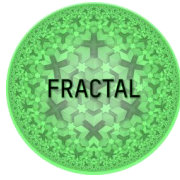


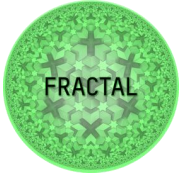
D5.1 Specification of AI methods for use case applications

Deliverable Id:	D5.1
Deliverable name:	Specification of AI methods for use case applications
Status:	Final
Dissemination level:	Public
Due date of deliverable:	2021-08-31 (M12)
Actual submission date:	2021-08-31 (M12)
Work package:	WP5 "AI and Safe Autonomous Decisions"
Organization name of lead contractor for this deliverable:	Rulex Innovation Labs
Authors:	<p>Enrico Ferrari, RULEX Christina Schwarz, AVL Debora Short, AVL Harisyam Manda, AVL Daniela Angela Parletta, MODIS Marco Cappella, MODIS Luciano Bozzi, MODIS Cristina Ganado Arteaga, PROINTEC David Sanz Muñoz, IFT Kevin Villalobos, IKER Fernando Eizaguirre, IKER Juan Manuel Besga, IKER Andoni Angulo, ZYLK Alfonso González, ZYLK Sergio Martín, ZYLK Iñigo Angulo, ZYLK Alexander Flick, PLC2 Stefano Delucchi, AITEK Nadia Caterina Zullo Lasala, ROT Damiano Vallocchia, ROT Jacopo Motta, AITEK Alberto Carlevaro, AITEK Martín Rivas, INDRA</p>



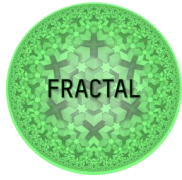
Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

Reviewers:	Stefano Delucchi, AITEK Alexander Flick, PLC2
<p>Abstract: D5.1 introduces a first description of the AI methods and tools to be used for use case applications. It starts from the functional and non-functional requirements of the use cases to define which are the methods and the tools to be employed. This deliverable constitutes a first step for defining the AI platform for FRACTAL. The next WP5 deliverables (in particular, D5.6) will continue this activity collecting use cases specifications in a more mature phase and benefitting from the work done by Task 5.1 for the definition of the theoretical bases of the FRACTAL AI.</p>	

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

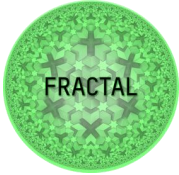
Contents

1	History	6
2	Summary	7
3	Introduction.....	8
4	AI methods overview	9
4.1	Learning vs Inference	9
4.2	Centralized/Decentralized Learning.....	11
4.3	Context Awareness.....	13
4.4	Training paradigms.....	15
4.5	Model Compression	17
4.6	Weight quantization.....	18
4.7	Model Lifecycle Management (MLOps)	19
4.7.1	Machine Learning Deployment Workflow	20
4.7.2	Continuous Delivery for Machine Learning	22
4.7.3	End-to-end ML Process	23
5	UC functional and non-functional requirements	24
5.1	Use Case 1	24
5.2	Use Case 2	27
5.3	Use Case 3	28
5.4	Use Case 4	29
5.5	Use Case 5	30
5.6	Use Case 6	33
5.7	Use Case 7	34
5.8	Use Case 8	36
6	Proposed AI methods.....	38
6.1	Video analysis using Convolutional Neural Networks	38
6.2	Yolo.....	43
6.2.1	YOLOv1.....	43
6.2.2	YOLOv2.....	47
6.2.3	YOLOv3.....	48
6.2.4	YOLOv4 and Tiny-YOLO.....	49



Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

6.3	Audio Stream Analysis	50
6.4	Reinforcement Learning	52
7	Proposed AI tools, frameworks and standards	56
7.1	Tensorflow.....	56
7.2	Keras.....	57
7.3	Apache TVM.....	57
7.4	Apache MXNET	58
7.5	Pandas.....	59
7.6	Scikit	59
7.7	Stable Baselines	60
7.8	Gym	60
7.9	Pytorch	60
7.10	Caffe	61
7.11	Darknet	62
7.12	OpenCV	62
7.13	Comparison among different tools	63
7.14	ONNX and ONNX Runtime	64
8	Conclusions	67
9	Bibliography	68
10	List of Figures	72
11	List of Tables	74
12	List of Abbreviations	75

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Acknowledgement

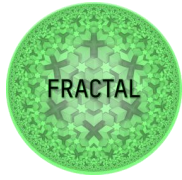


ECSEL Joint Undertaking

Electronic Components and Systems for European Leadership



This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 877056. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Italy, Austria, Germany, Finland and Switzerland.

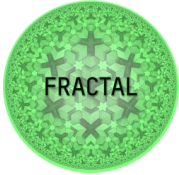


Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

1 History

Version	Date	Modification reason	Modified by
0.1	2021-07-14	First Draft – integrating first contributions	Enrico Ferrari et al.
0.2	2021-07-21	Integrating second round of contributions	Enrico Ferrari et al.
0.3	2021-07-28	Integrating third round of contributions	Enrico Ferrari et al.
0.4	2021-08-02	Document ready for review	Enrico Ferrari et al.
0.5	2021-08-25	Addressing reviewers’ comments	Enrico Ferrari et al.
0.6	2021-08-27	Some formatting changes	Enrico Ferrari et al.
1.0	2021-08-31	Document ready for submission	Enrico Ferrari et al.

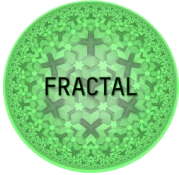
Table 1 – Document History

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

2 Summary

According to the DoA, D5.1 “Specification of AI methods for use case applications” is aimed at collecting the “specification of AI methods to provide use case applications with context awareness, prediction, and autonomous decision making, to run on the FRACTAL AI platform”. This deliverable contains the first outcomes of T5.3 “Applied FRACTAL AI” since it contains the first description of the methods that emerged as needed for AI applications in the FRACTAL use cases.

The deliverable contains a first overview about AI and the dimensions that should be considered in defining AI specifications. Then the use cases are considered one by one to gather the requirements related to AI. At last, the methods, tools and standard emerged from the use case analysis, are described.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

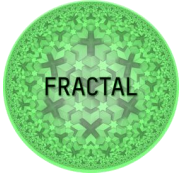
3 Introduction

This deliverable accounts for the first activities related to FRACTAL Task 5.3, regarding “Applied FRACTAL AI”. In particular, the goal of this task is ensuring that methods and workflows developed in WP5 are coherent with the objectives of the demonstrators and including in the FRACTAL AI framework all the methods and tools that are needed for a smooth implementation of AI applications in the real-world studied situations.

In particular, Deliverable D5.1, entitled “Specification of AI methods for use case applications” is aimed at providing a first description of the AI methods and tools that will be needed by the use case for implementing AI modules.

Since some AI applications in use cases are in an early stage of definition, this deliverable should be regarded as a first version of the AI methods. A more detailed and definitive version will be provided in other deliverables of WP5 and in particular in D5.6, named “Final AI methods for use case applications and mechanism for AI transparency interactions”. This deliverable will take D5.1 as the starting point and will benefit from the ongoing discussion about use cases and AI specifications. Moreover, D5.6 will also take into account the outcomes deriving from the other WP5 tasks, in particular Task 5.1 (“FRACTAL AI Theory”), which will provide guidelines about the theoretical framework behind the FRACTAL AI framework and whose deliverables will be finalized at M24.

The logical structure of this deliverable starts from a general analysis of the parameters that should be considered when deciding which solutions are to be adopted. This includes for example the distribution paradigm (centralized/non centralized), the training mode (online/offline) and the feasibility of an operation-level implementation of AI solutions. Then, use cases specifications are gathered with a specific focus on the AI methods and tools needed for use case implementation. Starting from the specifications emerged from this analysis, the needed methods and tools are described in detail. Some conclusions close the deliverable summing up the main achievements.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

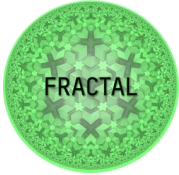
4 AI methods overview

The activities in WP5 aim at reaching the goals of Objective 3 of Fractal which consists in “evaluating and validating the analytics approach by means of AI to help the identification of the largest set of working conditions still preserving safe and secure operational behaviours”. To this aim, the knowledge of the environment should be combined with AI algorithms to obtain a Predictive, Prescriptive and Trusted edge. In this section a general overview of AI is provided, including the characteristics that should be considered and evaluated when designing an AI platform, with particular focus on the use case applications. In particular these topics will be considered: Learning vs Inference, Centralized/Decentralized Learning, Context Awareness, Cloud/Edge Training, Model Compression, Weight Quantization and Model Lifecycle Management (MLOps).

More concretely, and taking into account the FRACTAL scenario, it is important to go over topics such as Learning vs Inference (Paragraph 4.1), and the ability to separate AI functionalities (or micro-elements) to divide the learning problem into simpler tasks. Another key point is Centralized/Decentralized Learning (Paragraph 4.2), to analyse the different approaches that enable distributed learning over a variable number of agents. The strategies to orchestrate learning tasks can be taken by a single Master node, or by all nodes in a collaborative way. For this, Context Awareness (Paragraph 4.3) is crucial to sense the environment and make decisions based on its inputs. Also, AI model adaptation to particular devices is an important research line, especially when taking into account smaller nodes with low computational resources. Thus, deployment strategies will comprise Model Compression (Paragraph 4.5) and Weight Quantization (Paragraph 4.6) tasks, to be able to run models in lighter nodes. Lastly, MLOps (Paragraph 4.7) will define and control the Machine Learning system pipeline, which will be defined by the previously mentioned concepts and steps. The goal here is to obtain a reliable and resilient system that allows to control data and models, reproduce data science experiments, and permits continuous integrations and monitor the whole system performance.

4.1 Learning vs Inference

The FRACTAL AI framework is aimed at providing the FRACTAL node with tools that allow to take decisions on the edge quickly and without much computational effort. As the name says, AI methods are built with the objective of emulating the human intelligence, whose strongpoint is the ability of performing inductive reasoning, i.e. of analysing data about a phenomenon and in deriving models from them. For this reason, AI methods need a phase where the model is built starting from available data. In FRACTAL, data could be already available, or retrieved in some accessible database or collected during the project. In any case, data should be processed to extract a general model that could be applied on new situations that could happen in a real context. It is worth noting that, to ensure that the AI methods work well in (almost) all the situations, it is crucial that the available data includes a wide range of different configurations. Moreover, the behaviour of the studied system could

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

change over time, so data that have been collected in a given period, could be inappropriate for applications after some time.

In general, when talking about data-driven, a distinction is needed between the learning phase and the inferencing phase. In the first one, historical data are used to train the AI models while in the second one the already built models are used to inference and take decisions about the new data. Every time new data is provided to the learning engine, a new model is obtained which, of course, is different from the one obtained with the old data. In this case, we are saying that the model is updated.

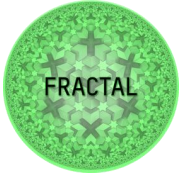
If the system's behaviour is not changing very frequently or suddenly, the training phase could be done sporadically and moreover it is not needed to have the updated model immediately since the old one is still available to take decisions.

In this case, the training phase is done once or is repeated on a regular basis if the system is supposed to undergo some changes in the behaviour. In the second case, it is important to detect the drifts in the behaviour on time. So, either the model is refreshed with a fixed frequency (high enough to catch changes) or some validation is done to check if the decision taken by the model are still valid. For example, it is possible to define some indicators that account for the accuracy of the model (e.g. the number of false negatives or false positives) and then, when one (or more) of these indicators falls below a given threshold, the training is repeated. Only if changes are so sudden that latency of the connection could be an issue or if data could not be sent due to privacy reasons, training should be performed on the edge device.

So, in general, two different scenarios could be considered:

- The training is done offline. Data are transferred manually on some repository used to build a model. Then the model is manually transferred in the FRACTAL node where it is used to make predictions on new data. The first tests with the FRACTAL node will probably follow this approach since it allows a quick implementation of the models in the device.
- The training is done periodically. In this case (see Figure 1) data are automatically transferred to a cloud service (via 5G or Wi-Fi connection) where they are stored in a database and used for building a model. Due to huge amount of data or privacy concerns it could be unfeasible or inappropriate that all the generated data are sent to a cloud system. In this case, the FRACTAL should be able to do some aggregation operations in order to do some basic processing on the data and reduce the amount of data to be sent over the network. The model is then sent back to the device where it is used for subsequent elaborations. Notice that in this case the network is not a bottleneck since FRACTAL node can continue its processing even without the new model, since the old one is still able to take decisions. So, if for some reason the connection is not guaranteed the system can go on working.

It is possible to imagine also that the training could be performed on the FRACTAL node, but usually the computational resources needed for training a model fit better

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

with a cloud environment. As some preliminary analysis showed, no use cases require update frequencies forcing to perform the training on the edge.

As regards the inferencing phase, this is performed on the FRACTAL node since the decisions should be taken very quickly, which is not compatible with sending data over the internet. Moreover, the system should work also without connection, so the node must be autonomous in taking decisions.

As studied in the AI specifications, the FRACTAL node will be equipped with a layer able to perform some basic pre-processing operations on the data and an AI module able to use already built models to make decisions about new incoming data. The elaboration must be very quick because in some use cases, decisions are expected to take place with a frequency of some Hertz (the exact frequency for all use cases is still to be defined but for most of them it could be estimated in the range 10-20 Hz).

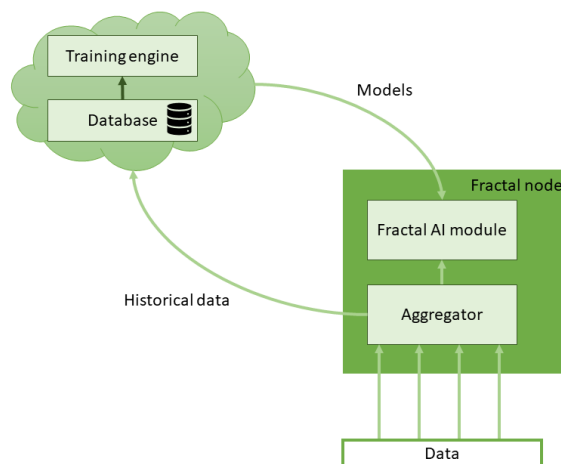
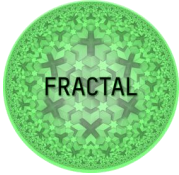


Figure 1 – Functioning of the online training for the FRACTAL node

4.2 Centralized/Decentralized Learning

The proliferation of IoT devices has opened the possibility for many different application fields to capture and share vast amounts of data over the Internet, from which valuable information and knowledge can be extracted. However, the captured data by these devices leads to so large and complex datasets that it becomes difficult to process and extract knowledge from such “Big Data” with traditional techniques. Consequently, the application of machine learning techniques to the data captured by these IoT devices, has aroused an increasing interest during the last years. On these techniques a machine learning model is *trained* to find patterns and underlying data structures over a known dataset to *learn* to find those patterns over new (unseen) datasets and make inference or predictions.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Standard machine learning approaches require sending the captured data by these IoT devices to a single processing node, usually in the cloud, where data is centralized into a common dataset, on the top of which different machine learning models are built and trained. Predictions are also usually made in the cloud by using *streaming* data that the nodes send, although the nodes can download the global model from the cloud and made the inference locally. This approach, known as *centralized learning*, can be seen on the left part of Figure 2. In this approach, the machine learning models are trained with data from all the available nodes, and thus, usually they can generalize properly not only for all the known nodes, but also for other new (unknown) nodes that are similar to the known ones.

