledge graph through LLM supported interaction. This way we aim to address non-experts to use the framework in order to support them with proper model selections independently of their skill level.

3 Conclusion

In this deliverable, addressing tasks T1.1, T1.2, and T1.3 from WP1, the primary research focused on the role of knowledge throughout the machine learning pipeline, as outlined in Figure 1. With particular emphasis on the initial three iterations of the pipeline, our investigation explored how knowledge can be systematically captured and propagated to improve the overall efficiency of ML model development.

Deliverable D1.3 will detail the implementation of our knowledge base and introduce the code structure, which forms the core of the SustainML framework. Looking ahead, the knowledge base will not only be expanded with more domain-specific information but also integrate into more advanced systems that can fully leverage this wealth of information. Additionally, we plan to incorporate models from platforms such as HuggingFace into the knowledge base, enabling curated models from HuggingFace to be linked with corresponding nodes in the database, thereby facilitating more effective and informed solution queries.

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.