



D3.1 Life cycle management framework for machine learning models

Dissemination level	Public (PU)
Work package	WP3
Task:	T3.1
Deliverable lead:	VICOMTECH
Version	V3.0
Submission date	31/10/2023
Due date	31/10/2023

Authors

Authors in alphabetical order		
Name	Organisation	Email
Alexandru Forrai	SIEMENS-NL	alexandru.forrai@siemens.com
Guido Küppers	IKA	Guido.kueppers@ika.rwth-aachen.de
Ilma Okanovic	VIF	Ilma.okanovic@v2c2.at
Jos den Ouden	TU/e	j.h.v.d.ouden@tue.nl
Paola Cañas	VICOM	pncanas@vicomtech.org
Raul Ferreira	CONT	raul.ferreira@conti-engineering.com
Till Beemelmans	IKA	till.beemelmans@ika.rwth-aachen.de

Control sheet

Version history			
Version	Date	Modified by	Summary of changes
1.0	04/07/2023	VICOM-Paola	ToC
2.0	09/10/2023	All authors	Contributions from all partners and formatting
2.1	12/10/2023	Jos den Ouden, TU/e	Introduction, reordering of entire text, conclusions
2.2	16/10/2023	Paola -VICOM	References and formatting
3.0	30/10/2023	All authors	Final changes after reviews and formatting

Peer review		
	Reviewer name	Date
Reviewer 1	Bonnie Fenton (RC)	20/10/2023
Reviewer 2	Justyna Beckmann (FIA)	20/10/2023

Disclaimer



Funded by
the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

Table of contents

Table of contents	3
1. Introduction	5
1.1. Althena concept and approach	5
1.2. Purpose of this deliverable.....	5
1.3. Trustworthiness in ML models	6
1.4. Structure of this deliverable.....	6
2. Life cycle management framework for ML models.....	7
2.1. ML model life cycle	7
2.1.1. Data.....	7
2.1.1.1. Data sources (data collection)	7
2.1.1.2. Data exploration.....	8
2.1.1.3. Data versioning	8
2.1.1.4. Data pre-processing.....	9
2.1.1.5. Labelling/annotation of data	9
2.1.2. Training.....	9
2.1.2.1. ML software framework.....	9
2.1.2.2. Repository	10
2.1.2.3. Model versioning	10
2.1.2.4. Training monitoring	10
2.1.3. Testing.....	10
2.1.3.1. Continuous integration and continuous delivery (CI/CD).....	10
2.1.3.2. Model serialization.....	10
2.1.4. Deployment	10
2.1.4.1. API.....	11
2.1.4.2. Model monitoring	11
2.1.4.3. Model optimization.....	11
2.2. Operational environment	11
2.2.1. Storage	11
2.2.1.1. AI-focused data storage	11
2.2.1.2. Feature storage.....	12
2.2.1.3. Model storage	12

2.2.1.4.	DPU.....	12
2.2.2.	Computing	12
2.2.3.	Virtualization.....	12
2.2.4.	Orchestration	12
2.3.	Tools for ML management framework.....	13
2.3.1.	Data.....	13
2.3.2.	Training.....	13
2.3.3.	Testing.....	13
2.3.4.	Deployment	13
2.3.5.	General	14
2.4.	Diagram	14
3.	Robustness assessment using ML tools	15
4.	Model Cards.....	18
4.1.	Definition	18
4.2.	Proposed Model Card.....	20
5.	Description of ML frameworks	25
5.1.	ML model #1	25
5.1.1.	Model #1 life cycle diagram.....	25
5.1.2.	Model Card.....	26
5.2.	ML model #2	26
5.2.1.	Model #2 life cycle diagram	27
5.2.2.	Model Card.....	27
6.	Conclusions.....	27
	Bibliography.....	29
	List of acronyms and terms	31
	Appendix A – Model Card Model #1.....	32
	Appendix B – Model Card Model #2.....	36

1. Introduction

1.1. Athena concept and approach

Connected, Cooperative and Automated Mobility (CCAM) solutions have emerged thanks to novel Artificial Intelligence (AI) which can be trained with huge amounts of data to produce driving functions with better-than-human performance under certain conditions. The race on AI keeps on building Hardware (HW) and software (SW) frameworks to manage and process even larger real and synthetic datasets to train increasingly accurate AI models. However, AI remains largely unexplored with respect to explainability (interpretability of model functioning), privacy preservation (exposure of sensitive data), ethics (bias and wanted/unwanted behaviour), and accountability (responsibilities of AI outputs). These features will establish the basis of trustworthy AI, as a novel paradigm to fully understand and trust AI in operation, while using it at its full capabilities for the benefit of society. Athena will contribute to build Explainable AI (XAI) in CCAM development and testing frameworks, researching three main AI pillars: data (real/synthetic data management), models (data fusion, hybrid AI approaches), and testing (physical/virtual X- in the loop (XiL) set-ups with scalable Machine Learning Operations (MLOps)). A human-centric methodology will be created to derive trustworthy AI dimensions from user identified group needs in CCAM applications. Athena will innovate proposing a set of Key Performance Indicators (KPI) on XAI, and an analysis to explore trade-offs between these dimensions. Demonstrators will show the Athena methodology in four critical use cases: perception (what does the AI perceive, and why), situational awareness (what is the AI understanding about the current driving environment, including the driver state), decision (why a certain decision is taken), and traffic management (how transport-level applications interoperate with AI-enabled systems operating at vehicle level). Created data and tools will be made available via European data sharing initiatives (OpenData and OpenTools) to foster research on trustworthy AI for CCAM.

1.2. Purpose of this deliverable

This deliverable D3.1 is the first output of WP3 (model development), related to T3.1. This task focuses on specification, design, and development of life-cycle management framework for machine learning (ML) algorithms. As the first output of Task 3.1, this deliverable aims to **provide an overview of ML frameworks** from literature and from the use of ML tools by the consortium, and it **shows a first implementation of the use of ML model cards** on several different use cases, which can later be used as a template to automated traceability of code, datasets and artifacts. Using the model cards, the developed AI algorithms can be explained, made more accountable and ensure robustness. Later in the project, the results of Task 3.1 will be embedded in D3.2 and D3.3.

With this overview, this deliverable provides input for two of Athena's general objectives:

1. GO-2: XAI models – research and development of explainable AI models.
2. GO-8: Testing and validation – development of testing and validation methodologies for AI-based CCAM applications.

As one of the first deliverables of Athena, and part of the AI development work package, this deliverable mainly focuses on providing a first draft of tools to AI developers that can help make their AI algorithms more robust, explainable, accountable, and trustworthy. The project has another deliverable, D1.1, *Methodology for Evaluating the Ethics, Transparency, Accountability, Privacy, Accuracy and Robustness of AI-based Systems in CCAM Applications*, which focuses more on the strategies that could be taken to

make AI-based CCAM more human centric (broader policy point of view), while D3.1 focuses more on the technical / engineering point of view.

This deliverable presents the methodology and tools for the development and life-cycle assessment of individual ML algorithms. The methodology is applied to the development of ML algorithms for the use cases of the project and is based on Althena's human-centric approach, which is described in detail in D1.1.

1.3. Trustworthiness in ML models

This deliverable builds upon the work done in WP1. According to the ethics guidelines for trustworthy AI (AI, 2019), which will also be further detailed and analysed in D1.1, the development, deployment, and use of AI systems should meet seven key requirements for trustworthy AI:

- human agency and oversight,
- technical robustness and safety,
- privacy and data governance,
- transparency,
- diversity, non-discrimination, and fairness,
- environmental and societal well-being and
- accountability.

D1.1 provides checklists and guidance for use by regulators and those considering implementing a CCAM system within their jurisdiction as well as high level strategies and tools for developers and testers. This deliverable, by contrast, provides some more **concrete tools for developers and testers**, that they can use to make their models adhere to the seven requirements.

1.4. Structure of this deliverable

This deliverable is structured as follows:

- Chapter 2 presents the life-cycle management framework for ML models, giving a pipeline of a typical AI development process, based on literature to be used within the Althena project. It also presents the tools used for the ML framework, giving an overview of the variety of tools available currently.
- Chapter 3 presents a robustness assessment using the life-cycle management framework and some of the tools, to provide a concrete example of how Althena is currently employing this framework.
- Chapter 4 presents the model cards applied to several of the Althena use cases, which can provide an additional tool to provide trustworthy AI during development and deployment.
- Chapter 5 presents examples of how to use the model cards in two Althena relevant use cases.

2. Life cycle management framework for ML models

2.1. ML model life cycle

Every Machine Learning (ML) model must go through a learning process, which, as can be expected, is a key phase in its development. However, there are other equally important phases such as the preparation of data for training, the gathering of results to evaluate the performance of the model, and the actual implementation of the model in a real environment. These processes result in **four main stages: Data, Training, Testing, and Deployment**.

The execution of these main stages is usually done one by one in a manual process. However, automating these tasks, not only for the execution of one stage but for all stages in a consecutive way, can result in a faster and more efficient development of a model. This is because during the development, many experiments and changes must be performed to get to a final version. Once an automated pipeline is created, these changes can be applied more quickly, and the developer can obtain results sooner. This is related to MLOps (Machine Learning Operations), which is the practice of applying DevOps (Development Operations) principles that aim to bring together the development, testing, and operational aspects of software development. It focuses on collaboration between developers, operations, and data science to shorten a system's development life cycle and provide continuous delivery with high software quality (Microsoft, s.f.). This is applied to machine learning workflows to improve the efficiency and effectiveness of ML model development and deployment.

In this section, we will discuss each of these stages in detail and explore the subtasks that can be performed during each stage to ensure the successful development and deployment of an ML model. Tools that can support each process described here, are presented in 2.3 Tools for ML management framework, their references are included as a separated list on the Bibliography section.

2.1.1. Data

This first phase covers all actions performed around data, from collecting it, to processing it and annotating it. The quality and quantity of the data directly impact the performance of the model. It should be representative of the problem intended to be modelled by the ML algorithm, including samples of all possible situations, cases, etc. This is also important from an ethical perspective, when constructing algorithms that will function on people or will affect people, it is important that each person is represented on the dataset, otherwise the model's predictions could be biased, leading to unfair outcomes. On the other hand, having redundancy in the data will only affect the learning process of the model and consume more resources, so it is important to avoid it.

Among the processes that can be done in this stage, there is:

2.1.1.1. Data sources (data collection)

With the rise of big data and Internet of Things (IoT) technologies, it has become easier to collect data in some fields of study. Common sources include:

- Public datasets or data pools. Many of these are available on the internet and are free to use for machine learning projects. They can be found on platforms like [Kaggle](#), [UCI Machine Learning Repository](#), etc.

- Building your own dataset is also a possibility by recording the data that is relevant for the model. Depending on the sensor chosen, there could be an API for recording.
- Another way is to access data streaming; for example, from an IoT device that continuously generates data about its operation and the environment it is in. [Apache Kafka](#) is a popular tool for working with real-time data streams.
- If none of the previous alternatives fit the data requirements, there is the option to create synthetic data. This is artificially generated and is often used in scenarios where collecting real-world data is challenging for reasons such as privacy concerns, rarity of events, or cost. For a CCAM context, there are some simulators that allows configuration of traffic scenarios, different vehicles behaviours, etc. Some of the simulators are [CARLA simulator](#), [SVL Simulator](#), [CarSim](#), [Simcenter PreScan](#), etc.
- Finally, data augmentation is a technique widely used to increase the ability of the ML model to generalize; this involves creating transformed versions of images. Transformations include rotations, translations, zooming, flipping, etc., making the model invariant to scale and orientation change. Built-in functions from deep learning frameworks like [Keras](#), [Pytorch](#), etc., can be used for image data augmentation.