However, it is worth mentioning that this generalization becomes harder to ensure when there is a large variety of nodes (as in IoT scenarios). Moreover, collecting data from such distributed nodes in a centralized dataset has become more and more problematic in the last years, for two main reasons: on the one hand, the data captured by these nodes could be sensitive or confidential and therefore, should remain on-site. This along with the novel data protection policies, and in general the increasing public awareness to issues related to data handling, makes privacy one of the main drawbacks of centralized learning. On the other hand, the limitations and costs for the communication of large amounts of data (e.g., latency, bandwidth) are out of alignment with the increasingly demands of low latency and real-time or stream data processing requisites from the application domains. Regarding this last aspect, it is worth mentioning that new technologies, such as 5G, are emerging to cope with bandwidth and latency limitations. Moreover, latency and bandwidth may represent a limitation only for the inference phase and not for training.

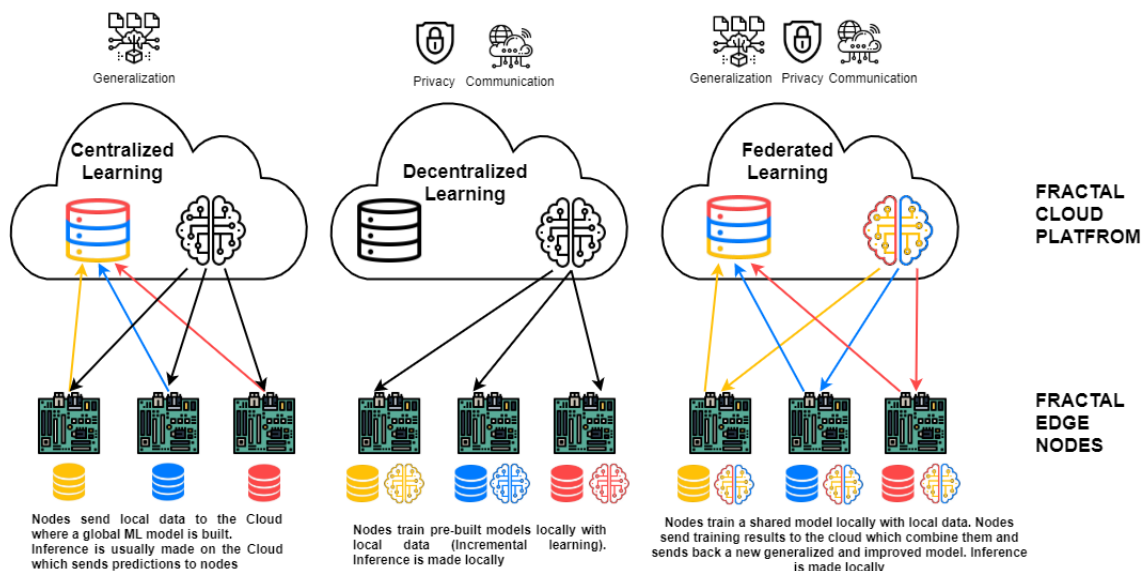
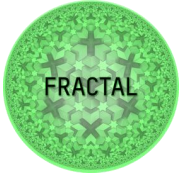


Figure 2 – Learning Approaches in FRACTAL: (left) Centralized Learning; (centre) Decentralized Learning; (right) Federated Learning.

Consequently, recently different efforts are advocating for the decentralization of this approach by locally training machine learning models on various decentralized nodes holding their own local data. This approach, known as *decentralized learning*, can be

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

seen on the central part of Figure 2. In this approach, each node builds its own machine learning model which is trained with the data captured by the node (locally). On the one hand, training models with local data, mitigates some of the risks related to data privacy (critical data remains on-site). On the other hand, using locally trained models for inference, reduces the data transfer rates between the nodes and the cloud, optimizing communications and decreasing the latency for real-time applications and stream data processing.

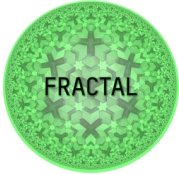
In addition, training models with local data, usually leads to smarter models with better performance, since, as they do not need to learn underlying data structures and patterns from a large dataset from different nodes, each of them with their peculiarities, they could focus more on the specificities of their own local data. However, this specialization, reduces the models' capability for generalization, which usually achieve a poor performance even when applied on similar nodes (new nodes). Another option is to use a pre-built model which has been trained with a common dataset from the different nodes and then, deploy it on-site to train it with local data (*incremental learning*).

Finally, in the last years, federated learning (Li, Sahu, Talwalkar, & Smith, 2020), (Abdulrahman, et al., 2021) and (Zhang, et al., 2021) has emerged as an interesting approach on which nodes learn collaboratively from a shared model while keeping their own training data locally. The shared model is first trained in a centralized fashion on a server using a large-scale centralized dataset and then, the distributed nodes download the model and improve it by using their own local data (federated data). Eventually, nodes send models related data (such as performance indicators, weights, parameters, etc.) to the *centralized* node where are combined with the shared model, to improve the overall performance of the models. Then, this shared model is sent back to the distributed nodes where it could be fine-tuned again with local data. This way, federated learning ensures to keep the generalization capabilities of models built over a large-scale dataset, while keeping the privacy of sensitive (and critical) data and a low latency for real-time predictions or stream data processing.

In FRACTAL, the three approaches should be considered to meet the requirements of a variety of machine learning models for the different use cases. On the one hand, the decentralization of machine learning models leads to smarter models (which better fits to the specific requirements of a particular node), lower latency, and less power consumption. On the other hand, for more complex models requiring large-scale data from different sources and high computing capabilities, it may be necessary to train the models on the cloud and then deploy them on the edge devices (nodes). Moreover, federated learning will be also considered for some use cases (e.g., intelligent totems in UC6) on which models need to learn collaboratively to solve complex tasks or there are stringent privacy constraints.

4.3 Context Awareness

Context Awareness: Definition

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

The FRACTAL AI Node will be implemented with high-efficiency AI algorithms and change adaptation methods. These “changes” refer to any possible modification or deviation in the node’s environment, i.e., number of active nodes, computing processes, resources availability... and they need to be addressed in an optimal and efficient way. Context awareness is therefore a required feature in the FRACTAL Node, as it enables other key aspects which ultimately affect energy consumption and resource availability. The node must be able to know its surroundings and environment and also to act accordingly for optimal performance.

For humans, context awareness is assumed to be something happening “in the background” of brain activity, take for example two people having a conversation: They both are able to answer questions not related to the conversation topic but the environment and surroundings, e.g., temperature, time, space location, object presence...

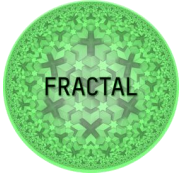
When dealing with computer devices, programming an application to act accordingly to external changes is a non-trivial task. The main field of application of context awareness in computer systems is in improving user experience through recommendations or adaptations to the application user’s context. Context awareness in computing systems has been traditionally defined as “Using context to provide relevant information and/or services to the user, where relevance depends on the user’s task” (Dey, 2000). While this definition is correct for general context-aware applications, the concept must be expanded for the FRACTAL platform, as the context information is not going to be used directly by the user, but the platform itself. Then, the FRACTAL Node being context aware implies that it will have a detailed notion of the environment surrounding it so that it is able to perform appropriate actions and adapt to environmental changes. This information about the environment and adaptative responses will be handled by AI algorithms to maximize the efficiency of processes and minimize the impact coming from external sources.

Context Awareness in AI Applications

Once our applications are able to run on a context-aware device, the application’s *behaviour* (outputs, actions...) must be different for any given context, while discriminating between relevant context knowledge and taking into account this knowledge to adapt the service to be fully aware of the context and give a context-dependent answer.

Some of the actions the FRACTAL AI Node is able to perform include using ML methods for edge inference and running AI algorithms, so interpretation, learning and predictions will be performed continuously throughout the nodes. Being context aware allows the node to perform the tasks in a **contextualized manner**, such that energy consumption, resource usage and latency are minimized, while also being able to adapt to external changes to keep these parameters optimal during the whole application lifecycle.

FRACTAL Context Awareness

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

The proposed FRACTAL Context Awareness expands some of the concepts of classical context awareness in applications to fulfil the platform requirements and ensure optimal performance. Firstly, the FRACTAL Context Awareness leaves aside the “user context” and focuses on “environment context”. The reason for this is that the target of the information extracted from the context is not the end user, but the node itself.

In addition, several constraints have already been applied to the FRACTAL Node, like minimal energy consumption and resource-limited capabilities, which are intrinsic limitations to Edge architectures. Note that these context awareness features will not interfere with other applications and processes running in the node. For this purpose, the applied methods for resource-constrained architectures will ensure that context awareness processes and further adaptation will be run in a fault-tolerant and robust architecture, while consuming low resources.

Implementation

The main feature a context-aware computing system must provide is supporting the rest of functionalities. Efforts must be made in this way to ensure that a significant overall performance boost results from the addition of context awareness capabilities. The node must be able to discover, process and take advantage of any available context information. Context information can potentially be: (1) any parameter that can be obtained through a sensor, or any information that can be derived from the sensor's data processing, (2) information coming from other Edge nodes, like position, resource capabilities, status reports, etc. However, specific context information to be collected may not be available yet for all Use Cases, given the heavy reliance this information has on the specific Use Case scenario.

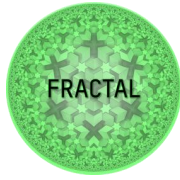
The implementation of Context Awareness capabilities into the FRACTAL Node will be done through two agents, the context agent and the actuator agent, the first will be responsible for collecting and processing the context data from sensors or other nodes and processing them to obtain context-defining information. The latter will receive this information as input and perform an analysis on the actions to decide which state of the system will best fit the performance requirements. This information will later be taken as an input by T6.2 for Node adaptation actions.

4.4 Training paradigms

In the machine learning world, model training refers to the process of allowing a machine learning algorithm to automatically learn patterns based on data.


There are four main categories of Machine Learning, as shown in Table 2.

Supervised Learning	Supervised Learning build a model from labelled training data, with which is possible to make predictions on unavailable or future data. Supervision therefore means that in a set of samples (or datasets), the desired output signals are already known since they have been previously labelled. In this type of learning, based on
---------------------	--



Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

	<p>discrete class labels, it is possible have a task based on classification techniques.</p> <p>Classification is a technique used in supervised learning where the goal, based on the analysis of previously labelled data, is to be able to predict the labelling of future data classes. Labels are unordered discrete values that can be considered to belong to a group of a class.</p> <p>Another type of technique used in supervised learning is regression, where the output signals are continuous values. In Regression, you have a number of predictive (descriptive) variables and one continuous target variable (the result). In this type of problem, we try to find a relationship between these variables in order to predict a result. Given a predictive variable x and a response variable y, a line is drawn in order to decrease the distance between the points and the line itself. Taking the slope and the intersection point as a reference, a target variable can be predicted from this data to be used as a reference for new data.</p>
Unsupervised Learning	<p>In unsupervised learning, unlike supervised learning, there are unlabelled data or unstructured data. Using these techniques, it is possible to observe the structure of the data and to extrapolate information loaded with meaning but, however, you cannot rely on a known variable relating to the result or on a reward function.</p> <p>Clustering is an exploratory technique that allows you to aggregate data within groups (called clusters) for which we have no prior knowledge of belonging to groups. It will be produced large datasets where the data within them have similar elements to each other. Within each individual group (or cluster) we will therefore find data that have many similar characteristics.</p> <p>The Reduction of Dimensionality without supervision is a widely used approach in the pre-processing of features (characteristics), with the aim of eliminating the "noise" from the data. This reduction can also cause lower predictive performance, but it can also make the dimensional space more compact in order to keep the information more relevant.</p>
Semi-supervised learning	<p>Semi-supervised learning is an approach to machine learning that combines a small amount of labelled data with a large amount of unlabelled data during training. Semi-supervised learning falls between unsupervised learning (with no labelled training data) and supervised learning (with only labelled training data). It is a special instance of weak supervision.</p>

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Reinforcement Learning	The goal of this type of learning is to build a system (agent) that improves its performance through interactions with the environment. In order to improve the functionality of the system, reinforcements are introduced, i.e. reward signals. This reinforcement is not given by labels or correct truth values but is a measurement of the quality of the actions taken by the system. For this reason, it cannot be assimilated to supervised learning.
------------------------	--

Table 2 – Different categories of algorithms in Machine Learning

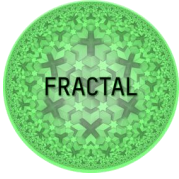
4.5 Model Compression

The goal of model compression is to obtain a *compressed* model from a given input model without significantly diminished accuracy. A compressed model is one that has a smaller size and/or latency than the original. Specifically, size reduction means that the compressed model has fewer and/or smaller parameters and, thus, requires less memory to be stored and used. Similarly, a reduction in the latency implies smaller prediction complexity (i.e. the time taken to make a prediction for a given input).

Both reductions are desirable in the context of typical IoT devices as such devices have limited memory and low compute power. Such constraints heavily undermine the applicability of powerful learning models such as deep networks (DN) or Nearest Neighbour (NN) classifiers. Those models, while attaining state-of-the-art performances features thousands of parameters, either in terms of the network weights or in terms of number of data points to memorize (respectively for DNs and NN methods). These parameters result ultimately in high-memory demands. In addition, NN neighbours – that in principle can obtain the optimal performances of the Bayes predictors – also have the disadvantage of high prediction complexities. This results either in high CPU-demand or in a high energy consumption. Making smaller and faster models is then a key challenge in order to make them applicable in the IoT context.

There are several techniques that can be used to compress a machine learning model:

- **Pruning** reduces the number of parameters, for example by removing redundant and unimportant connections in a neural network, cutting branches in a decision tree, or reduce the training set size in a NN classifier. This not only helps reduce the overall model size but also saves on computation time and energy.
- **Quantization** (for more details, see section 4.6)
- In **knowledge distillation**, a large, complex model is trained on a large dataset. When this large model can generalize and perform well on unseen data, it is transferred to a smaller network. The larger model is also known as the teacher model and the smaller model is also known as the student model.
- **Selective attention** is the idea of focusing on objects or elements of interest, while discarding the others (often background or other task-irrelevant objects). It is inspired by the biology of the human eye. When we look at

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

something, we only focus on one or a few objects at a time, and other regions are blurred out. This requires adding a selective attention model upstream of your existing AI system or using it by itself if it serves your purpose. It depends on the problem you are trying to solve.

- **Matrix/tensor decomposition** is used to estimate the informative parameters. A parameter matrix A with $m \times n$ dimension and having a rank r is replaced by smaller dimension matrices. This technique helps by factorizing a large matrix into smaller matrices.
- Methods to reduce the prediction complexity, instead tend to be more specific. As an example, for NN methods smaller prediction complexity can be reduced by performing approximate NN searches in place of exact searches.

It is worth noting that all the above techniques are complementary to each other, so they can be applied as is or combined with one or multiple techniques. Most of the techniques discussed above can be applied to pre-trained models, as a post-processing step to reduce your model size and increase inference speed. But they can be applied during training time as well.

4.6 Weight quantization

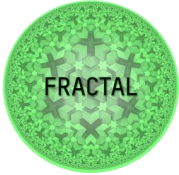
Quantization is the process of mapping values from a large set to values in a smaller set, meaning that the output consists of a smaller range of possible values than the input, ideally without losing too much information in the process.

In modern Deep Neural Network (DNN), weights are usually stored as 32-bit floating-point numbers. In this context, weights quantization consists in designing methods that represent these weights on 16-bits, 8-bits, 4-bits or even with 1-bit. The overall effect of weights quantization is that the size of the deep neural network can be significantly reduced. Of course, the quantization needs to be performed in a way that the accuracy of the resulting network is not much worse than the accuracy of the original one. Quantization is in principle enabled by the fact that most real-world datasets are noisy and modern DNs usually achieve 0 error. This mean that the networks are also learning the noise. At high level, the noise information is contained into the less significant digits of the weights. The effect of weights quantization then it to mostly remove the noise effect from the parameter.