2.1.1.2. Data exploration

Getting to know the data can be crucial for the machine learning pipeline. It involves understanding the characteristics of the data to help identifying the best approaches for preprocessing or building the ML algorithm.

- Statistics of the datasets can be extracted to know the dispersion of the data by calculating the mean, median, mode, standard deviation, etc., for each feature in the dataset.
- In classification problems, it's important to check if the classes are balanced or imbalanced. Imbalanced datasets might require special techniques like resampling or using specific evaluation metrics.
- Graphical representations like histograms, box plots, scatter plots, etc., can help understand the distribution of data and the relationships between features. Also, the visualization of random samples of data (e.g., Images) can help having a broader idea of how your data is or looks like.
- Along with the last item, it is important to study the correlation between features. This can identify redundancy.

Frameworks like [Keras](#) and [Pytorch](#) support with some of these functions, otherwise, they can be manually programmed. Other tools offer this data exploration once you upload the data.

2.1.1.3. Data versioning

Since data is so important in any ML development, many experiments with data variations can be performed. These variations can be different distribution in each split (training and test split), not working with all the classes of the dataset but only with a few, etc. Therefore, it is convenient to have implemented data versioning techniques to keep track of all the data for training changes and able to reproduce some results or go back after some transformations.

There are several tools that can assist with data versioning like [DAGsHub](#) and [DVC](#) (Data Version Control)

2.1.1.4. Data pre-processing

Depending on the nature of the data and the specific requirements of the machine learning algorithm to be used, various preprocessing steps might be applied. These can include data cleaning where missing values or outliers are handled, any data transformation that is required to create data for training or encoding categorical variables. All these processes can be done integrating functions from Python libraries like Pandas, NumPy.

2.1.1.5. Labelling/annotation of data

This is another important step in this stage due to the infrastructure and coordination needed and to the direct impact that the labels quality and accuracy have over the final performance of the model. This process involves assigning a 'ground truth' label that represents the desired output of the machine learning model to each data sample in your dataset. Essentially, it defines the correct response expected from the model given a specific data sample.

Depending on the complexity of the task for which you're training your model and the size of your dataset, the labelling process can be both time-consuming and costly.

In today's digital age, numerous online and offline tools are available to assist with label creation. The choice of tool often depends on whether it meets the requirements of the annotation process. This includes supporting the desired label type and data format for annotation. The format of the annotations is also a crucial consideration as it will be interpreted by the model.

To reduce costs and time, semi-automatic labelling processes are increasingly being adopted. This involves using pre-trained models with low precision in their predictions to perform an initial round of annotations, significantly reducing manual effort.

2.1.2. Training

Once the data is prepared, the next phase is training. The objective here is to train an algorithm to learn patterns from the data, enabling it to make accurate predictions or decisions when presented with new data. A significant challenge in this stage is explaining the model's predictions or decisions. For a long time, machine learning algorithms have been viewed as 'black boxes,' with the basis of their decisions or what they've learned from the data remaining unknown. This is changing due to the general tendency and demand on making the algorithms explainable. This involves implementing techniques during the model's development and testing stages that help understand the reasoning behind its decisions and provide more information about how each feature contributes to each individual instance's prediction.

The following aspects should be considered during this stage:

2.1.2.1. ML software framework

Thanks to the progress in the development of ML frameworks and software libraries for ML algorithms, it has become accessible and easier to create a ML model. Numerous frameworks support various stages of model development, including training and testing, by providing functions that automate the learning process. The choice of framework depends on its provided functionalities, ease of integration with other tools, efficiency, scalability, and community support for troubleshooting. As of the time of writing this

document, [TensorFlow](#) and [Pytorch](#) are among the most popular ML frameworks due to their comprehensive features and extensive user communities.

2.1.2.2. Repository

In this context, a repository is a storage location where all the models or derivative data of training ML model are stored. This includes data, models, results, documentation, and scripts for training, evaluating the model, pipelines, and APIs. Setting up an online repository can make collaboration among developers easier. It is important to choose a repository that is compatible with the rest of the pipeline.

2.1.2.3. Model versioning

Besides storing the model, it is crucial to do a versioning as well. This process helps track and manage different models resulting from various experiments, including not just the model file, but also the parameters and algorithms used to create that model. Implementing versioning ensures reproducibility, traceability, and accountability of experiments, it allows for recovery of past models, and provides a clearer view of the experimentation strategy. Some tools, like DVC, allow for both data and model versioning simultaneously.

2.1.2.4. Training monitoring

Monitoring the training of the algorithm involves real-time observation of the learning process. This allows developers to identify potential issues early or anticipate overfitting. Some tools provide graphical monitoring of training and validation metrics during the model's learning process online, enabling monitoring outside the development lab facilities.

2.1.3. Testing

At this stage, you test the model performance. The processes in this stage are like the previous training stage, since the final model can be saved in a repository, versioned and is dependent on a framework. Besides these, testing involves other sub-processes, such as:

2.1.3.1. Continuous integration and continuous delivery (CI/CD)

As inferred from this document, a machine learning lifecycle consists of some stages which can be iteratively executed. Automating the testing and deployment of your machine learning models using continuous integration and continuous delivery pipelines can accelerate the process of having a final stable model working in its operational environment. This means that once the model is tested and it passes the quality and performance requirements, it can be automatically deployed to start functioning. Some examples of CI/CD tools are [Jenkins](#), [GitHub Actions](#), and [Azure DevOps](#).

2.1.3.2. Model serialization

Saving and loading the ML model in a standard format, allows it to be easily transferred and deployed across different platforms and environments. This also helps in sharing the model without having incompatibility issues. Using a standard format for the model improves its accountability. Some examples of serialization formats are [ONNX](#), and TensorFlow SavedModel.

2.1.4. Deployment

Deployment is the final stage in the machine learning lifecycle, this is when a final model is placed in its operational environment to start making predictions. Here, some processes are shared with the previous stages, such as versioning and serialization. Other deployment sub-processes include:

2.1.4.1. API

One way of deploying a machine learning model is to create an API (application programming interface) that can receive requests and return predictions from the model. An API is a standardized way of communicating with the model, regardless of the underlying platform or technology. This can make the model accessible to other users or services that connect through the API and use the model to various applications and platforms, such as web apps, mobile apps, IoT devices, etc. Some examples of API frameworks are [FastAPI](#) and [Azure Machine Learning](#).

2.1.4.2. Model monitoring

When testing the model, its performance over the testing data is calculated. However, the important performance metric is when the model performs the task it was trained for, when it has been deployed in a real environment. It is important to keep monitoring the model after its deployment, this is done by collecting and analysing metrics and logs to ensure it is performing as expected. Some measurements to observe can be the accuracy, latency and availability. After the deployment, the model can have data drift or concept drift. This is when the data distribution changes from the one that the model was trained with or the task it was trained for changes over time. It is important to monitor these events to ensure that the model is still valid and reliable at the current conditions, mitigating possible risks.

2.1.4.3. Model optimization

Today's rapid inclusion of Internet of Things (IoT) devices which can have low computational resources, into people's daily scenarios and activities, makes it difficult to develop ML solutions with heavy and robust models. Some algorithms demand real-time functioning and sometimes this is only possible through the optimization of the model. There are some techniques to help higher the performance and efficiency of the model, such as pruning, quantization, distillation, and compression. Optimization can help you reduce the size, memory, and latency of the models, as well as increase their accuracy and robustness. Some examples of optimization tools are TensorFlow Lite, PyTorch Mobile, and Azure Machine Learning.

2.2. Operational environment

Additionally, to the stages and processes that a machine learning model development has, there are other key tools that are or could be present in this life cycle. There are all the other processes that support the development and not necessarily affect the model. These can include:

2.2.1. Storage

In the context of the ML framework, storage can be concerned with the data that is applied for training/validating ML models or related to the artifacts generated during the training process such as features, or even related to the final model (after the training process). We highlight below some tools that can be used in one of these three steps:

2.2.1.1. AI-focused data storage

The objective of an efficient data storage is not just storing raw data but also to provide quick access to it and facilitate data sharing as well. For this task, there are some good alternatives such as [IBM AI storage](#) or even Amazon.

2.2.1.2. Feature storage

Each time that an ML model needs to be trained, a pre-processing step needs to be executed to prepare the data in the right format to be fed to the ML algorithm. Moreover, this pre-processed data also serves as transformed data with the purpose of enhancing the learning process of these algorithms. This pre-processed data is known as features and is very common that the same dataset (and features) is used across different ML algorithms with the objective of finding the best ML model.

The objective of the feature store is to store the features already pre-processed by ETL solutions and avoid new pre-processing and help the traceability of feature engineering processes.

Examples of databases to be applied with the objective of storing features: [Hopsworks](#), [Google Vertex](#), [Amazon SageMaker](#), [H2O.ai](#), [Featureform](#).

2.2.1.3. Model storage

For storing ML models, besides some tools such as Gitlab, Github, we can also mention [Neptune.ai](#), [Weights & biases](#), or even object storage databases.

2.2.1.4. DPU

Finally, a more recent concept is the use of a data processing unit (DPU). A DPU is a system on a chip, or SoC, that combines an industry-standard, high-performance, software-programmable, multi-core CPU. It has a high-performance network interface capable of parsing, processing, and efficiently transferring data at line rate, or the speed of the rest of the network, to GPUs and CPUs. It contains a rich set of flexible and programmable acceleration engines that offload and improve applications performance for AI and machine learning, zero-trust security, telecommunications, and storage, among others.

2.2.2. Computing

This refers to the platform where all the processes are run. There are at least two options: on-premises or cloud. For cloud services, there are many options in which the biggest ones are [Amazon AWS](#) and [Google cloud platform](#).

2.2.3. Virtualization

To make deployments easier, virtualization is a good method to create a working environment encapsulated, able to reproduce and duplicate. The most used solutions are [Docker](#) (container) and [VM Ware](#) (virtual machine). The main difference between these two choices is the level of isolation.

2.2.4. Orchestration

There are several types of orchestration which we can mention two: Service orchestration and release orchestration. For service orchestration, we can highlight [Kubernetes](#), [Ansible](#), [BMC TrueSight](#), and ActiveBatch. For release orchestration, we can highlight [Travis CI](#) (for continuous integration), and [Jenkins](#).

2.3. Tools for ML management framework

Below, we describe the main tools for data, training, testing, deployment, and general development for ML life cycle management. We also provide links to their respective websites in the table “references”. Worth mentioning that this is not an exhaustive list given the high-speed evolution of tools available every week. However, we believe that this list covers mostly the main aspects needed for performing a reliable ML life cycle management.

2.3.1. Data

Process:	Tools:
Pre-processing	Dataflow [1], Kedro [2], PyTorch [3]
Statistics	Dataflow [1], ClearML [5], Weights&Biases [6], CleanLab [7], PyTorch [3]
Versioning	Dataflow [4], DVC [8], ClearML [5], Weights&Biases [6], Kedro [2], ZenML [9], Comet [10], ActiveLoop [11], DagsHub [12], PyTorch [3]
Sources	ClearML [5], Kedro [2], ActiveLoop [11], AWS (S3) [14]
Labelling	WebLabel + VCD, CVAT [4]

2.3.2. Training

Process:	Tools:
Framework	ONNX [13], TensorRT [16], PyTorch [3]
Repository	Gitlab [15], MLFlow [17], ClearML [5], Weights&Biases [6], ZenML [9], Kubeflow [18], Snakemake [19], Comet [10], DagsHub [12], Github [20]
Versioning	Kedro [2], TensorBoard [21], Gitlab [15], MLFlow [17], ClearML [5], Weights&Biases [6], ZenML [9], Kubeflow [18], Snakemake [19], Comet [10], DagsHub [12], Github [20]
Monitoring	Kedro [2], MLFlow [17], ClearML [5], Weights&Biases [6], ZenML [9], Kubeflow [18], Snakemake [19], Comet [10], DagsHub [12]

2.3.3. Testing

Process:	Tools:
CI/CD	Gitlab [15], MLFlow [17], ClearML [5], Weights&Biases [6], ZenML [9], Snakemake [19], Comet [10], DagsHub [12], Github [20]
Framework	ONNX [13], TensorRT [16], PyTorch [3]
Versioning	MLFlow [17], ClearML [5], Weights&Biases [6], Kedro [2], ZenML [9], Kubeflow [18], Snakemake [19], Comet [10], DagsHub [12]

2.3.4. Deployment

Process:	Tools:
API	MLFlow [17], Weights&Biases [6], BentoML [22], Kedro [2], Kubeflow [18], Snakemake [19], Comet [10], PyTorch [3], Docker [23]

Serialization	ONNX [13], TensorRT [16], TensorRT [16], PyTorch [3]
Versioning	MLFlow [17], ClearML [5], Kubeflow [18], PyTorch [3], Docker [23]
Monitoring	MLFlow [17], ClearML [5], Weights&Biases [6], Kedro [2], Kubeflow [18], Snakemake [19], Comet [10]
Optimization	MLFlow [17], Weights&Biases [6], ONNX [13], TensorRT [16], PyTorch [3], Docker [23]

2.3.5. General

Process:	Tools:
Computing	PyTorch [3]
Visualization	ClearML [5], CVAT [4], TensorBoard [21], Weights&Biases [6], Kubeflow [18], Comet [10], Actueloop [11]
Storage	Docker [23], Neo4j [24]
Virtualization	Kedro [2], Docker [23]
Orchestrator	MLFlow [17], ClearML [5], SnakeMake [19], DagsHub [12]

2.4. Diagram

Considering the different stages and processes of a ML (Machine Learning) lifecycle identified above, a diagram was designed to represent the methods and tools that are implemented in a ML (machine learning) solution. This diagram, presented in Figure 1, allows to easily identify the flow of the lifecycle of the model and the compatibility between the used tools.

It is mainly composed by four main stages or pillars of a model lifecycle represented by rounded, non-filled rectangles: Data Collecting, Training, Testing and Deployment. Within each of these stages there are some subtasks or processes that are common to perform when building a ML model, they appear in the diagram by filled rectangles inside the stage block. Each of these are explained in Section 2.1 ML model life cycle.

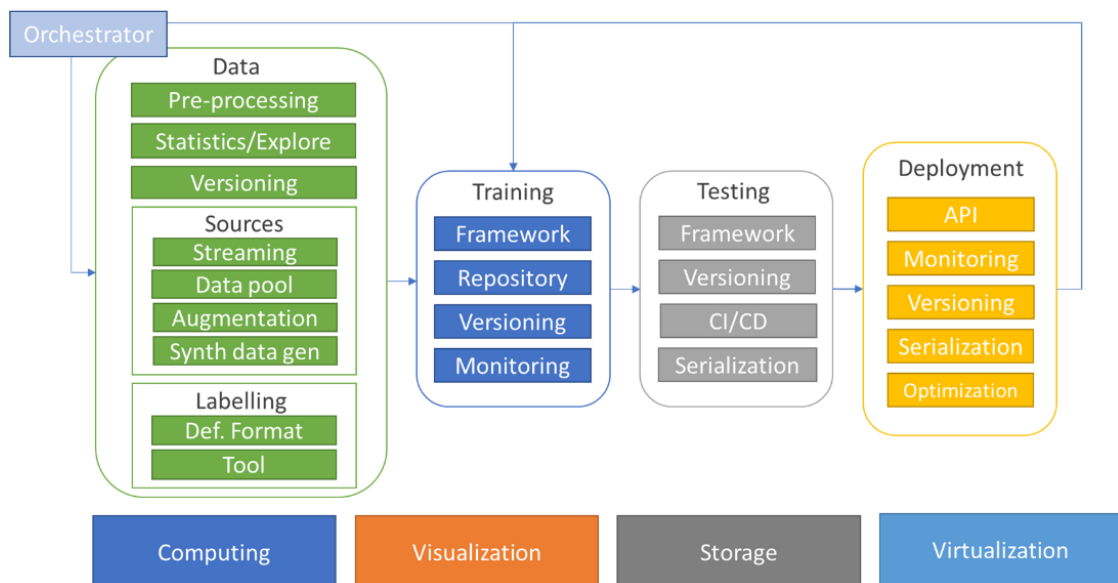


Figure 1: ML life cycle diagram

Additionally, to the stages, there are other components or processes that support the building of the model during the whole lifecycle (explained in Section 2.2 Operational environment). These are represented in the diagram as filled rectangles on the down part of the diagram; these are aspects as Computing, Virtualization, Storage and Visualization. Along with Orchestration that is positioned on the left top of the diagram indicating that it orchestrates the execution of the stages and other processes.

Furthermore, from the stage blocks, there are arrows indicating the flow of the lifecycle and the connection between them. These arrows can be changed to represent an iterative process or a one-pass development. This allows for flexibility in representing the flow of the model’s lifecycle and the connections between the different stages.

This diagram is proposed to provide a unified way of representing each solution within the project. It is designed to work in a dynamic way, where the developer can simply place the logo of the tool or method they used in each process on top of the rectangle representing that process. This allows for easy visualization and understanding of the tools and methods used in each stage of the ML lifecycle. Some implementations of this diagram for the description of models of the project are included in Chapter 3 Robustness assessment using ML tools and in Section 5 Description of ML frameworks.

3. Robustness assessment using ML tools

In the development of AI algorithms and systems engineering, assessment of the entire system’s robustness is a complex process. As presented in previous chapters, several different tools and workflows are available. In this chapter we provide an example using the framework and some of the tools presented, to provide practical insight in using the framework.

In the case of CCAM development, a typical, data-driven workflow (continuous integration and continuous deployment) is presented in Figure 2 (ASAM, n.d.) in which synthetic data plays a key role, especially since synthetic data-driven development is allowing designers to further improve and refine their machine learning models.

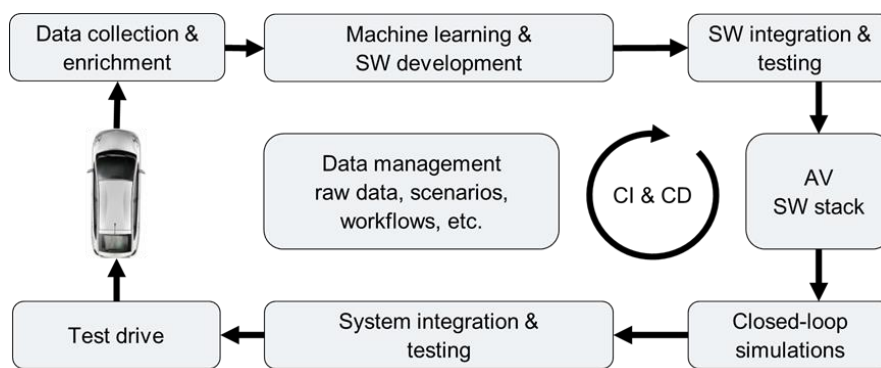


Figure 2: Continuous integration and continuous development

A possible application is shown in Figure 3, where the synthetic data is used to train, validate and assess the robustness of machine learning models.

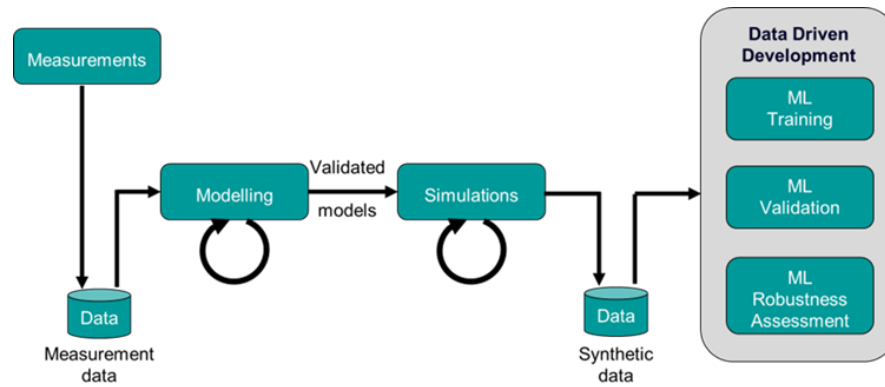


Figure 3: Application of the synthetic data set in ML

Currently, Siemens Industry Software Netherlands B.V. uses synthetic data for the robustness assessment of the sensing and perception stack, which relies on YOLOv3 (Redmon & Farhadi, 2018). We remark that the YOLOv3 has been trained using the real-world COCO data set and hereby the robustness assessment is done using a synthetic data set.

Robustness assessment of camera-based sensing and perception stack requires the definition of a reference data set, an augmented data set, a distance metric to assess the similarity between the reference data set and augmented data set as well as a performance metric such as the validation accuracy of the network. A block diagram of the robustness assessment of the sensing and perception subsystem is presented in Figure 4.

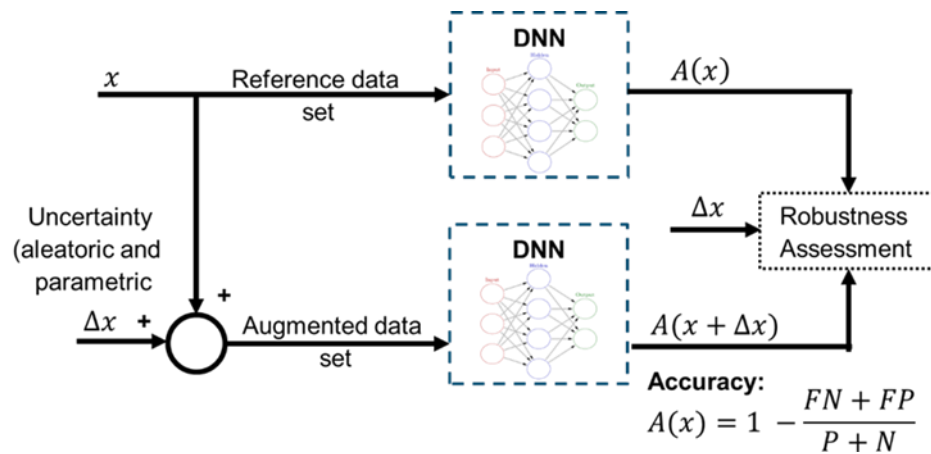


Figure 4: Robustness assessment of machine learning models

Since the development, deployment and use of AI systems should meet the seven key requirements for trustworthy AI (see again introduction of Section 1.3 Trustworthiness in ML models), our investigations presented hereby are related to technical robustness and safety considering diversity, non-discrimination, and fairness. In this sense, the robustness of the sensing and perception stack is studied, when different vulnerable road users, which belong to different age, gender and ethnic categories are considered.

Simulation tools, such as Simcenter Prescan (Siemens Digital Industries Software, 2023), already provide many different predefined vulnerable road users (VRUs). As part of further work, the following additional classes of VRUs, have been considered: VRUs = {Male_Regular (as reference), Female_Regular, Male_African, Male_CyclingCyclist, Child_Regular, Female_wBuggy,

Male_Old_White_WithStick, Female_wShoppingCart}, object classes defined in Simcenter Prescan. The scenario considered is an urban driving scenario, when the VRU is crossing in front of the vehicle.