Similarly, quantization may also be applied to the output of the network operators, such as the activation functions and the autoencoders to mention a couple.

Regardless of the object being a weight or an operator, there are several approaches to quantization. The principal features to classify a quantization algorithm can be summarized as:

- **Symmetric vs Asymmetric.** A quantizer whose co-domain is symmetric about the origin is called symmetric, otherwise is said to be asymmetric. Symmetric quantizer are appropriate when the variable being quantized has an empirical distribution close to the uniform over the quantization range $[-a,$

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

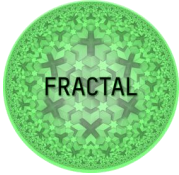
- $a]$, with $a > 0$. On the other hand, if the empirical distribution is supported on a general interval $[a, b]$, then an asymmetric quantizer is more appropriate.
- **Static vs Dynamic.** If the quantization range $[a, b]$ is determined beforehand by the user, then the quantizer is said to be static; otherwise, the quantizer is said to be dynamic. Specifically, a dynamic quantizer will determine the appropriate quantization range based on the data to quantize. There is an obvious trade-off in these cases: dynamic quantizers are usually more accurate than static quantizer, while requiring more computations to be calculated.
 - **Granularity.** A single quantization scheme may be employed for the entire architecture, or a different quantization scheme can be employed for each layer or even kernel in a network. This design choice determines the granularity of the quantization. In this context the quantization algorithm itself is assumed to be fixed, but its parameters can be chosen on the basis of the fixed granularity.
 - **Uniform vs non-Uniform.** A quantizer whose co-domain is made of uniformly spaced values is called uniform, otherwise is said to be non-uniform. Uniform quantizer are appropriate when the variable being quantized has an empirical distribution close to the uniform over the quantization range $[a, b]$. On the other hand, if the empirical distribution is skewed, then a non-uniform quantizer is more appropriate.
 - **Quantization-Aware Training.** Quantization can be performed either as a post-training procedure or as part of the training; in the latter case the training is said to be quantization-aware. At high level, such methods require, at each step of the training, a projection of the floating-point weights into the quantization lattice. In order to avoid numerical instability, due to the vanishing gradient phenomenon, some care in the projection needs to be reserved. Quantization-Aware training, when performed correctly, leads to superior performance, but typically also requires more computations.

Finally, there are more advanced techniques that are emerging and whose performances are not be completely established yet. Those include **Zero-Shot quantization** which perform quantization without looking at the data; and, **Stochastic Quantization** per the quantization algorithm itself adopts some internal randomness.

4.7 Model Lifecycle Management (MLOps)

The term MLOps (also known as AIOps) is the application of DevOps concepts and techniques to Machine Learning (ML) systems. DevOps includes the techniques and tools required to successfully maintain and support existing production systems, and it is commonly used by IT professionals in the present.

Machine Learning systems have increased complexity compared to other traditional data architectures within the field. This is due to several factors, such as multidisciplinary team collaboration, extra steps on lifecycle management including data collection and quality assurance, etc. Moreover, the evolution from a purely

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

academic research area to an applied field is recent. Because of all this, ML system integration on production environments is still a strong research area, especially in the definition of unified methodologies.

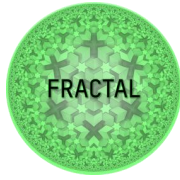
4.7.1 Machine Learning Deployment Workflow

At the present, there exist different approaches for MLOps systems. This is due to a lack of unified methodology for MLOps solution. However, most of them share a common view of “a multiple step workflow that manages data, models and code”. To describe these steps, four main stages are proposed which involve particular tasks: data management, model learning, model verification and model deployment (Paleyes, Urma, & Lawrence, 2020).

The **Data management** phase focuses on the importance of data and quality assurance, which is then used to train the models, and it is generally the first stage in the ML pipeline. Tasks such as data collection, pre-processing and augmentation are included here, to avoid biases and behaviour inconsistencies in the final result. Then, **Model learning** involves the stage in which the model actually learns about the problem to solve. Some key steps to take into account are model selection (trade-off between model complexity and productive necessities), training and Hyperparameter selection (decision-making for minimizing these expensive and resource-heavy practices).

Model verification includes multiple aspects to bear in mind in order to verify a model, such as requirement encoding (technical viability and business gaining), formal verification (use-case) and test-based verification (with new data). Finally, the **Model deployment** stage involves the tasks to set a Machine Learning system into a productive environment. These will include integration (supported by existing infrastructure), monitoring and updating (regular retraining, continuous learning).

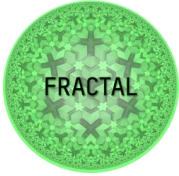
Table 3 offers a more detailed view of the previously defined stages, as well as the tasks included in each of them, and main issues and considerations regarding these ones. The table shows what issues are most commonly found at each stage of the workflow.



Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

Deployment Stage	Deployment Step	Considerations, Issues and Concerns
Data management	Data collection	Data discovery
	Data preprocessing	Data dispersion Data cleaning
	Data augmentation	Labeling of large volumes of data Access to experts Lack of high-variance data
	Data analysis	Data profiling
Model learning	Model selection	Model complexity Resource-constrained environments Interpretability of the model
	Training	Computational cost Environmental impact
	Hyper-parameter selection	Resource-heavy techniques Hardware-aware optimization
Model verification	Requirement encoding	Performance metrics Business driven metrics
	Formal verification	Regulatory frameworks
	Test-based verification	Simulation-based testing
Model deployment	Integration	Operational support Reuse of code and models Software engineering anti-patterns Mixed team dynamics
	Monitoring	Feedback loops Outlier detection Custom design tooling
	Updating	Concept drift Continuous delivery
Cross-cutting aspects	Ethics	Country-level regulations Focus on technical solution only Aggravation of biases Authorship Decision making
	End users' trust	Involvement of end users User experience Explainability score
	Security	Data poisoning Model stealing Model inversion

Table 3 – Considerations, issues and concerns explored in (Paleyes, Urma, & Lawrence, 2020).

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

4.7.2 Continuous Delivery for Machine Learning

Over the previously defined MLOps stages, model deployment and integration on productive environments have gained great attention lately. One of the most researched topics is Continuous Delivery for ML, as it is a complex task that involves three different domains: the data, the code and the model (Paleyes, Urma, & Lawrence, 2020). This concept can be defined as “a software engineering approach in which a cross-functional team produces machine learning applications based on code, data, and models in small and safe increments that can be reproduced and reliably released at any time, in short adaptation cycles” (Sato, Wider, & Windheuser, 2019).

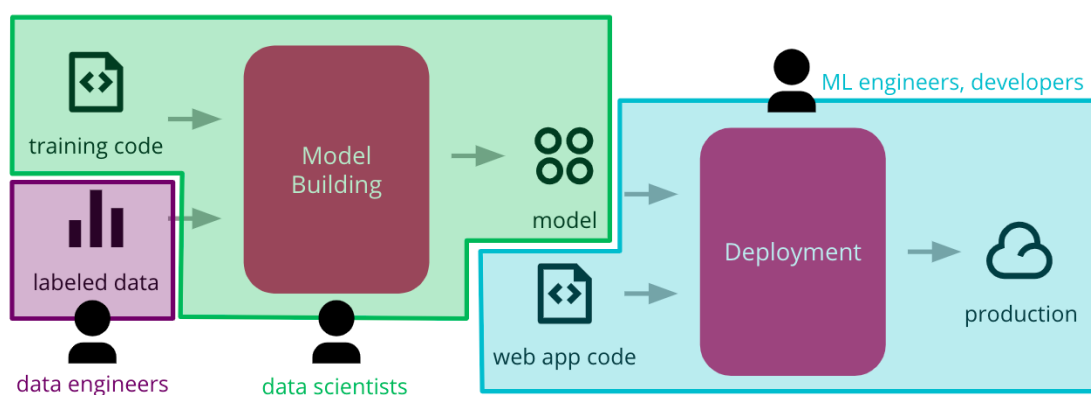
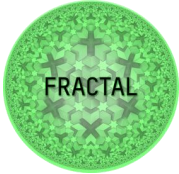


Figure 3 – Common functional silos in large organizations can create barriers, stifling the ability to automate the end-to-end process of deploying ML applications to production. Picture taken from (Sato, Wider, & Windheuser, 2019).

Common challenges when building a ML system include multidisciplinary team collaboration, process reproducibility and auditability. The heterogeneity of teams and abilities required during the workflow lead to problems that are illustrated in Figure 4. To overcome these challenges, different strategies and tools can be found nowadays as valid alternatives. As previously mentioned, there is not a unified methodology for building a MLOps system at the present. Current solutions are intended to tackle particular aspects of the ML workflow. To describe some of them:

- **Data Discoverability and Accessibility** are often based nowadays on Big Data architectures such as Data Lakes (or Data Mesh) are proposed, which allow to provide data over its whole lifecycle.
- Another key aspect is the **Reproducibility of Model Training**, and its necessity to be reproducible and version-controlled, similarly to traditional code. There exist different tools to achieve this, such as Data Science Version Control (DVC), Pachyderm, MLFlow.
- **Model Serving**, for which several tools and frameworks can be found, such as PMML and ONNX (for embedded models); and Apache Submarine and MLFlow (for model as a service approaches).

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

4.7.3 End-to-end ML Process

The FRACTAL MLOps system should accomplish all the previously described phases. At the moment, different strategies are being studied for achieving this. Also, different tools and frameworks are being tested in order to offer the best alternatives to deal with the system necessities.

The End-to-end Machine Learning Processes aim to reach a unified and standard methodology to cover all the phases of the ML models lifecycle, from the design of models to the final delivery and deploy. In the perfect scenario, each of the phases would be covered by a well-defined and robust software stack that is able to build, train, deploy, and maintain the models into a production environment without the need of very specific frameworks for each phase. Figure 4 is a good example of an overview of MLOps target system, in which the main phases and artifacts can be seen.

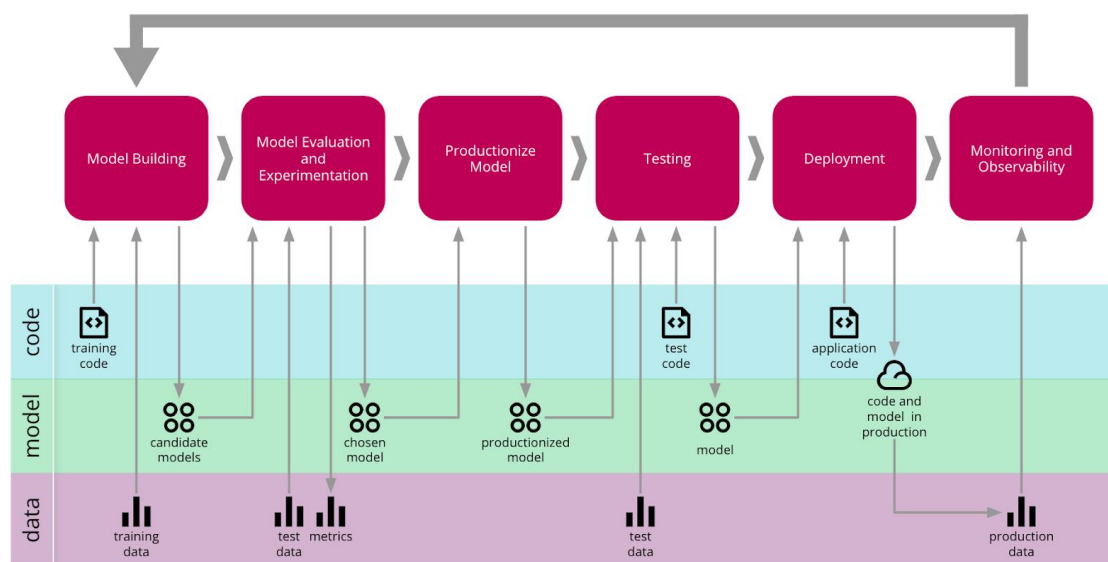
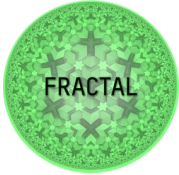


Figure 4 – Continuous Delivery for Machine Learning end-to-end process (Sato, Wider, & Windheuser, 2019).

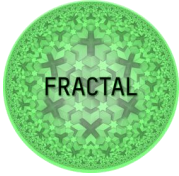
	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

5 UC functional and non-functional requirements

5.1 Use Case 1

Use Case 1 includes two demonstrators exploiting two rather different scenarios and AI applications. Table 4 and Table 5 report the requirements for the two demonstrators.

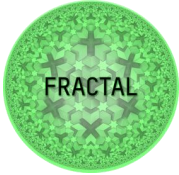
UC1 (Demonstrator 1): UAV supervision of critical structures	
Short Description of the Use Case	This demonstrator will be focused in the supervision of critical structures as bridges or viaducts, where images of the structural status will be collected through the use of UAVs, systematizing the visual inspection in near-real-time to detect failures and cracks in the concrete surface of the structure
AI Application(s)	The algorithm used to extract information from the images is based on a deep learning model with convolutional neural networks (CNN). In order to solve the segmentation problem, the architecture of the deep neural network is U-Net (and ResNet), with a custom loss-function (combination of Categorical Cross entropy and IOU).
Expected outcome	It is expected to obtain the pixel-wise segmentation of the cracks with high accuracy. The segmented images will be inspected by an operator, so the real hazard that the cracks may pose can be correctly inferred.
Input Variables	Video frames.
Algorithms to be used	<ul style="list-style-type: none"> • Deep Neural Network with U-Net architecture. • Image augmentation: translations, rotations, ... • Image augmentation: superposition of texture images with the raw images of cracks (alpha compositing). • Opening, closing and double threshold to close the gaps in-between the cracks.
Training vs Inference	The training phase will be done offline. The training dataset is composed by collected images of cracks on concrete surfaces

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

	<p>and augmented images (translations, rotations, superposition of different textures, ...).</p> <p>The inference is done online and in real-time on the UAV.</p>
Tools / frameworks to be used	The programming language is Python, and the libraries that we will use are TensorFlow 2.0, OpenCV and NumPy.
Required hardware for AI applications	Device or platform with GPU and enough computing power capabilities (i.e. Jetson Nano, Raspberry pi+Coral board or Versal platform).
Expected performances	Still to be defined.

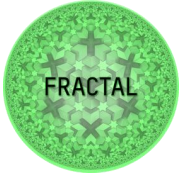
Table 4 – Requirements for Use Case 1 – Demonstrator 1

UC1 (Demonstrator 2): Wireless Sensor Network (WSN) for safety at construction sites	
Short Description of the Use Case	Demonstrator 2 will be focused in monitoring of both workforce and machinery within a construction area through the deployment of a WSN that will provide information about the status and location of the workers, the workforce in real time. This information will be managed through an IoT platform, registering possible dangers and alarms, apart from establishing a protocol in case of emergency.
AI Application(s)	Application for monitoring of machinery and workers at construction site through a WSN and an IoT platform
Expected outcome	<p>The expected outcome is a system capable of obtaining a variety of data (position of machinery and workers, daily alerts, heat maps...) and process them with real-time analytics AI software to provide security alarms and correlations.</p> <p>Predictions must be made about risk situation modelling and prevention, potential collisions, pattern extraction from previous alerts, etc with an XAI approach (using explainable models)</p>
Input Variables	<ul style="list-style-type: none"> • Time-series • Alphanumeric data • Real-time produced metrics with object identification (workforce/machinery)

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

	<ul style="list-style-type: none"> Positioning (spatial coordinates)
Algorithms to be used	<p>It is expected to use 1-3 AI models.</p> <p>The type of algorithms will be defined on the future, but we will probably try to explore different possibilities</p> <p>As a first approximation, simple ML algorithms are expected to be used for testing and validation purposes. Then they can be upgraded to DL algorithms (e.g. CNN)</p>
Training vs Inference	<p>Training will be done in the cloud and inference in the edge:</p> <ul style="list-style-type: none"> Training: <p>Pretrained models are probably going to be used. Depending of included pre-trained model libraries, further research on this will be done in WP5.</p> <p>The training process is going to be carried out with historical data on first approach, but retraining process on the edge will be studied.</p> <p>The expected input and output will be defined in WP5 and UC, closely related to data availability.</p> <p>Distributed training process or federated learning will be studied.</p> <ul style="list-style-type: none"> Inference will be carried out on the edge
Tools / frameworks to be used	<p>Different framework and libraries will be studied, making sure that Machine Learning and Deep Learning algorithms and models are covered: Tensorflow, ONNX, Keras, Apache TVM and Apache MXNet.</p>
Required hardware for AI applications	<p>Versal (or preferably some platform compatible with Versal sw stack: Vitis...) Other options will be considered depending on resource necessities (Zynq Ultrascale, Zynq-7000, or others)</p> <p>Other possibilities apart from Versal and Xilinx SW/HW stacks can also be considered (Noel-V, Ariane)</p>
Expected performances	<p>Not defined yet.</p>

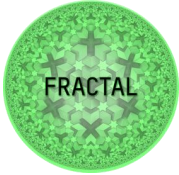
Table 5 – Requirements for Use Case 1 - Demonstrator 2

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

5.2 Use Case 2

Requirements for Use Case 2, regarding AI-based air path control are reported in Table 6.

UC2: AI-based air path control	
Short Description of the Use Case	In this Use Case parts of the standard controllers for the air path control of an internal combustion engine will be replaced by a reinforcement algorithm to improve the quality of the controls and reduce calibration effort. Furthermore, different driving behaviours, environmental changes and engine production variation will be considered by an adaptation algorithm and retraining in the cloud.
AI Application(s)	The application is an AI-controller for the combustion engine air path actuators (throttle and exhaust gas recirculation valve).
Expected outcome	The expected outcome of this use case is an AI-based controller for the air path which has the same or even better accuracy than the conventional controllers.
Input Variables	In this use case mainly sensor and virtual of the combustion engine will be used such as exhaust mass flow, emissions, pressures, temperatures, ...
Algorithms to be used	In this use case the multi agent reinforcement learning approach will be adopted.
Training vs Inference	The training will be done in a simulation environment using an engine model and already existing input data. The inference will be done on the node.
Tools / frameworks to be used	Tensorforce (Tensorflow), KerasRL (Keras), Scikit-learn, Scipy, Stable Baselines, Gym, Pytorch.
Required hardware for AI applications	GPU (at least 4GB of RAM) or multi-core CPU (16GB of RAM) for training, NN accelerators (Versal AI engine) for inference.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

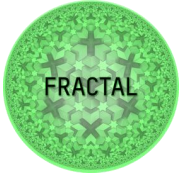
Expected performances	The AI methods are evaluated by means of a simulation of the real driving cycles where the control algorithm is verified. For the assessment of the results similar evaluation metrics, as the ones used for standard control strategies will be used (e.g., model fit, NRMSE, standard deviation of relative error distribution, ...).
-----------------------	---

Table 6 – Requirements for Use Case 2

5.3 Use Case 3

Requirements for Use Case 3, regarding AI-based system for reading conventional meters, are reported in Table 7.

UC3: Smart meters for everyone	
Short Description of the Use Case	The goal of UC3 is developing a low-cost machine-vision based applications to read conventional meters.
AI Application(s)	The goal of the AI applications is to read and recognize numbers from conventional meters.
Expected outcome	The expected outcome of the inference is the extracted meter in form of a digital number.
Input Variables	The input of the models will be images.
Algorithms to be used	Convolutional Neural Networks will be used to recognize numbers in the images.
Training vs Inference	<p>Training will be performed offline using real images. Data could be both labelled (i.e. containing the information about the output) or unlabelled. Data are currently not available in their entirety but will be collected/integrated during the project.</p> <p>Inference will be done on the edge. It is expected that data flows from a camera few times per day and some latency is</p>

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

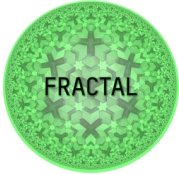
	acceptable since there is no safety critical operation to be done.
Tools / frameworks to be used	Not known yet
Required hardware for AI applications	One of the challenges of this use case is to reduce the complexity of the system, since energy consumption of the node is a key factor. For this reason, the final system will be kept as simple as possible. Ideally, it will consist of 1 accelerator (CNN) + memory + non-volatile storage + processor. Moreover, the system will be designed to be kept on only for a short time to save battery's life.
Expected performances	To be defined.

Table 7 – Requirements for Use Case 3

5.4 Use Case 4

Requirements for Use Case 4, regarding a low-latency object detector, are reported in Table 8.

UC4: Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications	
Short Description of the Use Case	The use case will develop a detection system able to perform real-time object recognition for industrial applications where automation replaces manual work. AI features enhances the automation process with intelligent capability to detect and recognize objects visually.
AI Application(s)	The goal of AI methods is to design a machine vision-based object detection and recognition algorithm in form of a Low-Latency Object Detection (LLOD) building block.
Expected outcome	The expected outcome is the detection and labelling of the objects from the video stream in real time.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

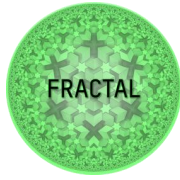
Input Variables	Input variables will derive from images and video streams.
Algorithms to be used	Tiny Yolo for image analysis.
Training vs Inference	Training is not the focus of the use case. It will be performed offline out of the Fractal node and used for inference as it is.
Tools / frameworks to be used	Darknet.
Required hardware for AI applications	RISC-V and Arm64 will be tested in the use case.
Expected performances	Performances will be evaluated in terms of correctness of detection and classification. Expected performance will be further analysed, but at the present time we a 85% accuracy in recognizing objects could be considered a target value.

Table 8 – Requirements for Use Case 4

5.5 Use Case 5

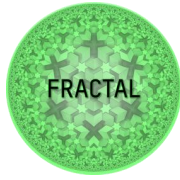
Requirements for Use Case 5, regarding AI-based techniques for improving the safety of autonomous trains, are reported in Table 9.

UC5: Increasing the safety of an autonomous train through AI techniques	
Short Description of the Use Case	The Use Case is focused on automatic accurate stopping and safe passenger transfer for autonomous train operation (traction and brake commands and doors opening and closing software modules are outside of the scope of the use-case that will provide only the detection capacity), using Computer Vision AI-enhanced techniques.
AI Application(s)	This Use Case is composed of two applications: <ol style="list-style-type: none"> Automatic accurate stop. This application includes: <ul style="list-style-type: none"> Detection of the platform area, based on train localization information and different visual patterns detection/identification. Platform detection functionality



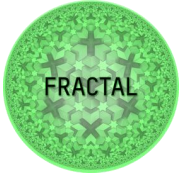
Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

	<p>will enable automatic train approximation to accurately stop the train.</p> <ul style="list-style-type: none"> Detection of stop signals at platform for accurate automatic stop at door equipped platforms, aligning the vehicle and platform for correct passenger transfer. <p>2. Safe passenger transfer, for which it is necessary to perform a correct detection of the passengers who are getting in/out the train, avoiding any door closing operation before all train's doors are free of crossing-passengers or obstacles.</p>
Expected outcome	<p>The expected outcomes for each application will be:</p> <ol style="list-style-type: none"> Neural network for <i>Platform detection</i> and <i>Stop signals detection</i> and <i>distance</i> estimation to those stop signals in the platform. Neural network for <i>Passenger or obstacle detection</i> around train doors, making sure the train is completely stopped in the platform area (using visual sensors) avoiding a) door opening operation if the train and platform doors are not precisely aligned and b) door closing operation if any passenger is getting in/out the train or any obstacle is blocking the doors.
Input Variables	<p>The input data in both applications of the Use Case will be:</p> <ul style="list-style-type: none"> Videos recorded from train cabin (front view) Videos recorded from rear mirror camera. <p>For training the models, historical recorded and labelled images will be used. For inference, recorded data during train operation will be used.</p>
Algorithms to be used	<p>In this Use Case, for both applications the neural network YoloV3 (You Only Look Once) will be used. Yolo is a well-known real-time object recognition neural network, based in Convolutional Neural Network (CNN).</p> <p>For passenger detection, a standard YoloV3 neural network will be used with pretrained weights. For platform and stop signals detection, a standard YoloV3 neural network will be retrained (transfer learning) to detect platform and specific signals. Distance estimation will be computed from a stereo</p>



Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

	<p>camera setup in the front of the train. Both cameras will detect the signals with YoloV3 and a triangulation computer vision algorithm will be used to match both images and detect distance to the signal.</p>
Training vs Inference	<p>The models will be trained offline, in the office labs, using Darknet framework. The training is going to be done in the Office GPUs. The retraining of the models also will be done in the office GPUs</p> <p>The inference will be performed in the edge (Fractal Nodes) using EDDL framework. The predictions must be done in real time, with 10fps as target inference speed.</p>
Tools / frameworks to be used	<p>The tools or libraries to use must be able to read the ONNX (Open Neural Network Exchange) format to be able to load YoloV3 networks. For vision-based measurement models (distance estimation) the OpenCV library will be used.</p> <p>The training of the models (offline) will be done using Darknet/YoloV3 open-source neural network framework on Linux workstations with powerful GPUs.</p> <p>To perform the inference on the edge, EDDL will be used.</p>
Required hardware for AI applications	<p>The inference will be made in Fractal node, which will require that it will provide:</p> <ul style="list-style-type: none"> • 4 cores (at least). • multi-threading support. • 60 GFLOPS (at least). • 16GB DDR RAM (at least). • HW accelerators based on GPU • Linux OS • C++ compiler • Different interfaces (and their Linux drivers): <ul style="list-style-type: none"> ○ 2xGbit peripherals ○ 2xUSB3.0 peripherals ○ 1xHDMI peripherals <p>In addition to this, the node shall support OpenCV library, provide ONNX interpreter, and the HW accelerators shall be compatible with TensorFlow's framework outputs.</p>

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		


Expected performances	Both data arrival for inference and predictions shall be made in real time, considering 10fps (frames per second), or 100ms cycle, as real-time.
-----------------------	--

Table 9 – Requirements for Use Case 5

5.6 Use Case 6

Requirements for Use Case 6, regarding a smart totem equipped with sensors and actuators, are reported in Table 10.

UC6: Intelligent Totem	
Short Description of the Use Case	The UC6 is centred on the development of a smart totem, equipped with smart sensors and actuators, that collect data and implement AI based content analysis. The totem should have an impact on retail and shopping mall business, providing to the customers personalized advertisements, product recommendations and guiding them towards products in the store.
AI Application(s)	<p>This use case includes three specific AI based blocks:</p> <ul style="list-style-type: none"> to process images collected by cameras to detect heterogeneous data like user age/gender, detect and count people at totem proximity, etc. to process audio signal collected by microphones to detect speaker age, gender, and language to process data generated from the aforementioned AI blocks and from other data sources to select content and information to be provided, output channels among those available and other eventual actions.
Expected outcome	The goal of the AI blocks is to make the totem more accessible and faster to use for the customer in the reference scenario. Will be developed advanced AI approaches, which will be deployed on the edge to process collected data in order to extract proximity information and to understanding its surrounding.

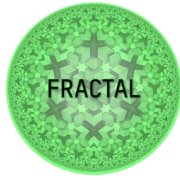
	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Input Variables	<p>Input variables will include:</p> <ul style="list-style-type: none"> • images from video recordings, • audio streams, • variables from interactions with the totem and with apps.
Algorithms to be used	<ul style="list-style-type: none"> • Convolutional Neural Networks, Resnet for video processing. • Audio feature extraction + Support Vector Machines for audio processing. • Rule-based methods for standard classification.
Training vs Inference	<p>Training will be done offline. In a first implementation, the model deployed on the edge will be static, but it is also possible to set up a regular update of the model. Also a federated approach will be explored. The inference is done on the edge. Each node should be able to take decision autonomously both having information from other nodes and without such information.</p>
Tools / frameworks to be used	TensorFlow, Keras, OpenCV.
Required hardware for AI applications	<p>Training could be done in a cloud environment. Inference requires the possibility of storing data locally. In general, the implementation will be light so, no large resources are needed. On x86_64 architecture, a 4-core/8-thread CPU, and 8 GB of RAM should suffice to process one camera. On ARM architecture, hardware acceleration is more likely required (i.e. integrated GPUs on NVidia Jetson boards).</p>
Expected performances	Still to be defined

Table 10 – Requirements for Use Case 6

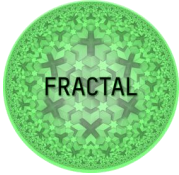
5.7 Use Case 7

Requirements for Use Case 7, regarding an autonomous robot, are reported in Table 11.



Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

UC7: Autonomous Robots (SPIDER)	
Short Description of the Use Case	SPIDER is an autonomous robot prototype. Functions of the robot shall be ported to FRACTAL edge nodes. Relevant functions are the collision avoidance and the path tracking function
AI Application(s)	Evading obstacles along a predefined path based on a cost map (occupancy grid) using AI.
Expected outcome	The AI function delivers control values for the robot which allow the robot to follow a path while evading obstacles.
Input Variables	Costmaps (generated out of lidar point clouds), precise positioning and vehicle odometry (speed, orientation, etc...)
Algorithms to be used	Reinforcement learning using a reward strategy by rewarding the proximity to the path and penalizing crashes with obstacles.
Training vs Inference	<p>Training is not performed on the edge since it requires CUDA. Some data for training the model are already available, but the training set will be continuously extended to improve the performances. The training phase will be done once, there is no need of retraining the model.</p> <p>Running the model shall be possible on a simple Linux platform. Recommended dual core with about 2GB memory. Learning will be centralized (no need for federated learning).</p>
Tools / frameworks to be used	C++ or Python API, OpenCV Library, TensorFlow, Keras (for training only, not needed on the edge node).
Required hardware for AI applications	The target device is PULP RISC-V. Arm64 is desirable. At least 2 GB RAM and 2 CPU cores are required.
Expected performances	The performances of the AI methods will be evaluated in terms of:

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

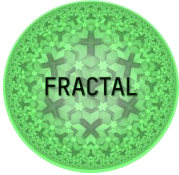
	<ol style="list-style-type: none"> 1. Proximity of the calculated output trajectory 2. Awareness of obstacles 3. Frequency and reliability of predictions <p>A 3D simulation will be used for verification of the vehicle functions.</p>
--	---

Table 11 – Requirements for Use Case 7

5.8 Use Case 8


Requirements for Use Case 8, regarding an autonomous warehouse system, are reported in Table 12.