According to the Althena ML development and life cycle management framework for robustness assessment only the following components are used (highlighted items – see Figure 5). Matlab 2021 is used as Orchestrator. For the data versioning, Github is implemented. Simulated data is generated from Simcenter Prescan 2302, Matlab 2021 and SUMO 1.12.0, making augmentations with Simcenter Prescan 2302 and Matlab 2021. The labelling is done with the same simulation tool and in KITTI format. For the testing, Matlab 2021 is implemented as framework and serialization tool. Finally, to have versions, Github is used.

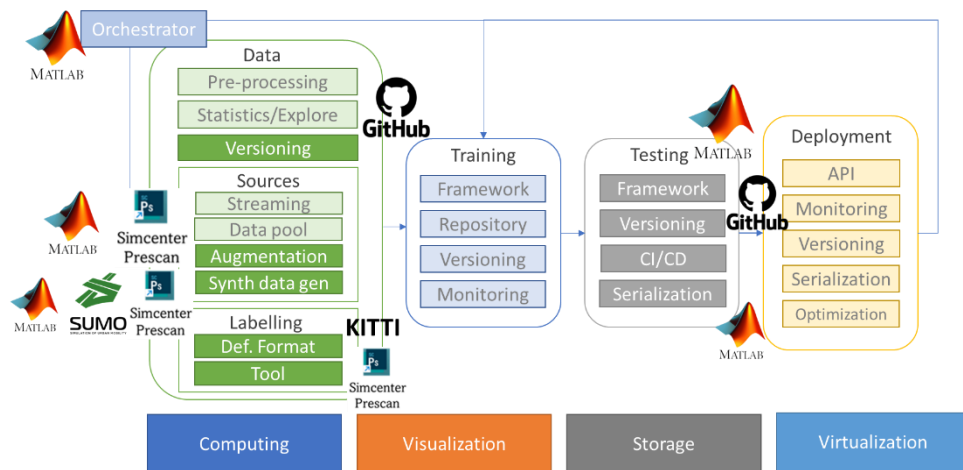


Figure 5: ML development and life cycle management framework diagram

A detailed overview of the ML model robustness assessment life cycle management framework (Hotait & Forrai, 2023) is presented in Figure 6, where the robustness assessment workflow is further detailed in Figure 7.

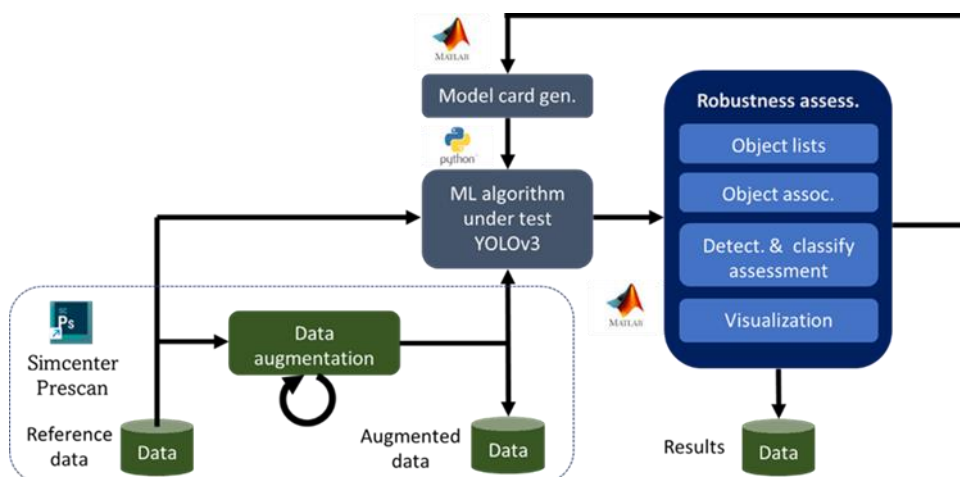


Figure 6: Model robustness assessment life-cycle management framework

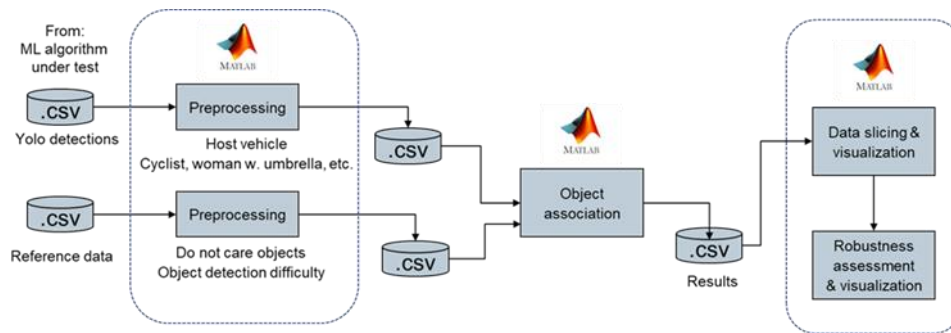


Figure 7: ML robustness assessment workflow in detail

4. Model Cards

4.1. Definition

Many mature industries have long been reliant on standardized documentation practices to ensure the trust of their products' users. This trust was built upon the transparency over the product's performance metrics evaluated under relevant conditions. Additionally, adopting a standardized documentation practice allowed users to make informed decisions about the suitability of the product. Being an emerging industry, the field of artificial intelligence has been set at adopting its own documentation practices drawing inspiration from declarations of conformity, datasheets, and even nutrition labels. The progress so far can be contributed to the recent work done by some of the most prominent tech leaders. Initiatives led by researchers at Google, Microsoft, IBM, and Meta, have started the conversation as to what should be the content of the documentation, some already offering toolkits and templates to generate the documentation.

At the time of its publishing, the paper on Model Cards (Mitchell, et al., 2019) was the first to introduce a documentation framework that was to accompany a trained machine learning (ML) model. The goal was to provide a standardized documentation procedure that will communicate not only the performance evaluation of the model but extend it with model development and usage information. From just this short document, stakeholders across varying levels of expertise could be informed of the contexts in which the model should and should not be used. As the authors explained, the proposed model cards should be seen as "a step towards the responsible democratization of machine learning and related artificial intelligence technology, increasing the transparency into how well artificial intelligence technology works."

Nonetheless, proposing of a standardized model reporting was not all what authors did. They have additionally brought attention to the lack of ethical considerations in performance evaluation, primarily those of fairness and inclusivity. These considerations are significant in building trust, especially in human-centric applications that may leave a great impact on people's lives. Therefore, it was necessary to include ethical considerations among other model card sections, in hopes of incentivizing developers to report model behaviour across impacted groups.

Recognizing the importance of model evaluation across different sociocultural groups is not unique to model reporting. Often used as a complement to model cards, datasheets for datasets (Gebru, et al., 2021) (and other dataset documentation frameworks) report handling of sociocultural data at different stages of dataset generation. As many socially sensitive questions remain open, it has become evident that a more interdisciplinary approach (Jo & Gebru, 2020) would be necessary to define how the groups relevant to the intended use of the model or dataset should be defined.

It is important to notice that ML-centred documentation can be provided not just at the model or dataset level, but at the level of entire services or systems. Parallel to the introduction of model cards, IBM researchers published a paper presenting another documentation alternative called FactSheets (Richards, Piorkowski, Hind, Houde, & Mojsilović, 2020) (Arnold, et al., 2019). Using the supplier's declaration of conformity as inspiration (SDoC), FactSheets were aimed at the consumers of entire AI services. As such, FactSheets were intended to provide information like one found in model cards but about AI services that are often built from many ML models trained on many datasets. Similarly, in the case of ML systems, it would be insufficient to only report on individual building blocks of the system. Instead, system-level documentation should, in addition to component information, provide information on how components interact and how is the data transferred between them. To address the question of system-level documentation, Meta introduced the most recent addition to documentation practices called System cards (Chavez, et al., 2022). System cards are intended to provide both expert and nonexpert stakeholders with an overview and detailed information about the system. It is the claim of providing detailed information that determines the future use of model cards as authors achieve the level of component detail with continued use of model cards. Therefore, making the model cards a fundamental building block of ML-centred documentation.

However, what preceded the adoption of model cards as part of an integrated documentation framework such as System cards was its exclusive use for ML model reporting. Many ML developers have recognized the necessity and soon the benefit of a standardized model documentation practice that the model cards offered. The same year the concept of model cards was introduced, Google developed a Model Card Toolkit (MCT) (Fang & Miao, 2020) that was to be integrated into the ML pipeline allowing the developers to automate the compilation of the information on the model's development and performance. Even before the release of MCT, Google was already generating and publishing model cards along with the publication of computer vision models developed by their MediaPipe team (Google, 2019). Another prominent name whose developers adopted the use of model cards was OpenAI. Starting from the earlier version of their natural language-generating model GPT-2 (Open AI, 2019), followed by GPT-3 (Open AI, 2020), as well as their text-to-image generating model DALL-E (Open AI, 2021), OpenAI has been releasing accompanying model cards on their GitHub repository. Accompanying ML models featured in NVIDIA's NGC catalogue (NVIDIA, 2023) are versions of model cards that have been recently, after months of research and surveying, upgraded to their Model Card++ (Boone, Pope, Xiao, & Anandkumar, 2022). NVIDIA's team identified that besides performance and licensing information, the third most important section was the section on ethical considerations. This led to the incorporation of ++Promise, a list of standards NVIDIA's developers can be held to, and additional subsections detailing information on bias, explainability, privacy, safety, and security.

From their first introduction, model card sections, and their definitions have undergone multiple iterations of changes and improvements. However, recent research points out that more iterations are yet to come. Some even raise the question if it is even possible to create one standardized model card that could be exhaustive enough and yet understandable to all technical and nontechnical individuals alike. Whether such a representative model card can be accepted across the entire ML field, it is evident that it would still benefit from the flexibility of adapting to a specific ML model. Therefore, the model card template used to document models developed as part of the Althena project has been based on the templates seen in use today, but with sections extended with model-specific checklists.

4.2. Proposed Model Card

As part of the contribution of this deliverable, there is a proposed Model Card Template for the Althena project. Here, it is included all the principles from the original paper but is adapted to be more compressed.

This template provides detailed information about the content that should be included in each section, along with a checklist to guide developers in providing the right and exhaustive amount of information about their models. Following the user-centred approach adopted in the Althena project, the checklist is a more user-friendly way to request information from developers. It allows for highlighting what is relevant to include when describing a model and provides guidance on the amount and level of detail that should be included. This helps ensure that all relevant information is presented in a clear and comprehensive manner, being also flexible to add extra details about the model. This allows for greater transparency and understanding of the model and its capabilities.