UC8: Autonomous warehouses	
Short Description of the Use Case	Handling, storage, and retrieval of warehouse goods by automated shuttles are optimized using Artificial intelligence techniques. AI will also optimally organize and analyse the masses of generated data, in order to improve the warehouse throughput.
AI Application(s)	<p>Two AI applications will be taken into consideration:</p> <p>Application 1: pathfinding. An AI module will be developed to allow optimization of paths in order to find in real time the best option for the shuttles to maintain the throughput high.</p> <p>Application 2: predictive maintenance. To allow predictive maintenance features to be developed, machine learning is required in order to predict failures of certain parts and devices.</p>
Expected outcome	<p>Application 1: The AI of the FRACTAL node should support pathfinding to find the best available shuttle for an order to keep the throughput high.</p> <p>Application 2: the occurrence of failures of certain parts and devices. Patterns related to anomalies and correct behaviours of the system.</p>

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Input Variables	Data deriving from images will be used for both Application 1 and Application 2.
Algorithms to be used	Application 1: Deep Learning (Convolutional Neural Networks). Application 2: Traditional machine learning (classification, anomaly detection – still under discussion)
Training vs Inference	Model will be trained using the Fraunhofer Warehouse und Industry Datasets. https://www.iis.fraunhofer.de/de/ff/lv/lok/opt1/warehouse.html Models could be retrained on the edge when needed. Learning will be centralized (no need for federated learning).
Tools / frameworks to be used	Python, C++, Pytorch.
Required hardware for AI applications	The target could be either the VERSAL or the PULP platform. At least 2 CPU cores, 4 GB RAM and 32 GB e MMC are needed. GPUs will be used for training and inference.
Expected performances	Still to be defined.

Table 12 – Requirements for Use Case 8

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

6 Proposed AI methods

6.1 Video analysis using Convolutional Neural Networks

Convolutional Neural Network is a specialized kind of neural network for processing data with a known grid-like topology, like images or time series. For our purpose we focus particularly on images.

CNNs differ from Deep Neural Networks (DNNs) in three main respects:

- Sparse interaction**
 In traditional DNNs every output unit interacts with every input unit. In CNNs, instead, an object called "kernel" or "filter" moves all over the input producing an output (which is still a weighted sum) typically smaller than the input, providing sparse interaction (also referred to as sparse connectivity or sparse weights) among the neurons. The pros is that we have to learn fewer parameters, with improvements in memory requirements, statistical efficiency (statistical strength for more samples per weight, reduced variance when estimating the parameters) and computations (from $O(m \times n)$ to $O(k \times n)$ with k much smaller than m).
- Parameter sharing**
 The same parameter is used by more than one function in the model: in traditional DNNs each weight is used exactly ones when computing the output, while in CNNs each member of a kernel (a weight) is used at every position of the input per layer.
- Equivariant representations**
 The specific type of parameter sharing causes the layers to show equivariance to translation. A function is equivariant to a function if

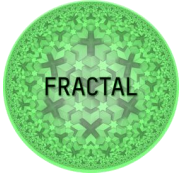
$$f(g(x)) = g(f(x))$$

In the case of CNN, g is a function translating (shifting) the input, while f is the convolution operator:

$$f(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n),$$

for a 2D input array.

As shown by Figure 5, the process inside a CNN can be divided in two steps: in the first one, the "convolutional step", the image (represented by a tensor ($rows \times cols \times d$), where d takes into account the colours, for example $d = 3$ if the image is RGB) is first flattened in a vector; then a filter is applied and after the convolution phase an output is obtained and the parameters have to be estimated; after that, a pooling

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

process can be performed. A pooling layer provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are average pooling and max pooling that summarize the average presence of a feature and the most activated presence of a feature respectively. Then the and a reduced image is then passed to the next layer where another convolutional process is performed. In the second phase, after that a suitable number of convolutional+pooling operations has been done, a classification is performed by a fully connected NN which, through a softmax¹ function, associates to each part of a grid overlapping the image a certain probability that the object belongs to.

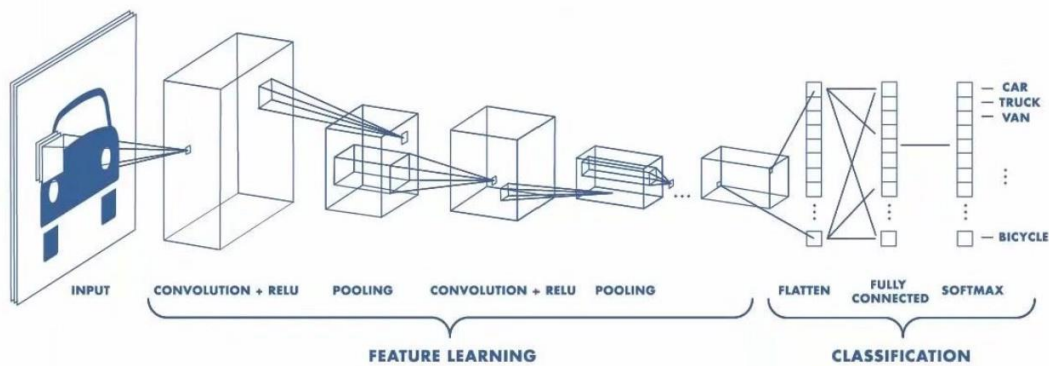


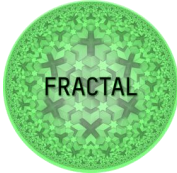
Figure 5 – Classic structure of a CNN for image recognition

A typical convolutional layer can be divided in three parts:

- **Convolution**

The convolutional layer computes the convolutional operation of the input images using kernel filters to extract fundamental features. The kernel filters (see Figure 6 and Figure 7) are of the same dimension but with smaller constant parameters as compared to the input images. As an example, for computing a $35 \times 35 \times 2$ 2D image, the acceptable filter size is $f \times f \times 2$, where $f = 3, 5, 7$, and so on. The filter mask slides over the entire input image step by step and estimates the dot product between the weights of the kernel filters with the value of the input image, which results in producing a 2D activation map.

¹ It is a generalization of the logistic function and it is used in multinomial logistic regression and is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes: $\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}$ for $i = 1, \dots, K$ and $\mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

$$\begin{array}{|c|c|c|c|} \hline X_{11} & X_{12} & X_{13} & X_{14} \\ \hline X_{21} & X_{22} & X_{23} & X_{24} \\ \hline X_{31} & X_{32} & X_{33} & X_{34} \\ \hline X_{41} & X_{42} & X_{43} & X_{44} \\ \hline \end{array}
 *
 \begin{array}{|c|c|} \hline W_{11} & W_{12} \\ \hline W_{21} & W_{22} \\ \hline \end{array}
 =
 \begin{array}{|c|c|c|} \hline Y_{11} & Y_{12} & Y_{13} \\ \hline Y_{21} & Y_{22} & Y_{23} \\ \hline Y_{31} & Y_{32} & Y_{33} \\ \hline \end{array}$$

$$\begin{aligned}
 Y_{11} &= X_{11}W_{11} + X_{12}W_{12} + X_{21}W_{21} + X_{22}W_{22} \\
 Y_{12} &= X_{12}W_{11} + X_{13}W_{12} + X_{22}W_{21} + X_{23}W_{22} \\
 Y_{13} &= X_{13}W_{11} + X_{14}W_{12} + X_{23}W_{21} + X_{24}W_{22} \\
 &\dots\dots
 \end{aligned}$$

Figure 6 – Action of a convolutional kernel over a 4x4 input: this is how a convolutional layer works inside a CNN.

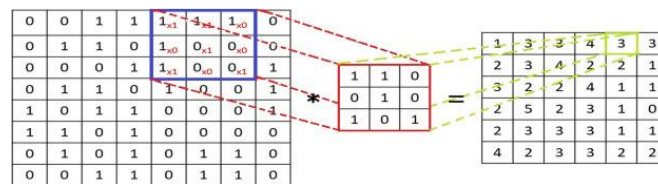


Figure 7 – Convolution between an input matrix 8x8 with a kernel filter 3x3. The output is a matrix 6x6.

Three parameters control the size of the output of a layer:

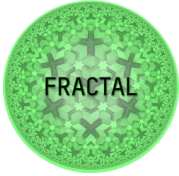
- **Depth**, the number of filters (kernels) of the layer.
- **Stride**, the step used to slide the filter on the input.
 - When stride > 1 we are down-sampling the input data.
 - **Tiling** refers to the special case where stride = kernel span.
- **Padding** to enlarge the input and allow for kernels application in each one of the (original) point.

In particular, this formula holds:

$$O = \frac{W - K + 2P}{S} + 1$$

Size BEFORE convolution → W
 Kernel size → K
 Padding → P
 Size AFTER convolution → O
 Stride → S

- **Non-linear activation**
 Training a DNN means (as usual) learning the values for the model parameters (weights, bias terms) from the training set. Training is usual performed via gradient descent and backpropagation algorithm (more details can be found in (Jordan, 2017))

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

An essential element of training is (as usual) the loss function. The goal of the loss function is to evaluate how well the network, with its current weights, is performing. More formally, this function expresses the quality of the predictions as a function of the network parameters (weight - w , biases). The smaller the loss, the better the parameters are for the chosen task. Since the loss functions represents the goal of the networks, there are as many different functions as there are tasks, with some function more commonly used than others.

The objective is to minimize the function of the error:

$$w^* = \arg\min_w \frac{1}{n} \sum_{i=1}^n L(f(x^i, w), y^i)$$

where

$$f(x) = \sigma \left(\sum_i w_i x_i + w_0 \right)$$

and σ manages the non-linearities. A typical choice is ReLU: it is applied after each convolutional layer and only preserves non-negative values. It's easy to implement, preserves saturation and prevent the vanishing gradient problem (Brownlee, 2019). Non-linearity is needed in activation functions because its aim in a neural network is to produce a nonlinear decision boundary via non-linear combinations of the weight and inputs.

- **Pooling**

Pooling is a way to summarise the information inside an input. It is simply a way to further reduce the dimensionality of the description. It provides invariance to small shifts of the inputs. An example is shown in Figure 8.

Pooling functions:

- **Average pooling:** average activation of the convolutional layer
- **Max pooling:** max activation of the convolutional layer

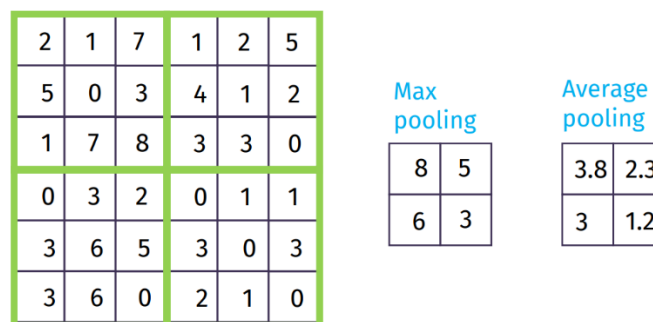
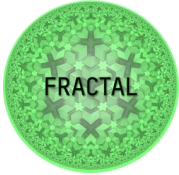


Figure 8 – Example of pooling

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Finally, training in CNN is performed in the same way as a DNN via a forward pass and a backward pass (backpropagation), as shown in Figure 9.

The "**forward pass**" refers to calculation process, values of the output layers from the inputs data. It's traversing through all neurons from first to last layer. A loss function is calculated from the output values. And then "**backward pass**" refers to process of counting changes in weights (de facto learning), using gradient descent algorithm (or similar). Computation is made from last layer, backward to the first layer.

Backward and forward pass makes together one "iteration".

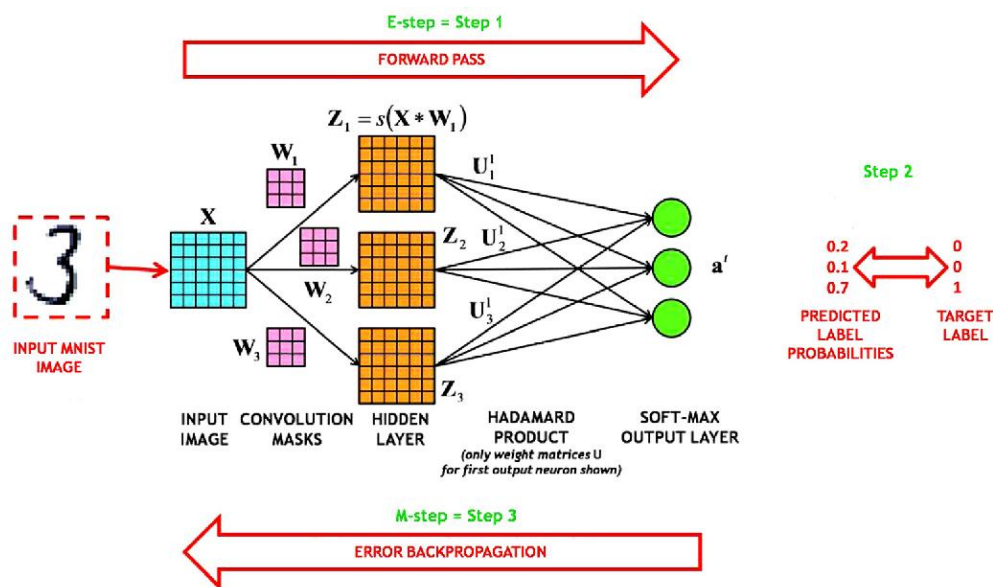


Figure 9 – Forward and backward pass in backpropagation. Keep in mind that the forward propagation computes the result of an operation and save any intermediates needed for gradient computation in memory. Backward: apply the chain rule to compute the gradient of the loss function with respect to the inputs.

The intuition behind the backpropagation, chain rule, of a CNN could be resumed in the diagram in in Figure 10.

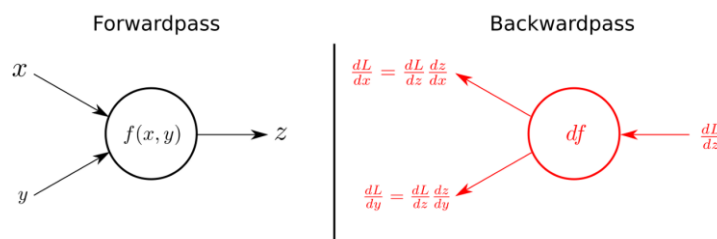
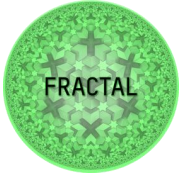


Figure 10 – The forward pass on the left calculates z as a function f(x,y) using the input variables x and y. The right side of the figures shows the backward pass. Receiving dL/dz , the gradient of the loss function with respect to z from above, the gradients of x

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

6.2 Yolo

You Only Look Once (YOLO) is an algorithm for the object detection developed in 2016 (Redmon, You only look once: Unified, real-time object detection, 2016) [6]. YOLO algorithm is an algorithm based on regression: instead of selecting the interesting part of an image, it predicts classes and bounding boxes for the whole image in one run of the algorithm (for this, Once).

Up to now there exists three different version of YOLO algorithm: YOLO(v1), YOLOv2 (Redmon & Farhadi, YOLO9000: Better, Faster, Stronger, 2017), YOLOv3 (Redmon & Farhadi, YOLOv3: An Incremental Improvement, 2018) and YOLOv4 (Bochovski, Wang, & Mark, 2020). In this deliverable we focus especially on the first one, reporting then the major differences and novelties with respect to the other methods.

6.2.1 YOLOv1

To understand how YOLO works, first of all it is necessary to understand what is actually being predicted: a class of an object and the **bounding box** specifying object location. Each bounding box can be described using four descriptors:

- Center of the box (**x**, **y**)
- Width (**w**)
- Height (**h**)
- Value **c** corresponding to the class of an object

Along with that we predict a real number p_c , which is the probability that there is an object in the bounding box.

YOLO system divides the input image into an $S \times S$ grid (typically 19×19). If the centre of an object falls into a grid cell, that grid cell is responsible for detecting that object. Due to this property the centre co-ordinates are always calculated relative to the cell whereas the height and width are calculated relative to the whole image size.

For each grid cell:

- it predicts B bounding boxes and each box has one **box confidence score**
- it predicts C **conditional class probabilities**

The **box confidence score** reflects how likely a box contains an object and how accurate is the bounding box while the **conditional class probability** is the probability that the detected object belongs to a particular class (one probability per category for each cell). These confidence scores reflect how confident the model is that the predicted box contains an object and also how accurate it thinks the box is. The combination of these scores is called **class confidence score**.