The Althena Model Card includes the following sections and sub-sections:

- **Introduction.** In this section, the developer can provide a brief and general information about the authors and the model. This includes, contact details of the author, affiliation, etc, and a summary of the model, relevant information as well as license. It has the following sub-sections:
 - **Authors details:** *Add information about the individuals or organizations responsible for developing the model.*
 - PROVIDED A DETAILED INFORMATION OF THE AUTHORS**
 - Added the name and the affiliations of the authors*
 - Added contact details*
 - Added important links*
 - Added affiliation logo*
 - **Model info:** *Here, it goes a summary of the model, relevant information as well as license.*
 - PROVIDED A BRIEF DESCRIPTION OF THE MODEL**
 - Added a brief description of the model*
 - Added the license of the model*
 - Added the version and the creation date of the model being described in this Model Card*
 - Added links to repositories or website*
 - Added relevant scientific papers citation and links*
 - Added an illustrative figure or an example of inference of the model*
 - **Model details.** This section provides more detailed information about the model, including its architecture, task, inputs and outputs. Information about the data used to train and test the model is also requested. Since a separate Data Card template will be provided for the Althena project, in-depth information about the data will not be included in the Model Card. It has the following sub-sections:
 - **Model architecture:** *Here, you must provide a detailed description of the model architecture, including hyperparameters, type of model (e.g., neural network, model decision tree), task (e.g., Classification, regression) the number of layers or nodes, or other relevant details. Include diagrams or visualizations to help explain the architecture.*
 - PROVIDED A DETAILED DESCRIPTION OF THE MODEL ARCHITECTURE**

- Described the type of model (e.g., neural network, decision tree)
- Specified the task of the model (e.g., classification, regression)
- Provided information about the number of layers or nodes
- Provided information about the inputs and outputs of the model
- Included diagrams or visualizations to help explain the architecture

- **Training data:** The user can provide detailed information about the data used to train the model. This can include source, size, preprocessing, data splits, data augmentation. *A Data Card can be provided instead.*

PROVIDED DETAILED INFORMATION ABOUT THE DATA USED TO TRAIN THE MODEL

- Described the dataset (e.g., size, format, number of instances, classes, features)
- Specified the source of the data (e.g., a public dataset, created data, simulated)
- Provided statistics or visualizations to show the distribution of classes
- Documented any preprocessing steps that were applied to the data

PROVIDED INFORMATION ABOUT HOW THE DATA WAS SPLIT INTO TRAINING AND VALIDATION SETS

- Described the method used to split the data (e.g., random sampling, stratified sampling)
- Provided statistics or visualizations to show the distribution of classes in each set

DOCUMENTED ANY DATA AUGMENTATION TECHNIQUES THAT WERE APPLIED TO THE TRAINING DATA

- Described the data augmentation techniques used (e.g., rotation, flipping)
- Provided examples of augmented data instances

- **Evaluation data:** The user can provide detailed information about the data used to evaluate the model. This can include source, size, preprocessing, evaluation method. *A Data Card can be provided instead.*

PROVIDED DETAILED INFORMATION ABOUT THE DATA USED TO EVALUATE THE MODEL

- Described the dataset (e.g., size, format, number of instances, classes, features)
- Specified the source of the data (e.g., a public dataset, created data, simulated)
- Documented any preprocessing steps that were applied to the data

- **Intended use.** Here, the developer can describe the intended use of the model. Mentioning possible scenarios of implementation, the description of the users expected to use the model and the ones the model will be analyse. It is also important to provide information about the limitation of the model, under what situations it may not achieve the desired performance, etc.

- **Use cases & users:** Here, there is the description of the intended use for the model. This can include information about the problem the model is intended to solve, the target audience or users of the model, and any specific scenarios, situations, or users in which the model is intended to be used.

CLEARLY DESCRIBED THE INTENDED USE CASES FOR THE MODEL

- Identified the problem the model is intended to solve

Provided examples of specific scenarios or situations in which the model is intended to be used (e.g., illustration, storytelling)

Provided an ODD (Operational Design Domain)

DESCRIBED THE TARGET AUDIENCE OR USERS TO USE MODEL

Demographics: [Information about the age, gender, location, etc. of the intended users]

Expertise: [Information about the level of expertise or technical knowledge required to use the model]

Needs: [Information about the specific needs or goals of the intended users]

DESCRIBED THE USERS TO BE ANALYSED BY THE MODEL (IF ANY)

Demographics: [Information about the age, gender, location, etc. of the affected individuals or groups]

Needs: [Information about the specific needs or goals of the affected individuals or groups]

Risks: [Information about any potential risks or harms that may appear during the use of the model]

- **Limitations:** Provide information about any limitations or restrictions on the use of the model. This can include technical limitations as well as ethical or legal limitations.

CLEARLY DOCUMENTED ANY LIMITATIONS OR RESTRICTIONS ON THE INTENDED USE OF THE

MODEL

Specified any technical limitations (e.g., the model only works with certain types of data, situations, scenarios)

Described any ethical or legal limitations (e.g., the model should not be used for certain purposes)

PROVIDED GUIDANCE ON HOW TO ADDRESS ANY LIMITATIONS OR RESTRICTIONS

Suggested ways to mitigate or work around any technical limitations

Provided guidance on how to use the model in an ethical and responsible manner

IDENTIFY POTENTIAL BIASES

Describe any potential biases in the model, such as biases in the training data or biases in the model's predictions.

Describe any steps that have been taken to mitigate potential biases or negative impacts of the model or the ones you suggest.

- **Evaluation & performance.** The evaluation or testing process of the model can be reported in this section, providing metrics and subjective observations about the performance of the model: accuracy, confusion matrix, etc. All the detail about how this evaluation was done and the criteria should be explained. Graphics or an example of the model's inference are valuable information to be included in this section.

- **Metrics:** Description of the specific metrics used (e.g., accuracy, precision, recall) to evaluate the model, as well as any relevant thresholds or criteria for determining good performance and evaluation techniques.

CLEARLY DESCRIBED THE PERFORMANCE METRICS USED TO EVALUATE THE MODEL

Used accuracy to measure overall performance

Used precision and recall to measure performance on specific classes

Used F1 Score to measure performance on an unbalanced dataset

Used other performance metrics (please specify): _____

- PROVIDED ANY RELEVANT THRESHOLDS OR CRITERIA FOR DETERMINING GOOD PERFORMANCE**
 - Established minimum acceptable values for each performance metric*
 - Compared the model's performance to relevant benchmarks or baselines*
- PROVIDED INFORMATION ABOUT HOW THE EVALUATION WAS PERFORMED**
 - Described the evaluation method used (e.g., cross-validation, holdout set)*
 - Provided statistics or visualizations to show the model's performance*
- **Results:** *Results of the performance evaluation, including any relevant statistics or visualizations are described in this section. This can include information about the overall performance of the model, as well as its performance on specific subsets of the data or scenarios (e.g., different demographic groups).*
 - PROVIDED THE RESULTS OF THE PERFORMANCE EVALUATION, INCLUDING ANY RELEVANT STATISTICS OR VISUALIZATIONS**
 - Reported overall performance using summary statistics (e.g., mean, median)*
 - Visualized performance using graphs or charts (e.g., confusion matrix, ROC curve)*
 - DOCUMENTED THE MODEL'S PERFORMANCE ON SPECIFIC SUBSETS OF THE DATA**
 - Reported performance for different demographic groups (if is the case)*
 - Reported performance for different subpopulations or scenarios*
 - Reported performance for defined ODD*
 - PROVIDED GUIDANCE ON HOW TO INTERPRET THE PERFORMANCE RESULTS**
 - Explained how to use the reported statistics or visualizations to assess the model's performance*
 - Provided context or comparisons to help users understand the results*
- **Ethical considerations.** *Models should consider ethical considerations such as fairness, privacy, and accountability. In this section, the developer can provide information of how the model is including these aspects. It can include the report of fairness metrics or bias, documentation of how the data is processed.*
 - **Fairness:** *In this section there is the discussion of any potential biases in the model or its intended use. This can include information about how the model was tested for fairness (e.g., using fairness metrics), as well as any steps taken to mitigate potential biases (e.g., reweighting the training data).*
 - TESTED THE MODEL FOR FAIRNESS USING APPROPRIATE [fairness metrics](#)**
 - Used statistical parity difference to measure group fairness*
 - Used equal opportunity to measure individual fairness*
 - Used other fairness metrics (please specify): _____*
 - MITIGATED ANY POTENTIAL BIASES IN THE MODEL OR ITS INTENDED USE**
 - Rewighted the training data to balance underrepresented groups*
 - Modify loss function to weight the loss contribution of the classes*
 - Used other bias mitigation techniques (please specify): _____*
 - DOCUMENTED ANY REMAINING BIASES OR LIMITATIONS IN THE MODEL CARD**
 - Explained how representative is the train and test data used for the intended use case*
 - **Privacy:** *Here, there is information about how the model handles user data and any privacy considerations. This can include information about what data is collected,*

how it is stored and used, and any measures taken to protect user privacy (e.g., data anonymization).

[] CLEARLY DOCUMENTED WHAT DATA IS COLLECTED, HOW IT IS STORED AND USED

[] Provided a data map showing what data is collected and how it flows through the system or clearly described the dataflow of the system

[] Described the data retention policy and how long data is stored

[] Explained how user data is used and shared with third parties

[] IMPLEMENTED MEASURES TO PROTECT USER PRIVACY

[] Used data anonymization techniques to de-identify user data

[] Encrypted user data at rest and in transit

[] Implemented access controls to restrict who can view user data

[] Used federated learning techniques

[] COMPLIED WITH RELEVANT PRIVACY REGULATIONS AND STANDARDS

[] Obtained user consent for data collection and use

[] PROVIDED USERS WITH THE ABILITY TO ACCESS, CORRECT, OR DELETE THEIR DATA

[] Informed the user of ways to ask for deletion of their data

- **Accountability:** *There is information about how decisions made by the model can be explained and reviewed. This can include information about the explainability of the model (e.g., using explainable AI techniques), as well as any processes in place for reviewing and auditing the model's decisions.*

[] ENSURED THAT THE MODEL'S DECISIONS CAN BE EXPLAINED AND REVIEWED

[] Used explainable AI techniques to make the model's decisions more transparent

[] IMPLEMENTED PROCESSES FOR REVIEWING AND AUDITING THE MODEL'S DECISIONS

[] Established a process for users to report concerns or errors

[] Conducted regular revisions to ensure that the model is performing as intended

[] DOCUMENTED ANY LIMITATIONS OR UNCERTAINTIES IN THE MODEL'S EXPLAINABILITY

[] Clearly communicated any limitations or uncertainties to users

[] Provided guidance on how to interpret the model's decisions in light of these limitations

- **Usage and risks.** The purpose of this last section is to provide detailed instructions and recommendations for using the model. Highlighting the risks and limitation when using the model, this can include ethical or technical issues.
 - **Usage and recommendations:** *Provide detailed instructions and recommendations on how to use the model. This can include information about the input data format, any required preprocessing steps, and how to interpret the model's output.*

[] PROVIDED DETAILED INSTRUCTIONS ON HOW TO USE THE MODEL

[] Described the input data format and any required preprocessing steps

[] Provided examples of how to call the model and obtain predictions

[] CLEARLY DOCUMENTED THE INPUT DATA FORMAT AND ANY REQUIRED PREPROCESSING STEPS

[] Specified the format of the input data (e.g., CSV, JSON, JPG)

[] Described any required preprocessing steps (e.g., normalization, feature selection)

[] EXPLAINED HOW TO INTERPRET THE MODEL'S OUTPUT

[] Described the format of the model's output (e.g., probabilities, class labels)

[] Provided guidance on how to use the model's output in downstream applications

- **Limitations & risks:** *This can include the same information as in the **Intended use** section (i.e., technical, ethical, and legal limitations), as well as any additional limitations or restrictions that may arise during the actual use of the model. Also, identify ethical risk that can be related with the usage of the model.*
 - [] CLEARLY DOCUMENTED ANY TECHNICAL LIMITATIONS OF THE MODEL**
 - [] Specified any restrictions on the type or format of input data*
 - [] Described any known failure modes or scenarios where the model may not perform well*
 - [] CLEARLY DOCUMENTED ANY ETHICAL OR LEGAL LIMITATIONS ON THE USE OF THE MODEL**
 - [] Identified any potential biases or fairness concerns*
 - [] Described any legal or regulatory restrictions on the use of the model*
 - [] PROVIDED GUIDANCE ON HOW TO ADDRESS ANY LIMITATIONS OR RESTRICTIONS**
 - [] Suggested ways to mitigate or work around any technical limitations*
 - [] Provided guidance on how to use the model in an ethical and responsible manner*
 - [] IDENTIFIED THE RISKS**
 - [] Lack of transparency and explainability: [Description of any risks related to the transparency or explainability of the model]*
 - [] Bias and discrimination: [Description of any risks related to bias or discrimination in the model or its intended use]*
 - [] Vulnerability to attacks: [Description of any risks related to the model's vulnerability to attacks or other security concerns]*
 - [] Privacy concerns: [Description of any risks related to the collection, storage, or use of sensitive personal data]*

Any other information that the developer considers essential can be added in the model card. On Section 5

Description of ML frameworks, there are some examples of the implementation of this model card with some developments of Althena.