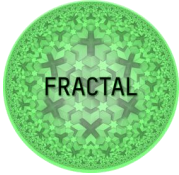
For each prediction box is computed as:

class confidence score = box confidence score × conditional class probability

It measures the confidence on both the classification and the localization (where an object is located).

Here are the mathematical definitions for your future reference.

- **Box Confidence Score** = $P_r(\text{object}) \cdot \text{IoU}$
- **Conditional Class Probability** = $P_r(\text{class}_i | \text{object})$
- **Class Confidence Score** = $\text{BCS} \cdot \text{CCP} = P_r(\text{class}_i) \cdot \text{IoU}$

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

We then have $CCS = BCS \cdot CCP$. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

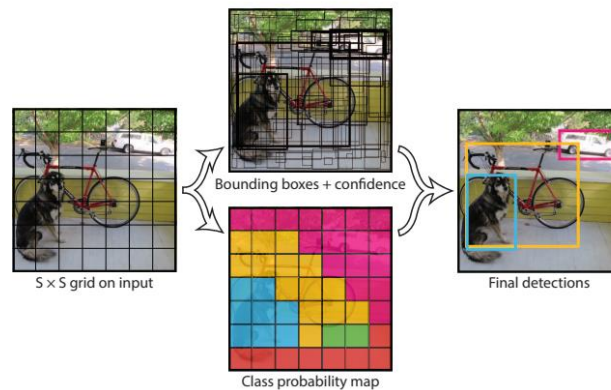


Figure 11 – Bounding boxes and class probability map. Source: (Redmon, You only look once: Unified, real-time object detection, 2016)

Intersection over Union, or Jaccard index, is an evaluation metric used to measure the accuracy of an object detector.

It computes size of intersection and divide it by the union. More generally, IoU is a measure of the overlap between two bounding boxes.

The higher the IOU the better is the accuracy.

An IoU score $IoU \geq 0.5$ is normally considered as a true positive.

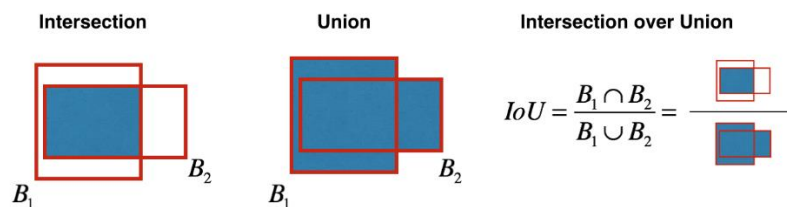


Figure 12 – IoU operation

After predicting the class probabilities, the next step is **Non-max suppression**, it helps the algorithm to get rid of the unnecessary boxes that are in fact different detections of the same object: there could be numerous bounding boxes calculated based on the class probabilities and then it's necessary to choose the best one. It calculates the value of IoU for all the bounding boxes respective to the one having the highest-class probability, it then rejects the bounding boxes whose value of IoU is greater than a threshold. It signifies that those two bounding boxes are covering the same object but the other one has a low probability for the same, thus it is eliminated. Summarizing the procedure, for each object class:

- discard all those bounding boxes where probability of object being present is below some threshold (0.6),
- take the bounding box with the highest score among candidates,
- discard any remaining bounding boxes with IoU value above some threshold (0.5)

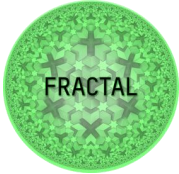
	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		



Figure 13 – The effect of non-max suppression in bounding box identification. Source: (Gupta, 2020)

The model is implemented as a convolutional neural network. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates. The overall architecture of the algorithm can be viewed below:

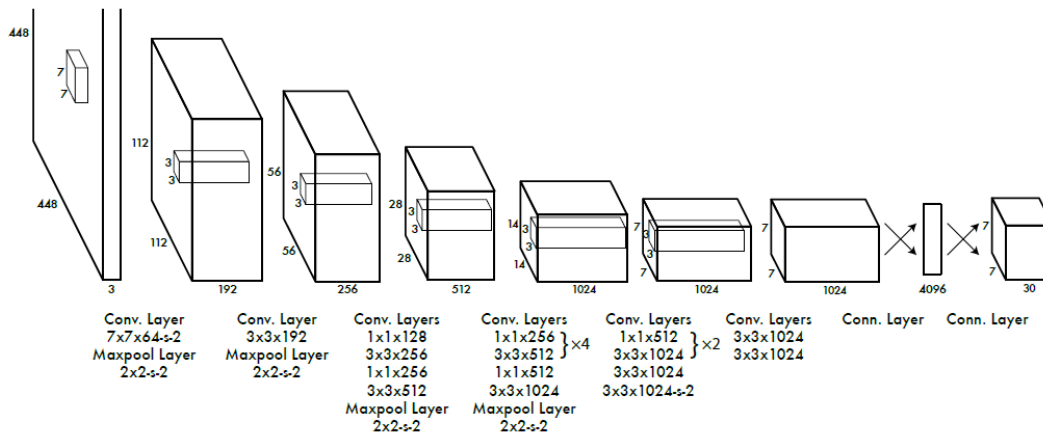



Figure 14 – YOLO Architecture (source: You Only Look Once: Unified, Real-Time Object detection)

Training in YOLO is made according to classical CNN, minimizing the empirical risk with the total loss function that is the sum of the three losses previously mentioned. By combining the three terms, the loss penalizes the error for bounding box coordinates refinement, objectless scores, and class prediction. By backpropagating the error, we are able to train the YOLO network to predict correct bounding boxes.

Precision-Recall curve

The precision-recall curve is used for evaluating the performance of binary classification algorithms. Any prediction relative to labelled data can be a true positive, false positive, true negative, or false negative.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

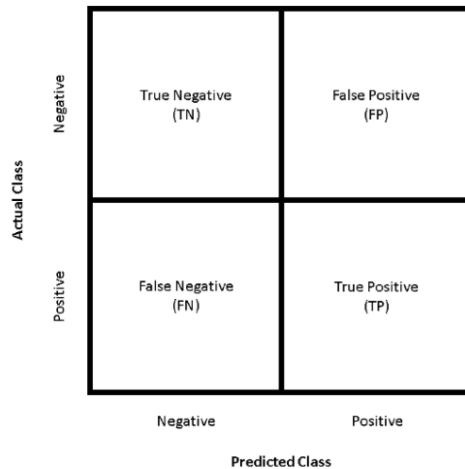


Figure 15 – Confusion matrix structure for binary classification problems

The precision-recall curve is constructed by calculating and plotting the precision against the recall for a single classifier at a variety of thresholds. For example, if we use logistic regression, the threshold would be the predicted probability of an observation belonging to the positive class. Normally in logistic regression, if an observation is predicted to belong to the positive class at probability > 0.5 , it is labelled as positive. However, we could really choose any probability threshold between 0 and 1. A precision-recall curve helps to visualize how the choice of threshold affects classifier performance and can even help us select the best threshold for a specific problem.

Precision (also known as positive predictive value) can be represented as:

$$Precision = \frac{TP}{TP + FP}$$

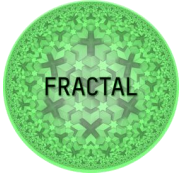
where TP is the number of true positives and FP is the number of false positives. Precision can be thought of as the fraction of positive predictions that actually belong to the positive class.

Recall (also known as sensitivity) can be represented as:

$$Recall = \frac{TP}{TP + FN}$$

where TP is the number of true positives and FN is the number of false negatives. Recall can be thought of as the fraction of positive predictions out of all positive instances in the data set.

The figure below demonstrates how some theoretical classifiers would plot on a precision-recall curve. The grey dotted line represents a “baseline” classifier — this classifier would simply predict that all instances belong to the positive class. The purple line represents an ideal classifier — one with perfect precision and recall at all

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

thresholds. Nearly all real-world examples will fall somewhere between these two lines — not perfect but providing better predictions than the “baseline”. A good classifier will maintain both a high precision and high recall across the graph and will “hug” the upper right corner in the figure below.

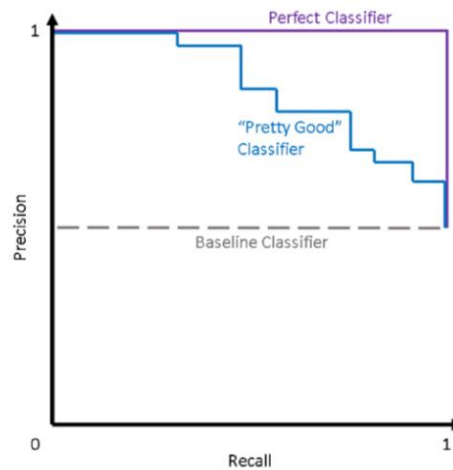


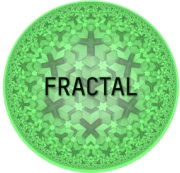
Figure 16 – Some theoretical precision-recall curves

6.2.2 YOLOv2

YOLO makes a significant number of localization errors. Furthermore, YOLO has relatively low recall. Thus, in the second version of YOLO they focused mainly on improving recall and localization while maintaining classification accuracy. According to (Redmon & Farhadi, YOLO9000: Better, Faster, Stronger, 2017), YOLOv2 has better performances due to these ideas:

- Batch Normalization
- High Resolution Classifier
- Convolutional With Anchor Boxes
- Dimension Clusters
- Direct location prediction
- Fine-Grained Features
- Multi-Scale Training

Figure 17 shows the improvements in mean Average Precision (mAP) (Arlen, 2018) of YOLOv2 with respect to YOLO. It is clear that due to the previous mentioned improvements, this new version of YOLO has been improved.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	78.6

Figure 17 – Differences between YOLO and YOLO v2 (source (Redmon & Farhadi, YOLO9000: Better, Faster, Stronger, 2017))

As far as the architecture is concerned, YOLO-v2 uses Darknet-19 classification network for feature extraction from the input image. The model has many 1x1 convolutional layers; this reduces the number of parameters compared to the "built-in" classifier of the first version of YOLO. Combinations of different datasets were used to train YOLO-v2, using the hierarchical class representation technique called WordTree (see Section "Classification" in (Hui, 2018)). This method allows YOLOv2 to recognize a large number of objects and, indeed, it is also known in literature as YOLO9000, since it can detect up to 9000 different objects in real-time.

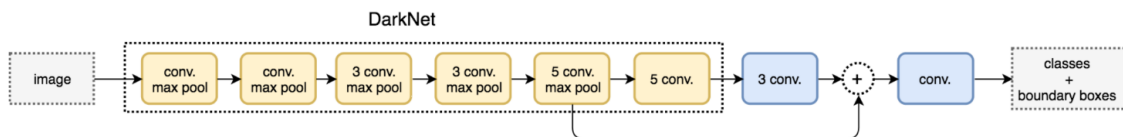


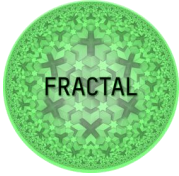
Figure 18 – YOLO with Darknet-19

6.2.3 YOLOv3

YOLO v3 is a better and stronger but not faster upgrade of YOLO v2. The main changes are in the architecture of the NN: here the authors propose to use a more refined version of the Darknet used for YOLO v2, increasing the number of the layers from 19 to 53 (Darknet-53 precisely). For the task of detection, 53 more layers are stacked onto it, giving us a 106 layer fully convolutional underlying architecture for YOLO v3. This is the reason behind the slowness of YOLO v3 compared to YOLO v2. The newer architecture boasts of residual skip connections, and up-sampling. The most salient feature of v3 is that it makes detections at three different scales. YOLO is a fully convolutional network and its eventual output is generated by applying a 1 x 1 kernel on a feature map. In YOLO v3, the detection is done by applying 1 x 1 detection kernels on feature maps of three different sizes at three different places in the network.

Other novelties and improvements are:

- Better at detecting smaller objects
- Increase the number of anchor boxes (9, 3 per each direction)

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

- Changes in the loss function: **object confidence and class predictions in YOLO v3 are predicted through logistic regression.**
- **Multilabel classification for objects detected in images**

Strictly speaking, YOLO-v3 is simply an improved version of YOLO-v2. With this latest update the authors have mostly integrated other people's ideas with their system and made minor changes to the model architecture.

6.2.4 YOLOv4 and Tiny-YOLO

YOLOv4 is an object detection algorithm that is an evolution of the YOLOv3 model. It is the real state of the art in this field, according to the scientific community. Compared to YOLOv3, the v4 achieves better results in AP (Average Precision) and FPS (Frames Per Second), by 10% and 12% respectively.

The new features used by YOLOv4 are essentially:

- *Weighted-Residual-Connections (WRC)*
- *Cross-Stage-Partial-connections (CSP)*
- *Cross mini-Batch Normalization (CmBN)*
- *Self-adversarial-training (SAT)*
- *Mish activation*
- *Mosaic data augmentation*
- *DropBlock regularization*
- *Complete Intersection over Union loss (CIoU loss)*

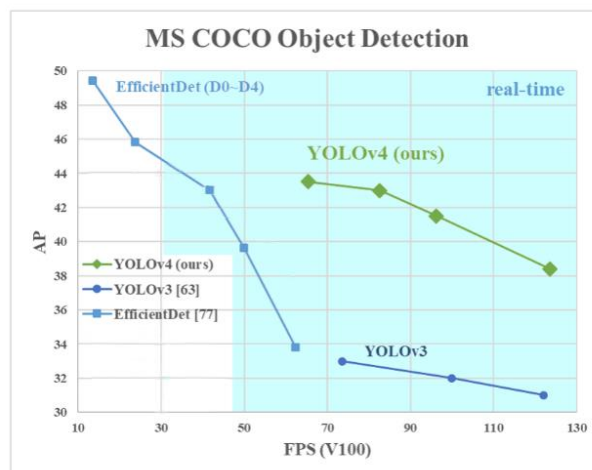
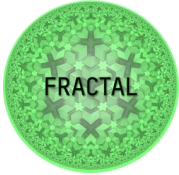


Figure 19 – The performances of the different YOLO versions on the MS COCO benchmark. Source (Bochovskiy, Wang, & Mark, 2020)

YOLOv4-tiny is the compressed version of **YOLOv4**. It is proposed based on YOLOv4 to make the network structure simpler and reduce parameters so that it becomes feasible for developing on mobile and embedded devices. The **FPS** (Frames Per Second) in YOLOv4-tiny is approximately eight times that of YOLOv4. However, the **accuracy** for YOLOv4-tiny is 2/3rds that of YOLOv4 when tested on the MS COCO dataset. For **real-time object detection**, **YOLOv4-tiny** is the better option when compared with **YOLOv4** as faster inference time is more important than precision or accuracy when working with a real-time object detection environment.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

6.3 Audio Stream Analysis

Some use cases would have benefit by employing techniques able to process and extract information from audio stream. Audio analysis has received a grown interest as witnessed by the wide variety of commercial applications including speaker verification and authentication procedures, gender recognition and language recognition. Other common applications of speech processing techniques lie in the range of accessibility solutions: the most remarkable examples are the speech-to-text and text-to-speech tools.

In general, we can identify two categories of applications: those focusing on the recognizing the speaker or some characteristics (such as the gender or the age) of the speaker (Speaker Recognition) and those concerning the speech itself (Speech Recognition).

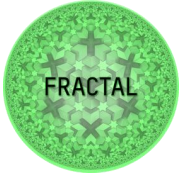
It is worth noting that in several real-world situations the speaker is surrounded by many smart devices, such as mobile phones and therefore understanding audio context represent an important tool allowing relevant applications. Consider, for example, the case of a speaker talking during a seminar, a conference call or a lecture in a classroom.