5. Description of ML frameworks

In this section, some applications of the diagram and the model card proposed for the Althena project will be presented.

5.1. ML model #1

5.1.1. Model #1 life cycle diagram

The life cycle diagram for the first ML model #1 follows the structure of the general ML life-cycle diagram (Computing, Visualization, Storage, Virtualization), presented in Figure 8. The model #1 is trained with a fixed, pre-labelled dataset for 3D Object Detection. In particular, the [nuScenes dataset](#) was used. The framework [Tensorflow-Datasets](#) was used for data-preprocessing and versioning of the datasets. Data augmentation and the streaming of the dataset was done in [Tensorflow](#). Consequently, the model training, monitoring and testing is also performed with the ML framework Tensorflow and Tensorboard.

The model is versioned and stored in a Gitlab storage system. All processes are performed in a CI/CD pipeline. For the deployment and optimization of the model [TensorRT](#), ROS2 and Docker are used.

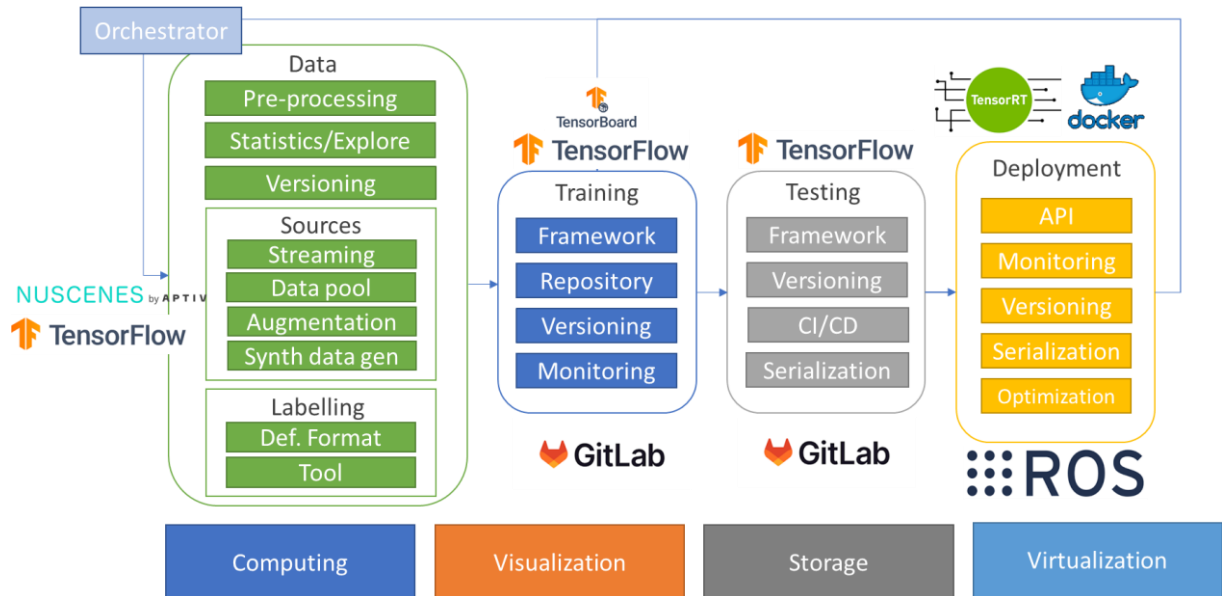


Figure 8: ML life cycle diagram of Model #1

5.1.2. Model Card

The model card of this model can be found in **Appendix A**. This model card reports preliminary information and results about the model. Further versions of this document can be found in further deliverables.

5.2. ML model #2

The following two sections include the ML life-cycle diagram and the model card of a Hybrid AI model being developed as part of the UC-2 AI extended Situational Awareness/Understanding. In general, the model is to fulfil the task of predicting the trajectories of traffic participants in an urban environment. Positioned as the observer of the traffic scene, the model will be presented with both static (e.g., map) and dynamic (e.g., traffic participants) information, based on which long-term trajectory prediction will be generated. One such model would have to cover a broad range of different map configurations and various classes of traffic participants. Therefore, these two factors were used to mark the stages of the model development starting from the stage that will be presented in this deliverable.

For the initial stage of the model development, the scope was restricted to intersection maps with one pedestrian moving within the mapped area per scenario. The task of the restricted scope can then be rephrased as the pedestrian’s trajectory prediction based on the static map information. As the model development progresses through its subsequent stages, more information, both individually and jointly, will be introduced to the model. It is worth mentioning that the added benefit of the proposed staged model development is that it allows for ablation studies to be conducted relative to the available scene information. Gaining insight into the contributions of different information sources will undoubtedly prove valuable for edge cases where the assumption of the available scene information does not hold.

5.2.1. Model #2 life cycle diagram

For the development of this model, the tools used are shown in Figure 9. The programming language chosen was Python along with the ML framework Pytorch. For the versioning of code, repository and storage, Gitlab was implemented. During the training, TensorBoard was used to monitor this process, allowing the visualization as well. Finally, this was executed using virtualization tools such as Docker and Podman.

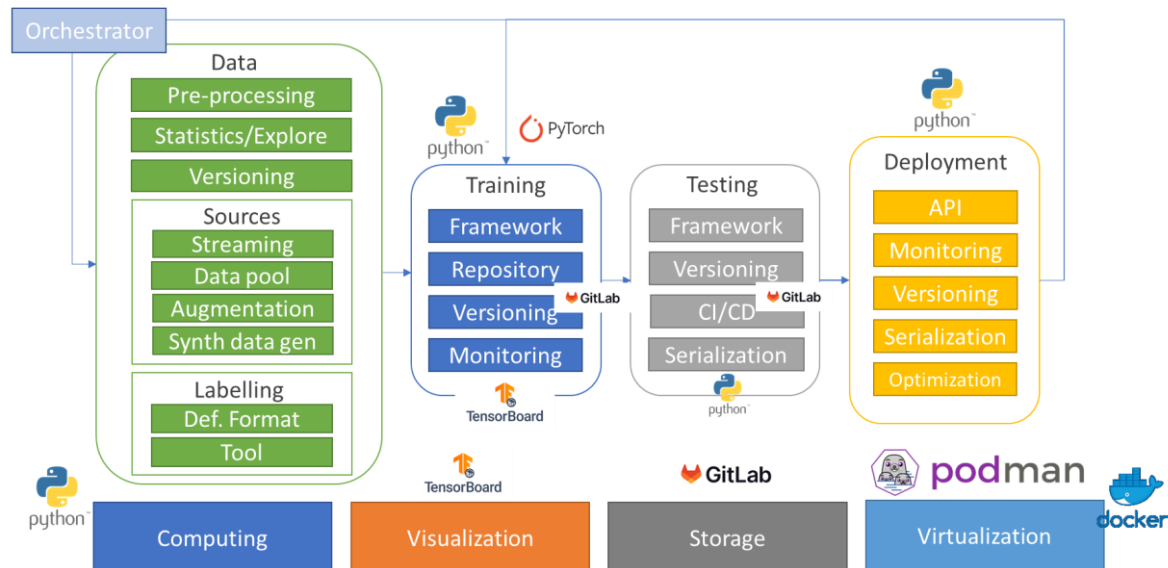


Figure 9: ML life cycle diagram of Model #2

5.2.2. Model Card

The model card of this model can be found in **Appendix B**. This model card reports preliminary information and results about the model. Specifically, sections highlighted in yellow are still to be defined since the model development is an ongoing task at the time of this D.3.1 writing. Further versions of this document can be found in subsequent deliverables.

6. Conclusions

As a first output of Task 3.1, this deliverable provides an overview of ML frameworks from literature and the use of ML tools by the consortium, and it shows a first implementation of the use of ML model cards on several different use cases, which can later be used as a template to automated traceability of code, datasets and artifacts. Using the model cards, the developed AI algorithms can be explained, made more accountable and ensure robustness. Later in the project, the results of Task 3.1 will be embedded in deliverables 3.2 and 3.3.

With this overview, this deliverable provides input for two Althena’s general objectives:

1. GO-2: XAI models – research and development of explainable AI models
2. GO-8: Testing and validation – development of testing and validation methodologies for AI-based CCAM applications

As one of the first deliverables of Althena, as well as being part of the AI development work package WP3, this deliverable mainly focused on providing a first draft of tools to AI developers that can help make their AI algorithms more robust, explainable, accountable and trustworthy. D1.1 focuses more on the strategies that could be taken (broader policy point of view), while D3.1 focuses more on the technical / engineering point of view.

This deliverable presented the methodology and tools for the development and life-cycle assessment of individual ML algorithms. The methodology is based on the Althena human-centric approach, which is described in detail in D1.1. The methodology is applied to the development of ML algorithms for the use cases of the project.

Input from this deliverable will be further taken into the project. First ideas are to automate the process of model cards and test its implementation in the project.

Bibliography

- AI, H.-L. E. (2019). *Ethics guidelines for trustworthy artificial intelligence*. Retrieved from <https://digital-strategy.ec.europa.eu/en/library/ethicsguidelines-trustworthy-ai>
- Arnold, M., Bellamy, R. K., Hind, M., Houde, S., Mehta, S., Mojsilovic, A., . . . Varshney, K. R. (2019, July). FactSheets: Increasing trust in AI services through supplier's declarations of conformity. *IBM Journal of Research and Development*, 63(4/5), 6:1-6:13. doi:10.1147/JRD.2019.2942288
- ASAM. (n.d.). *OpenTEST*. Retrieved 04 14, 2023, from <https://www.asam.net/members/product-directory/detail/open-test-framework/>
- Boone, M., Pope, N., Xiao, C., & Anandkumar, A. (2022). Enhancing AI Transparency and Ethical Considerations with Model Card++. *Enhancing AI Transparency and Ethical Considerations with Model Card++*. Retrieved from <https://developer.nvidia.com/blog/enhancing-ai-transparency-and-ethical-considerations-with-model-card/>
- Chavez, P., Cheema, A., Adkins, D., Alsallakh, B., Green, N., McCreynolds, E., . . . Wang, E. (2022, February). System-Level Transparency of Machine Learning. *System-Level Transparency of Machine Learning*, 1.
- Fang, H., & Miao, H. (2020). Introducing the Model Card Toolkit for Easier Model Transparency Reporting. *Introducing the Model Card Toolkit for Easier Model Transparency Reporting*. Retrieved from <https://blog.research.google/2020/07/introducing-model-card-toolkit-for.html>
- Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumé, H., & Crawford, K. (2021, December). Datasheets for Datasets. *Communications of the ACM*, 64(12), 86-92. Retrieved from <http://arxiv.org/abs/1803.09010>
- Google. (2019). *The value of a shared understanding of AI models*. Retrieved from <https://modelcards.withgoogle.com/about>
- Hotait, H., & Forrai, A. (2023). Digital twin for synthetic data generation - application for automated driving systems. *22nd Driving Simulation and Virtual Reality Conference and Exhibition*, 27-30.
- Jo, E. S., & Gebru, T. (2020, January). Lessons from archives: Strategies for collecting sociocultural data in machine learning. (pp. 306-316). Association for Computing Machinery, Inc. doi:10.1145/3351095.3372829
- Microsoft. (s.f.). *DevOps on Azure*. Obtenido de Microsoft Azure: <https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops>
- Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., . . . Gebru, T. (2019, January). Model cards for model reporting. (pp. 220-229). Association for Computing Machinery, Inc. doi:10.1145/3287560.3287596
- NVIDIA. (2023, July 24). *VehicleTypeNet Model Card*. Retrieved from <https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/models/vehicletypenet>
- Open AI. (2019). *GPT-2 Model Card*. Retrieved from https://github.com/openai/gpt-2/blob/master/model_card.md
- Open AI. (2020). *GPT-3 Model Card Model*. Retrieved from <https://github.com/openai/gpt-3/blob/master/model-card.md>