Since the environmental conditions deeply affects the quality of audio signals and, then, the possibility of extracting information from them, it is crucial to evaluate the performance of speech processing techniques in different noise conditions and with different configurations (including e.g. different distances between source and receiver).

In order to enhance the speech recognition performances in challenging environmental conditions, a noise and distance-robust speech/speaker identification method will be proposed for inclusion in the FRACTAL node. This method embeds a smart pre-processing method employing Voice Activity Detection (VAD) able to increase the system accuracy, which is usually evaluated in terms of the rate of correct classifications.

The task of Speaker Recognition consists in identifying persons from an audio sample of their voice. It can be tackled in two different scenarios: Closed-Set scenario, when the recognized speaker belongs to a given, a-priori known set of individuals, and Open-set scenario (also called out-of-set speaker identification, if the speaker to be identified could also be out of the pre-defined set of speakers. Both these scenarios will be described in this deliverable. It is worth noting that similar approaches can also be adopted in the scenario where the gender or the language is to be recognized instead of the speaker.

The first step to build a speaker recognizer consists in dividing the audio signal into short segments called frames, during which speech can be considered as stationary. Each frame has a length of $T = 25$ ms and it is selected so that the time distance between the centres of two adjacent frames is equal to 10 ms (i.e., two consecutive frames are overlapped for one third of their duration). Then, the feature extraction

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

step allows to convert the signal into a set of variables that can be used in the recognition task. Several different set of features could be computed: a possible choice that will be explored in the FRACTAL project consist in the first 13 Mel-Frequency Cepstral Coefficients (MFCC).

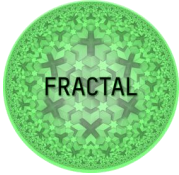
The MFCCs represents the short-term power spectrum of a sound, based on a a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency (Sabra, 2021). They are usually computed by means of the Fourier Transform of an excerpt of a signal. The Fourier spectrum is then mapped onto the mel scale and the powers at each of the mel frequencies is considered. They compose an array m of values. Then, the discrete cosine transform of m is computed. The amplitudes of the resulting spectrum are the MFCCs. Moreover, the corresponding 13 Delta-Delta will be considered: they represent the second order derivative of MFCCs. In this way each signal is converted into a set of 26 features.

This set of features is used to train a supervised classifier. The first step in order to obtain a good classification system is to pre-process the signal by removing frames that do not contain useful information. When dealing with audio speech signals, this step translates into eliminating the audio frames that do not contain speech utterances. This step is called Voice Activity Detection (VAD). To achieve this goal, many techniques have been proposed in the literature (Graf, Herbig, Buck, & Schmidt, 2015). One of the simplest methods (Uhle & Bäckström, 2017) consists in applying a Band-Pass Filter (BPF) centred over the speech bandwidth (e.g., from about 50[Hz] to 3500[Hz]). This action will remove unwanted frequency components that do not fall within the voice bandwidth, but it does not ensure that the remaining signal components are actually related to speech utterances.

In FRACTAL an alternative approach to voice activity detection will be proposed. It consists in taking into account only actual speech frames according to a proper algorithm. This pre-processing method is called SmartVAD and it consists in a short-time spectral analysis pre-processing filtering system. It is based upon two important indicators: *i*) Spectrum Flatness Index (SFI) and *ii*) Energy Ratio Index (ERI). The rationale behind these two parameters is that a speech frame exhibits a spectrum having most of its energy within the 1st [kHz] and which should not be flat. Finally, a threshold criterion is applied such that an audio frame is not discarded only if the value of the considered indicators satisfies the threshold condition. These are important indicators of a voiced frame, and they motivate the choice of the proposed VAD indicators, which effectively represent the nature of the considered signal.

Concerning artificial intelligent functions implementing the recognition phase, a Support Vector Machines (SVM) model is planned to be adopted. SVM is a supervised learning scheme that uses a binary approach to assign samples to a class, by dividing the feature space into different regions, one corresponding to each category. The SVM algorithm works in two separate phases: learning phase and the inference phase.

Training Phase: the feature vector is defined for a given frame out of F frames for each recording. Moreover, the vector containing all the classes (i.e., the names of all

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

the n_s speakers in the following but in general can be the gender of the speaker and/or the language) should be defined as well. The main idea of the SVM algorithm is to separate the feature space by means of one single hyperplane: in this project we employ both the One-Against-All (OAA) method that constructs a SVM for each class considered recognition function, and the One-Against-One (OAO) approach, which builds more models.

In the OAA approach a model is built for each of the considered known speakers. In each model a speaker is compared with all the other ones, which compose the “negative” class. On the other hand, the OAO approach trains a single model that includes all the speakers, each characterized by a different label.

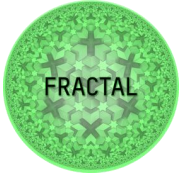
Starting from the audio signals of the training set, the single SVM, built for a class, can be obtained by computing the hyperplane that can be expressed as a function of its orthogonal vector. Analytical descriptions of the SVMs have been omitted for the sake of brevity, more details can be found here (Cortes & Vapnik, 1995).

Inference Phase: when a new audio sample has to be catalogued, the SVM model is applied. The output of SVM is a Probability Matrix (PM) for each of the separation hyperplane built by SVM. In the case of OAA, the number of PMs is the number n_s of the considered speakers, while the OAO approach produces a number of PMs equal to the number $\binom{n_s}{2}$ of combinations of n_s elements taken two at a time.

Each element of this matrix is the a-posteriori probability of a given feature vector belonging to the class identified by the binary label. From all the PMs a decision matrix is computed. It is a binary matrix where each element belonging to $\{-1, +1\}$ is obtained by associating the given frame with the binary label which has the highest probability. From the decision matrix we determine the scoring vector S , which represents the scoring of the a given. It is a measure of the likelihood of the input speech utterance to belong to such a speaker. Finally, from the scoring vector the Maximum Likelihood Index (MLI) is inferred. It is simply the index of the speaker, among the predefined set, who has the highest score value (i.e., it has the highest a-posteriori probability to have produced the input speech signal). The recognized speaker is then determined by employing a decision rule that change in case of Open-Set or Close-Set scenario. In other words, for what concerns the open-set scenario (which is the one targeted within the FRACTAL project), the maximum score obtained by the speakers belonging to the known speaker set is compared to a predefined threshold. If the highest score is above the considered threshold the recognized speaker is the one who produced the maximum score, otherwise the classifier chooses for an unknown speaker.

6.4 Reinforcement Learning

Over the past years, reinforcement learning techniques have evolved substantially and played an important role in the development of data-driven control strategies.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Reinforcement Learning is characterized by a continuous interaction of an agent with its environment. The goal of an agent is to find an optimal mapping between the observations that it receives and the actions that it produces, such that a numerical reward signal is maximized. This way, the agent learns the best actions to take for the given state.

The conventional (single agent) reinforcement learning setting is mathematically described as a Markov Decision Process (MDP), where the policy is maximized considering a stochastic stationary environment.

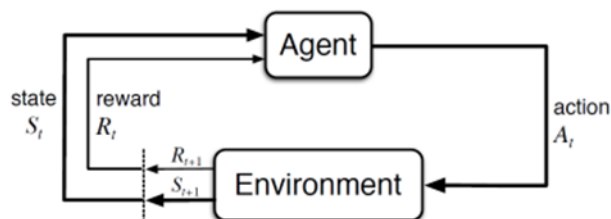
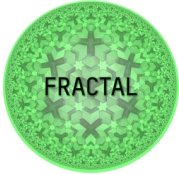


Figure 20 – The agent–environment interaction in a Markov decision process. (Barto, 2018)

Some important concepts related to RL algorithms are described here.

- **States and Observations:** The state of an environment is a full description of the current situation of the system in each moment, while an observation is a partial description of this state. If an agent is able to observe the complete state of the environment, this environment is called fully observable. If the agent only can see a partial observation, the environment is called partially observable.
- **Action spaces:** Set of all valid actions for a given environment.
- **Policies:** Set of rules used by an agent to determine what actions to take.
- **Reward:** Immediate feedback an agent receives from the environment after performing an action.
- **Return:** Reward accumulated by the agent in the long run (in the simplest case it is the sum of the rewards, but a discount factor can be applied to reduce the weight of rewards that will be received in the future).
- **Value Function:** The value function represents the expected return that comes from starting in a particular state and following a particular policy forever after. Bellman equations are commonly used for computing Value Functions, as they relate the value of the current state with the value of future states. This recursive relationship is a fundamental property of value functions and the basis of many RL algorithms.
- **Optimal Policy:** Policy producing the maximum return.

When the state and action spaces of the problem are small enough, the approximate value functions can be stored in arrays or tables and these set of RL algorithms are called *tabular solution methods*. However, for handling problems with larger state

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

and action spaces, neural networks are used as function approximators, which developed a new field called *Deep Reinforcement Learning* (Deep RL).

Multi-Agent Reinforcement Learning (MARL) represent the state-of-the-art reinforcement learning techniques for dealing with multi-agent systems, where multiple autonomous agents interact in a shared environment. In this case, the Markov decision process, that represents mathematically the single agent RL, is generalized to a Markov (stochastic) game.

In MARL, agents can interact in a cooperative, competitive or in a mixed setting. In the cooperative setting, all agents collaborate with each other to achieve a shared goal, while in the competitive setting the return of the agents should sum up to zero (agents competing against one another) and, in the mixed setting, the agents focus on improving their policies according to their own interests.

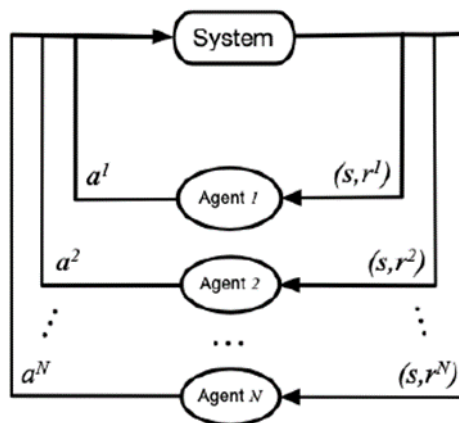
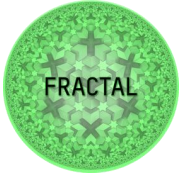


Figure 21 – The multi-agent interaction with a shared environment in a Markov game setup. (Kaiqing Zhang, 2021)

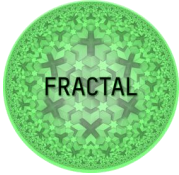
Since the objectives of the agents may not be aligned (as in competitive and mixed settings), new challenges not faced by single-agent learning are posed to MARL. Therefore, dealing with equilibrium solution concepts from game theory, such as Nash equilibrium, is required for addressing these difficulties.

MARL algorithms must fulfil requirements that guarantee its stability (convergence to a stationary policy) and capability to adapt to the changing behaviour of other agents. Convergence to equilibria is a basic stability requirement, and the above-mentioned Nash equilibria is often applied. Moreover, the trade-off between stability and adaptability needs taken into consideration. While stability is required in the learning process, the algorithm must be aware of the other learning agents. On the other hand, if adaptation is overweighted in comparison to stability, algorithms will only track the behaviour of the other agents.

MARL is a promising technique for improving strategies for multivariable control systems and it has already been successfully applied in the field of control. (Qingrui Zhang, 2020) proposed a multi-agent soft-actor critic (MASAC) algorithm using the “centralized-training-with-decentralized-execution” scheme and they guarantee the

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

closed-loop system stability by introducing a stability constraint to the policy. (Marco Wiering, 1999) combines evolutionary methods to RL techniques to compute strategies for a multi-agent soccer team and (D. Vidhate, 2017) uses Cooperative Multi-Agent Reinforcement Learning Models (CMRLM) to enhance traffic light control.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

7 Proposed AI tools, frameworks and standards

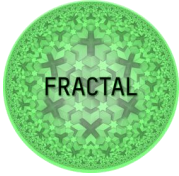
This section aims at introducing the tools, the frameworks and the standards that use case contributors are planning to use in their development. Of course, the tools have been considered to cover the AI methods that should be used: namely, standard machine learning, deep learning and reinforcement learning. However, it is worth noting that the list of available algorithms is not the only parameter that should be considered when selecting a tool for AI modules implementation. First, the AI modules should interact with other components, so the ability of interfacing with different systems/languages is crucial. So, in the choice of the tools to be used, the availability of proper APIs is of course a key factor. Moreover, since the AI models will be implemented in an edge device, the possibility of porting the AI models on different architectures is important the same. Since real time inferencing is expected in several cases, also acceleration capabilities are important. At last, since AI modules should be implemented in industrial applications, the license under which the tools are released must be considered. The following paragraphs introduce the proposed tools while a final paragraph presents a comparison among them.

7.1 Tensorflow

TensorFlow (TensorFlow, 2021) is a free and open-source library for machine learning. Even if it provides a wide range of algorithms for data analysis, TensorFlow is particularly focused on deep learning methods for the analysis of non-structured data, such as those deriving from natural language and images. It was first developed by Google Brain for Google internal use and since 2015 it is released under the Apache License 2.0. A TensorFlow Lite library is also available for mobile development.

TensorFlow is compatible with the most common 64-bit operating systems (Windows, Linux and Mac OS) as well as with Android. Moreover, it can run across a wide variety of platform (CPUs and GPUs) and also from desktops to clusters of servers to mobile and edge devices. Moreover, Google developed an Application Specific Integrated Circuit (ASIC) processor specifically designed for TensorFlow applications. This processor, named Tensor Processing Unit (TPU) is able to run TensorFlow operations much faster than standard CPUs. The third generation of TPU, released in 2018, provides up to 420 teraFLOPS of performance and 128 GB High Bandwidth Memory (HBM). When organized in clusters, TPUs can reach more than 100 petaFLOPS of performance and 32 TB HBM. Since 2018 TPU are also available in Google Cloud Platform. In 2018 an Edge TPU was released to allow implementation of TensorFlow Lite machine learning models on edge devices such as smartphones.

TensorFlow provides stable Python (for version 3.7) and C APIs and without API backwards compatibility also C++, Go, Java, JavaScript and Swift are supported. Third-party packages are available for C#, Haskell, Julia, MATLAB, R, Scala, Rust, OCaml and Crystal.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Among the algorithms provided by TensorFlow it is worth mentioning Automatic Differentiation, Convolutional Nets (a class of deep networks), Recurrent Nets, Boltzmann machines and Deep Belief Networks. Moreover, TensorFlow provides a set of pre-trained models than can directly be used for making inferences without the need of gathering data and training the model on them.

7.2 Keras

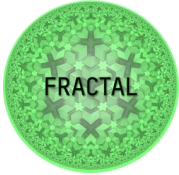
Keras (Keras, 2021) is an open-source Python library for machine learning and neural networks. It can be considered as a high-level abstraction interface to other lower-level libraries such as TensorFlow, Microsoft Cognitive Toolkit (CNTK) and Theano. It has been created as part of the R&D project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), mainly by François Chollet, a Google engineer and the code is currently hosted by GitHub. Its main strongpoint consists in the possibility of a quick and easy prototyping of solutions based on deep neural networks. Since 2017 Keras is officially supported by TensorFlow and can therefore be considered as a high-level abstraction layer for TensorFlow library. Thanks to this, Keras allow implementation of several AI models: besides standard machine learning, Keras includes modules for convolutional and recurrent neural networks and, moreover, it supports other common utility layers like dropout, batch normalization, and pooling. Keras allows the implementation of deep neural network models on mobile devices (iOS, Android), on the web or on Java Virtual Machines. In addition, also distributed training of deep neural networks is allowed on GPU or TPU clusters.