Open AI. (2021). *Model Card: DALL·E dVAE*. Retrieved from https://github.com/openai/DALL-E/blob/master/model_card.md

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *CoRR, abs/1804.02767*. Retrieved from <http://arxiv.org/abs/1804.02767>

Richards, J., Piorkowski, D., Hind, M., Houde, S., & Mojsilović, A. (2020, June). A Methodology for Creating AI FactSheets. *A Methodology for Creating AI FactSheets*. Retrieved from <http://arxiv.org/abs/2006.13796>

Siemens Digital Industries Software. (2023, August). *Simcenter Prescan*. Retrieved from <https://plm.sw.siemens.com/en-US/simcenter/autonomous-vehicle-solutions/prescan/>

Tools Website references
[1] https://cloud.google.com/dataflow
[2] https://kedro.org/
[3] https://pytorch.org/
[4] https://www.cvat.ai/
[5] https://clear.ml/
[6] https://wandb.ai/site
[7] https://cleanlab.ai/
[8] https://dvc.org/
[9] https://www.zenml.io/home
[10] https://www.comet.com/site/
[11] https://www.activeloop.ai/
[12] https://dagshub.com/
[13] https://onnx.ai/
[14] https://aws.amazon.com/
[15] https://gitlab.com/users/sign_in
[16] https://developer.nvidia.com/tensorrt-getting-started
[17] https://mlflow.org/
[18] https://www.kubeflow.org/
[19] https://snakemake.readthedocs.io/en/stable/
[20] https://github.com/
[21] https://www.tensorflow.org/
[22] https://www.bentoml.com/
[23] https://www.docker.com/
[24] https://neo4j.com/

List of acronyms and terms

Acronym, term	Definition
AI	Artificial Intelligence
CCAM	Cooperative Automotive Mobility
DX.X	Deliverable
KPI	Key Performance Indicators
MX.X	Milestone
TX.X	Task
UC-X	Use Case
WP	Work Package
XAI	Explainable Artificial Intelligence
MLOps	Machine Learning Operations
ML	Machine Learning
IoT	Internet of Things
VRU	Vulnerable Road User
API	Application Programming Interface
CI/CD	Continuous Integration/Continuous Delivery



LIDAR 3D Object Detection MODEL CARD

Author name/s and affiliation	Till Beemelmans, RWTH Aachen University
Contact details	till.beemelmans@ika.rwth-aachen.de
Link/s	Institute for Automotive Engineering
Affiliation logo	
Date	20/10/2023

Disclaimer



Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for the

INTRODUCTION



MODEL DESCRIPTION

PBOD is a 3D object detector designed for autonomous driving. It is a CNN-based detector that uses a multi-view feature engineering approach to improve robustness while maintaining real-time capabilities.

- **Model version:** *Model ID: Vo.125*
- **Model license:** Open source ([Link](#))
- **Citation:** <https://arxiv.org/abs/2007.10323>
- **Repository link:** [Github](#)

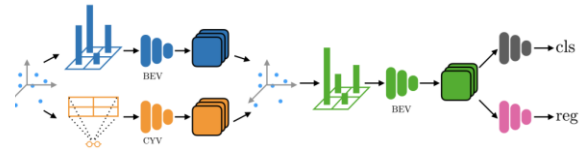


Figure 10 PBOD architecture overview. The model uses a multi-view feature engineering of the point cloud. Output of the model is computed by a classification and regression head.

Model Summary:

Layer (type)	Output Shape	Param #
pillar_net (PillarNet)	multiple	646240
sequential_16 (Sequential)	(None, 256, 256, 64)	185408
sequential_17 (Sequential)	(None, 256, 256, 64)	185408
sequential_18 (Sequential)	(None, 256, 256, 64)	185408
dense_5 (Dense)	multiple	64
dense_6 (Dense)	multiple	576
dense_7 (Dense)	multiple	512

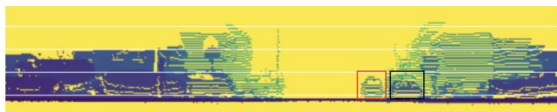
Total params: 1203624 (4.59 MB)
 Trainable params: 1198944 (4.57 MB)
 Non-trainable params: 4680 (18.28 KB)

MODEL DETAILS



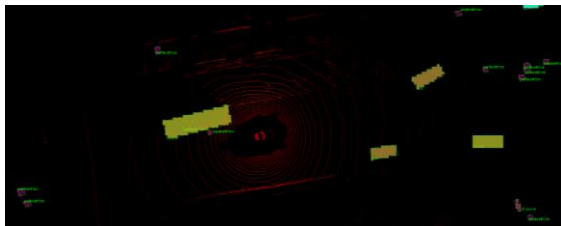
MODEL ARCHITECTURE

- **Model type:** Convolutional Neural Network
- **Model task:** 3D Object Detection
- **Inputs:** 3D Point Cloud (X, Y, Z, Intensity)
 - Cylindrical Point Cloud



(a) Cylindrical View

- Bird's Eye View Point Cloud



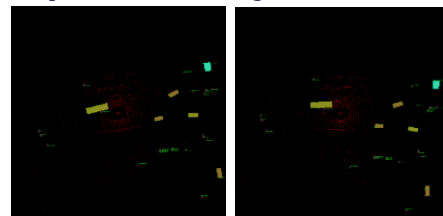
- **Outputs:** 3D Bounding Boxes and Classes
 - Classification (Class probability for each pillar)
 - Regression (Object Position and Dimensions for each object in the pillar)
 - Occupancy (if an object is present in the pillar or not)

$$p = f_{\text{cls}}(\phi^{\text{pillar}}) \quad \text{and} \quad (\Delta_x, \Delta_y, \Delta_z, \Delta_l, \Delta_w, \Delta_h, \theta^p) = f_{\text{reg}}(\phi^{\text{pillar}})$$



TRAINING DATA

- **Dataset description:** [nuScenes](#) 3D Object Detection dataset
- **Data splitting:** The official train/validation split setting of [nuScenes](#) was used.
- **Data augmentation techniques:**
 - Random Augmentation is applied during training. The point cloud and the labels are rotated by a randomly chosen angle of $[-25^\circ, 25^\circ]$, as shown in the Fig. below.



EVALUATION DATA

- **Dataset description:** The official train/validation split setting of [nuScenes](#) was used.

INTENDED USE



USE CASES AND USERS

- **Intended use cases:**
 - **Urban scenarios with many pedestrians:** The model can be used to detect pedestrians in urban environments, such as crosswalks, sidewalks, and intersections. This information can then be used by the self-driving car to avoid collisions with pedestrians.
 - **Sudden pedestrian appearance:** The model can be used to detect pedestrians who suddenly appear from behind obstacles, such as parked cars or buildings. This information can then be used by the self-driving car to brake or swerve to avoid a collision.
- **Target audience or users of the model:**
 - **Self-driving car developers:** The model can be used by self-driving car developers to improve the perception system of their cars.
 - **Automotive companies:** Automotive companies can use the model to develop self-driving cars that are safer and more reliable.
- **Users to be analysed by the model:**
 - **Pedestrians:** The model will analyze pedestrians in the surrounding environment to detect their location.



LIMITATIONS

- **Limitations or restrictions of the intended use:**
 - The model only works with sensor data similar to the data in nuScenes. Therefore, the LiDAR sensor should be of type Velodyne VLP32c or similar.
 - If a different sensor input is used, the model quality might suffer.
 - If the sensor mounting position is changed, the model quality might suffer too.
- **Guidance on how to address limitations or restrictions:**
 - Use a Velodyne VLP32C LiDAR sensor or similar.
 - Do not change the sensor mounting position.
- **Identification of potential biases:**
 - The model may be biased towards pedestrians of a certain age, gender, or size, due to the lack of data on such pedestrians in the nuScenes dataset.

EVALUATION AND PERFORMANCE



METRICS

- **Performance metrics:** The official 3D Object Detection metrics of [nuScenes](#) were used. The metric consists of the following sub-metrics:
 - **Average Translation Error (ATE):** Euclidean center distance in 2D in meters.
 - **Average Scale Error (ASE):** Calculated as $1 - IOU$ after aligning centers and orientation.
 - **Average Orientation Error (AOE):** Smallest yaw angle difference between prediction and ground-truth in radians. Orientation error is evaluated at 360 degree for all classes except barriers where it is only evaluated at 180 degrees. Orientation errors for cones are ignored.
 - **Average Velocity Error (AVE):** Absolute velocity error in m/s. Velocity error for barriers and cones are ignored.
 - **Average Attribute Error (AAE):** Calculated as $1 - acc$, where acc is the attribute classification accuracy. Attribute error for barriers and cones are ignored.
- **Relevant thresholds or criteria:**
nuScenes Detection Score
- **Evaluation method:** n/a



RESULTS

- **Results of the performance evaluation**

Measured on nuScenes Validation Split:

mAP: 0.2918

mATE: 0.6171

mASE: 0.4788

mAOE: 0.8698

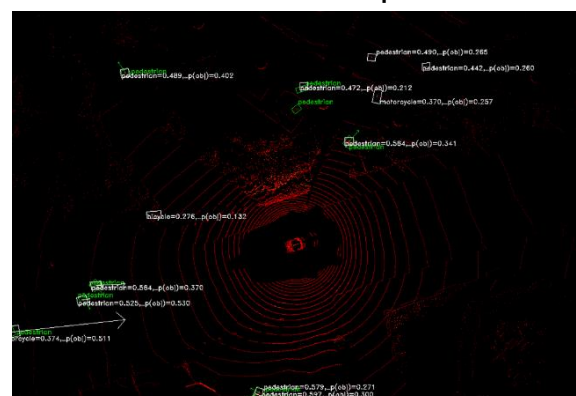
mAVE: 0.9231

mAAE: 0.5967

NDS: 0.2974

Per-class results:	AP	ATE	ASE	AOE	AVE	AAE
Object Class						
car	0.690	0.199	0.153	1.266	1.272	0.157
truck	0.491	0.420	0.270	1.135	0.344	0.433
bus	0.396	0.333	0.146	0.545	2.338	0.353
trailer	0.713	0.501	0.314	0.706	0.311	1.000
pedestrian	0.487	0.253	0.262	1.479	1.010	0.812
motorcycle	0.035	0.601	0.391	0.638	0.110	0.019

- **Performance on specific subsets of dataset:** n/a
- **Performance metrics interpretation:** n/a



ETHICAL CONSIDERATIONS



FAIRNESS

- **Fairness metrics:** No fairness measures or metrics were used to evaluate the model, and no data reweighting was performed during training. The model was not tested for fairness, and it is possible that it may exhibit bias against certain groups of people.
- **Remaining biases or limitations:** The reason is that the dataset does not contain additional meta data, which could be used for such analysis. The dataset contains only point clouds and 3D objects, but no information about certain groups or population.



PRIVACY

- **Data collecting and storing:** Private information, such as a person's identity, gender, or ethnicity, cannot be retrieved from a 3D point cloud with low resolution. This is because the point cloud is too sparse, meaning that there are not enough data points to accurately represent the fine details of the object/person.

For example, a 3D point cloud of a person's face might not have enough detail to identify their individual features, such as their eyes, nose, and mouth. As a result, it would be impossible to tell who the person is or what their gender or ethnicity is.
- **Users' access to data:** The dataset is [publicly](#) available.



ACCOUNTABILITY

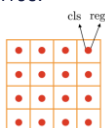
- **Model's decisions transparency through explainability techniques:** The model does not currently have any explainability mechanisms implemented. However, this feature will be implemented in the course of the project. Two potential explainability techniques that could be used are perturbation tests and attention maps. These techniques will be evaluated and implemented in the model in the future.
- **Reviewing and auditing processes:** n/a
- **Limitations or uncertainties:** n/a

USAGE AND LIMITATIONS



USAGE AND RECOMMENDATIONS

- **Model usage instructions:**
The model is designed to be used in an Automated Driving ROS2 stack. A sophisticated point cloud preprocessing pipeline is used to preprocess the input data. Is it not possible to run the model as a standalone model. Instruction to run the model can be found in the ROS2 Inference Node repository (access for Athena members available).
- **Input data format and preprocessing:**
 - Point Cloud Array of a VLP32-C or similar
 - Fields: (X, Y, Z, Ring, Intensity)
 - The coordinates must be in the LiDAR Frame
 - Max. 35000 Points per sample
- **Output interpretation:**
 - *The output of the model a grid representation on a predefined area around the ego vehicle (50m x 50m)*
 - *The Domain is discretised in cells (512x512) as shown in the Figure below. In each cell a prediction is made:*
 - *Occupancy: Binary Classification – if an object is present or not*
 - *Regression: Where is the object exactly located and which dimensions does it have ?*
 - *Classification: Multi-Class Classification - Which object class can be associated to the object if Occupancy is True?*




LIMITATIONS AND RISKS

- **Technical limitations:**
 - Domain Shift: A domain shift w.r.t. to the shape of vehicles might hurt the performance (North American Cars, European Cars); Special Types of vehicles (On Rails, Trains) might not be detected.
 - Different Sensor: The model might not be applicable to other sensor types, sensor configurations or sensor orientations; A similar input distribution as the train dataset is necessary.
 - Bad weather conditions (snow, rain, fog): might disturb the LiDAR sensor and thus the model might not be able to make reliable detections.
 - Small objects: Small objects have only a few data points on their surface, hence the detection becomes difficult or impossible.
- **Ethical or legal limitations and restrictions:**
 - *The model does not process any personal/private information.*
 - *The identification of personal data from the input sensor data is not possible.*
 - *It is not necessary to perform any anonymization of data prior to the input of the model.*
 - *People who use this model cannot capture sensitive data from pedestrians. The sensor data is very sparse and does not capture textures. Hence, it is almost impossible to identify pedestrians or capture personal information from them.*



Long-term pedestrian trajectory prediction model MODEL CARD

Author name/s and affiliation	Michael Stolz, Virtual Vehicle Research GmbH Ilma Okanovic, Virtual Vehicle Research GmbH Jan Ahmetspahic, Virtual Vehicle Research GmbH
Contact details	michael.stolz@v2c2.at , ilma.okanovic@v2c2.at , jan.ahmetspahic@v2c2.at
Link/s	Virtual Vehicle Research GmbH
Affiliation logo	
Date	01/10/2023

Disclaimer



Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

INTRODUCTION

MODEL DESCRIPTION



The model is to perform long-term prediction of detected pedestrians' trajectories based on map information and the history of pedestrians' states. Its intended use is to provide one of the functionalities necessary for traffic navigation of a robo-taxi. Therefore, any subsequent information about the model should be understood within the context of a robo-taxi service.

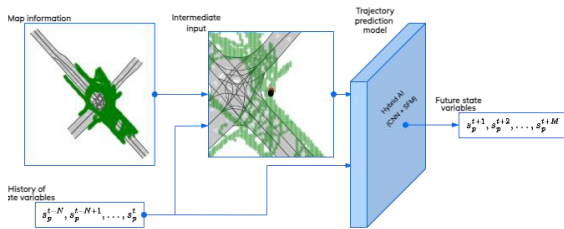


Figure 1. Prediction of future M states based on intermediate map information and history of N states (current state included).

- **Model version:** 0.0
- **Model license:** Open source
- **Citation:**

[1] F.-C. Chou *et al.*, "Predicting Motion of Vulnerable Road Users using High-Definition Maps and Efficient ConvNets," in *IEEE Intelligent Vehicles Symposium (IV)*, Las Vegas, NV, USA, 2020, pp. 1655–1662. doi: 10.1109/IV47402.2020.9304564.

[2] H. Cui *et al.*, "Deep Kinematic Models for Kinematically Feasible Vehicle Trajectory Predictions," in *IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020, pp. 10563–10569. doi: 10.1109/ICRA40945.2020.9197560.

- **Repository link:** [Long-term pedestrian trajectory prediction model](#)

MODEL DETAILS

MODEL ARCHITECTURE



- **Model type:** Hybrid AI model consisting of data-based (CNN) and physics-based (SFM) components.
- **Model task:** Regression
- **Number of layers:**
- **Number of nodes:**

- **Inputs:** A 250x250 RGB image (intermediate input) of the map with encoded information about the pedestrians' potential regions of movement and states s_p^t up to a certain time horizon of width N .
- **Outputs:** Future M states of the detected pedestrian.

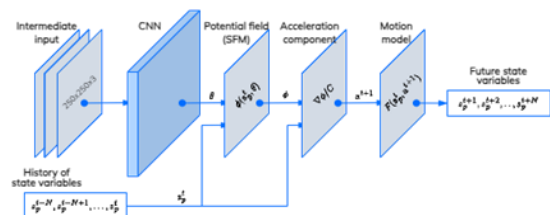


Figure 2. Hybrid AI model architecture consisting of convolutional neural network (CNN) and social force model (SFM)

MODEL DETAILS



TRAINING DATA

- **Dataset description:**
 - The dataset consists of ~500k .png images representing intermediate input generated based on three different intersection locations. Each pedestrian tracked in images has its ground truth states stored in .pkl files ready to be used for loss calculation.
 - The original recordings are part of the publicly available inD dataset. For more information on the inD dataset refer to its accompanying data card.
- **Data splitting:**
 - The intermediate input images were split into training and validation sets by random sampling. However, the link to their corresponding GT trajectory file was preserved.
- **Data augmentation techniques:**
 - Birdseye view of the traffic scene is contained within a 250x250 pixels RGB image.
 - For each timestamp, an image placing the tracked pedestrian at its centre will be generated. The map information will be restricted to 200 m in each direction of the frame axes relative to the frame centre.
 - The encoding of the map and state information is in accordance with the CommonRoad module and schema.

EVALUATION DATA

- **Dataset description:**
 - The dataset consists of ~100k .png images representing intermediate input generated based on the fourth remaining intersection location recorded in the inD dataset. Same as for the training data, the GT trajectories corresponding to the pedestrians tracked in images are stored in .pkl files.
 - The same data preprocessing steps described for the training dataset are performed for the testing dataset as well.

INTENDED USE

USE CASES AND USERS



- **Intended use cases:**
 - Trajectory prediction of vulnerable road users (VRU), specifically pedestrians.
 - Complex dynamic traffic situations common for urban roads and road segments such as intersections.
 - ODD: high visibility, absence of extreme weather conditions, smooth and even roads, no occlusion of pedestrians, urban inner-city roads.
 - The model aids in efficient traffic navigation of the robo-taxi all while ensuring the safety of all its primary (passengers) and secondary users (detected pedestrians).
- **Target audience or users of the model:**
 - The primary users of the model are pedestrians requiring a taxi service within a city in Germany or Austria.
 - The primary users do not require specialised expertise. However, all users should be aware that the robo-taxi relies on the traffic regulations that the users are expected to adhere to.
- **Users to be analysed by the model:**
 - The secondary users are the pedestrians whose trajectories will be predicted by the model.



LIMITATIONS

- **Limitations or restrictions of the intended use:**
 - The model can only be used with inputs formatted according to the specifications outlined in the data card.
 - The scenarios depicted in the input images must align with the assumptions stated in the ODD.
 - The adopted traffic participant class cannot be generalized to the excluded traffic participants.
 - Ethical limitations could be caused by the lack of representation of disabled individuals or children in the training dataset.
 - The model will not be relying on any information that is not already publicly available and will not be transmitting inputs or outputs beyond the scope of the system.
- **Guidance on how to address limitations or restrictions:**
 - Extension of the ODD of the future versions of the model.
 - Inclusion of new classes of traffic participants requiring the extension or generalisation of expert knowledge embedded in the model.
- **Identification of potential biases:**

EVALUATION AND PERFORMANCE



METRICS

- **Performance metrics:**
 - The accuracy measure will be based on the error between the ground truth and the predicted trajectory of each detected pedestrian.
 - Performance measures that quantify how well the predicted trajectories comply with the physical laws incorporated into the model. Additionally, the comparison will be done with respect to the motion models of varying complexity.
- **Relevant thresholds or criteria:**
 - The model's performance will be compared to the results stated in relevant literature including those proposing solutions solely data- or physics-based.
- **Evaluation method:**
 - The chosen evaluation method is cross-validation.



RESULTS

- **Results of the performance evaluation:**
- **Performance on specific subsets of dataset:**
- **Performance metrics interpretation:**

ETHICAL CONSIDERATIONS



FAIRNESS

- **Fairness metrics:**
 - Equal opportunity will be used to measure individual pedestrian subclass fairness*.
 - Statistical parity difference will be used to measure group fairness.
- **Mitigation of potential biases:**
 - Mitigation of mentioned biases should be addressed at the dataset preprocessing stage.



PRIVACY

- **Data collecting and storing:**
 - Once the perception system detects a pedestrian estimating their current state, such data is transferred to the GDB.
 - With every iteration of trajectory prediction, the GDB will be queried for the map information and the pedestrian's state values.
 - It is intended to store only the data that will increase the accuracy of the trajectory prediction. Therefore, only the information up to a certain time horizon will be stored and utilized for prediction.
 - Data will not be shared with third parties and will only be used within robo-taxi's navigation system.



ACCOUNTABILITY

- **Model's decisions transparency:**
 - To increase explainability, a hybrid of data- and physics-based models was defined. Using such a model implies that the embedded physical laws enforce learning compliance with them.
- **Reviewing and auditing:**
 - A method allowing users to express their concerns should be defined at the system level.
- **Limitations or uncertainties:**
 - It is not expected of users to have the expertise necessary to understand physical laws. However, the model still depends on the knowledge it acquires through intermediate input, which could be the source of the "black box" behaviour observed in traditional ML models.

*The fairness term is understood within its broader context encompassing ethical concerns.

USAGE AND LIMITATIONS



USAGE AND RECOMMENDATIONS

- **Model usage instructions:**
- **Input data format and preprocessing:**
 - PNG format will be used for the enriched map image, while the pedestrian state information (numerical) will be stored in a JSON format.
- **Output interpretation:**
 - Pedestrians' trajectory is predicted according to a defined precision threshold sampled at a specified sampling rate.
 - In the robo-taxi's navigation system, it is intended that the output of the prediction model be used for decision-making. After a decision on the future action is made, a control system will force the vehicle to continue navigating through the traffic.



LIMITATIONS AND RISKS

- **Technical limitations:**
- **Ethical or legal limitations and restrictions:**
- **Guidance on how to address limitations or restrictions:**
- **Risks identification:**