7.3 Apache TVM

Apache TVM (Apache TVM, 2021) is an open-source machine learning compiler framework for CPUs, GPUs, and machine learning accelerators. It aims to enable machine learning engineers to optimize and run computations efficiently on any hardware backend.

It provides two main features:

- Compilation of DL models into minimum deployable modules.
- Infrastructure to automatically generate and optimize models on more backends with better performance.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

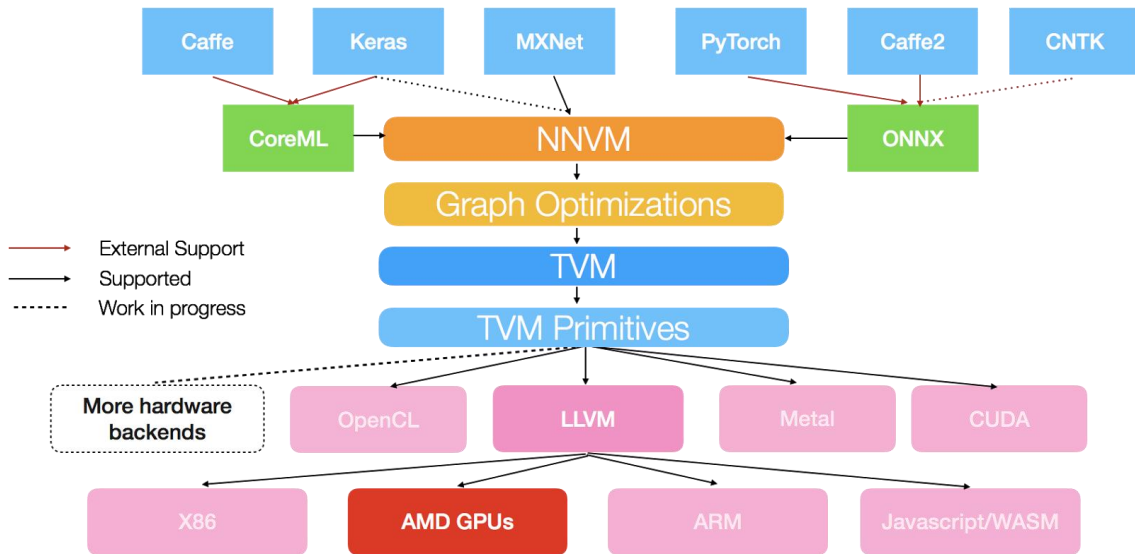


Figure 22 – Apache TVM Blog - Bringing AMD GPUs to TVM Stack and NNVM Compiler with ROCm.

As shown in Figure 22, Apache TVM supports a wide range of hardware backends, such as X86, AMD GPUs and ARM. In addition, it supports adding new hardware backends. It is easier to add a new hardware backend if it has LLVM support. If it is a GPU it is recommended to use CUDA, OpenCL or Vulkan backend, although it is not mandatory (<https://tvm.apache.org/2017/10/30/Bringing-AMDGPUs-to-TVM-Stack-and-NNVM-Compiler-with-ROCM>).

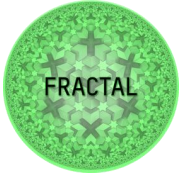
Additionally, TVM features AutoTVM, a module that performs automatic optimizations for any hardware backend. It uses a statistical cost model to learn which optimizations lead to less computational time, regardless of the model and hardware architecture.

Apache TVM leverages being part of Apache Software Foundation, one of the largest open-source developer communities, which offers continuous development and support. Some of the organizations that use or contribute to Apache TVM are AMD, AWS, Huawei, Intel, Microsoft, Nvidia and Samsung.

7.4 Apache MXNET

Apache MXNet (Apache MXNet, 2021) is an open-source framework for Deep Learning under the license of the Apache Foundation.

One of the key aspects of MXNet is that it allows for high-speed performance (MXNet is built on C++) and enables working with parallel computation. MXNet is extremely portable, supporting several languages such as Python, Scala, Java, Clojure, R, Julia, Perl or C++ and is available in the most common OS as Linux, MacOS or Windows. Its flexibility and availability have made MXNet one of the most widely used Deep Learning frameworks nowadays.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

For Deep Learning models, MXNet implements Gluon, one of the most versatile DL libraries available. Gluon applies hybrid programming, as opposed to TensorFlow or PyTorch, which use symbolic and imperative programming respectively, and it allows MXNet to work as imperative for developing but symbolic for deploying.

For Python development, Gluon implements several APIs like GluonCV, GluonNLP or GluonTS for computer vision, natural language processing or time series development, as well as predefined and pretrained models with Model Zoo, which can be used as a model library for quick access to user-defined model inference.

MXNet can target CPUs and GPUs both for training and inference. It is tightly integrated with Horovod, a toolkit which supports distributed CPU and GPU training.

MXNet has been developed for easy deployment. It supports natively ONNX format which is a transversal model ecosystem for the most common Deep Learning frameworks. ONNX allows conversion of any model made by one DL framework to another to perform inferences and model predictions directly.

7.5 Pandas

Pandas (Pandas, 2021) is a Python software library for data analysis and manipulation. In particular, it offers data structures and operations to manage numeric tables and temporal series. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

The main features of Pandas are:

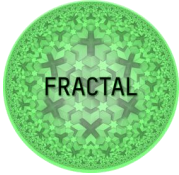
- Data uploading and saving using standard data format like CSV, TSV, Excel files and database formats.
- Simplicity in performing indexing and data aggregation operations.
- Simplicity in the execution of numerical and statistical operations.
- Simplicity in viewing the results of operations.

Three different modes are usually adopted when utilizing Pandas for data analysis:

- Convert a Python's list, dictionary or Numpy array to a Pandas data frame.
- Open a local file using Pandas, usually a CSV file, but could also be a delimited text file (like TSV), Excel, etc.
- Open a remote file or database like a CSV or a JSON on a website through a URL or read from a SQL table/database.

7.6 Scikit

Scikit-learn (Scikit Learn, 2021) is a Python library for machine learning built on top of libraries like NumPy, SciPy and matplotlib. The library provides various efficient tools for performing data pre-processing, data modelling, building reusable data pipelines and for predictive data analysis.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

The library has many in-built algorithms implemented for supervised, unsupervised and semi-supervised learning problems. This library does not support solving reinforcement learning problems but has useful tools to do model/algorithm selection.

7.7 Stable Baselines

Stable Baselines (Stable Baselines, 2021) is a Python library which consists of a set of improved implementations of reinforcement learning algorithms based on the OpenAI Baselines. This library is a refactored version of OpenAI baselines. The reinforcement learning algorithmic implementations in this library will serve as a baseline benchmark for the research community and will make it easier to replicate, refine and build up new implementations on top of the baselines.

The Stable Baselines library include popular reinforcement algorithms like Actor-Critic (A2C) methods, Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), Deep Q-Networks (DQN), Actor-Critic using Kronecker-Factored Trust Region (ACKTR), Deep Deterministic Policy Gradients (DDPG), etc. The library follows a unified interface similar to scikit-learn library and has support for training on any type of features, loading and saving algorithms.

Stable Baselines also provides a collection of pre-trained agents, the *RL Baselines zoo*, which can be used to speed-up the training process in common applications, with a simple interface to train, evaluate agents and do hyperparameter tuning.

7.8 Gym

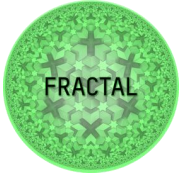
The Gym toolkit from OpenAI (Gym OpenAI, 2021) is a Python library for developing reinforcement learning algorithms. It supports training and teaching virtual agents various real-world physical actions in a specified virtual environment. The library is compatible with any numerical computation library such as TensorFlow or Theano.

The library contains a cumulative collection of test environments, that can be used to benchmark and compare when building reinforcement learning algorithms. The library provides standardization in environments which can also be used in scientific publications. It also provides an easy interface to register user-defined environments for researching on custom reinforcement learning problems.

7.9 Pytorch

Pytorch (PyTorch, 2021) is a Python external module implementing functions dedicated to machine learning, deep learning and Natural Language Processing. It is particularly useful for processing tensors using GPU acceleration of graphics cards.

The fundamental element of PyTorch are the tensors, multi-dimensional arrays of numbers, on which NumPy and much of the scientific calculation in Python are also based.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Key features and capabilities:

- Transition seamlessly between eager and graph modes with TorchScript, and accelerate the path to production with TorchServe
- Scalable distributed training and performance optimization in research and production is enabled by the torch.distributed backend
- A rich ecosystem of tools and libraries extends PyTorch and supports development in computer vision, NLP and more.
- Well supported on major cloud platforms, providing frictionless development and easy scaling

7.10 Caffe


Caffe (Berkeley Artificial Intelligence Research, 2011) is an Open-Source Deep Learning framework written in C++, and thus easily integrable into existing C++ systems. It has a Python and MATLAB interface, and can be run on both a CPU and a GPU; the switch between them is seamless and model independent. Caffe is mainly specialized in image classification and image segmentation, although it has been extended and adopted for other purposes, such as speech recognition. Caffe provides solutions for both academic research projects and industrial applications in artificial intelligence; it is employed by many universities and companies in projects which involve vision, robotics and language applications.

Caffe can work with many different types of deep learning architectures, such as Convolutional Neural Networks (CNN), Long-Term Recurrent Convolutional Network (LRCN), Long Short-Term Memory (LSTM) and Fully Connected Neural Networks (FCNNs). The framework allows for a quick introduction to machine learning and neural networks thanks to its large number of preconfigured models available.

Caffe is a fast, scalable and modular framework; its main features are:

- Speed: With a single Nvidia K40 GPU, Caffe can process over 60 million images per day.
- Expressive and modular architecture: the user can define the models and optimize them without hard-coding efforts. The framework allows switching between GPU and CPU by using a single-flag and train the ML model on a GPU machine.
- Multiple deployment options: the trained model can be deployed to mobile device platforms (Android) or commodity clusters; Caffe also runs on embedded CUDA hardware.
- Extensible code and community support: researchers and developers using Caffe have made many changes and improvements over the time, which can also be a good support for new users.

Caffe's main classes are Blob, Net, Layer and Solver: the framework uses blobs to store and process data. Blobs are N-D arrays for storing and communicating information. In Caffe, blobs can:

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

- Hold data, derivatives, and parameters.
- Shuttle between CPU and GPU: in fact, they can be thought of as an abstraction layer between the CPU and GPU: the data from the CPU is loaded into the blob, which is then passed to the GPU for computation.
- Lazily allocate memory on demand for the host and device.

The blobs are passed as input to the Layers and correspondingly output is generated. The layers have different setup options, like Convolution and Pooling, but they can also be extended to a new custom user layer implementation. Caffe uses a data structure called a directed acyclic graph for storing operations performed by the underlying layers. A typical Caffe model network starts with a data layer loading data from a disk and ends with a loss layer based on the application requirements; the model is trained by a fast and standard stochastic gradient descent algorithm. Data can be processed into mini batches which pass in the network sequentially.

Caffe also supports fine-tuning, in order to allow an existing model to be used to support new architecture or data. The previous model weights are updated for the new application and new weights are assigned wherever needed.

Built on Caffe, Caffe2 has been developed as a lightweight, scalable and modular deep learning framework and it was, subsequently, merged with PyTorch.

7.11 Darknet

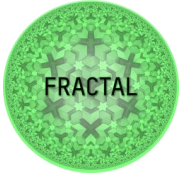
Darknet (Redmon, Darknet, 2015) is an open-source neural network framework. It is a fast and highly accurate framework (accuracy for custom trained model depends on training data, epochs, batch size and some other factors) for real-time object detection (can also be used for images). It is particularly fast thanks to the fact that it is written in C and CUDA and it can be integrated with CPUs and GPUs.

Darknet is installed with only two optional dependencies: OpenCV if users want a wider variety of supported image types or CUDA if they want GPU computation. Darknet displays information as it loads the config file and weights then it classifies the image and prints the top-10 classes for the image. Moreover, the framework can be used to run neural networks backward in a feature appropriately named Nightmare.

Advanced implementations of deep neural networks can be done using Darknet. These implementations include You Only Look Once (YOLO) for real-time object detection, ImageNet classification, recurrent neural networks (RNNs), and many others.

7.12 OpenCV

Open-Source Computer Vision Library (OpenCV) (OpenCV, 2011) is a open source library mainly devoted to artificial vision in real time. It has been originally developed by Intel and is currently released for free under the open/source Apache 2 License.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

OpenCV is written in C++ and interfaces mainly with C++, but it still retains a less comprehensive interface in C. Also interfacing with Python, Java and Matlab is allowed by means of APIs. Moreover, wrappers in several programming language have been developed during the years.

OpenCV can make use of Intel's Integrated Performance Primitives to accelerate its execution. Moreover, CUDA and OpenCL interfaces for GPUs computations have been developed.

OpenCV is currently available on several operating systems, namely Windows, Linux, macOS and the BSD OS family. Moreover, also implementation on mobile devices (Android, iOS, in particular) is allowed.

7.13 Comparison among different tools

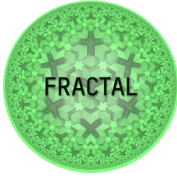
The tools considered in this chapter are listed in Table 13 together with a brief recap of their characteristics.

Tool	<i>Creator</i>	<i>OS</i>	<i>Written in</i>	<i>Interface</i>	<i>Open source</i>	<i>Algorithms</i>	<i>GPU support</i>	<i>License</i>
TensorFlow	Google Brain team	Linux, Mac OS, Windows	C++, Python	Python, C/C++	Yes	RNN, CN, RBM, DBN	Yes	Apache 2.0 ²
Keras	François Chollet (ONEIROS project)	Ubuntu, Windows, macOS	Python	Python	Yes	High level abstraction to TensorFlow methods	Yes	MIT ³
Apache TVM	Apache Foundation	Ubuntu, Windows, macOS		Python, C++, Rust, Go, Java, JavaScript	Yes	It allows deployment of models generated by other tools	Yes	Apache 2.0 ²
Apache MXNET	Apache Foundation	Linux, macOS, Windows	C++	Python, Scala, Julia, Clojure, Java, C++, R, Perl	Yes	Deep learning algorithms included in Gluon library	Yes	Apache 2.0 ²
Pandas	W.McKinney	Linux, macOS, Windows	Python	Python	Yes	Data manipulation	No	BSD ⁴

² (Apache Software Foundation, 2004)

³ Massachusetts Institute of Technology License (MIT, 1988)

⁴ Berkeley Software Distribution (Berkeley University, 1990)

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Scikit	F. Pedregosa et al.	Linux, macOS, Windows	Python	Python	Yes	Data pre-processing and standard machine learning	No	BSD ⁴
Stable Baselines	A. Raffin and A. Hill	Linux, macOS, Windows	Python	Python	Yes	Reinforcement Learning: A2C, PPO, TRPO, DQN, ACKTR	No	MIT ³
Gym	OpenAI	Linux, macOS, Windows	Python	Python	Yes	Reinforcement Learning	No	MIT ³
PyTorch	A. Paszke et al.	Linux, Android, Mac OS, iOS, Windows	Python	Python, C++	Yes	Machine learning, deep learning, Natural Language Processing	Yes	BSD ⁴
Caffe	Berkeley Vision and Learning Center	Ubuntu, Max OS X, Windows	C++, Python	C++, Python, MATLAB	Yes	Deep learning: CN and RNN	Yes	BSD ⁴
Darknet	Joseph Redmon	Linux, macOS	C, CUDA		Yes	Deep learning: Yolo	Yes	YOLO ⁵
OpenCV	Intel	Windows, Linux, Mac OS, Android, iOS	C++	C, C++, Python, Java	Yes	Computer vision, machine learning.	Yes	Apache 2.0 ²

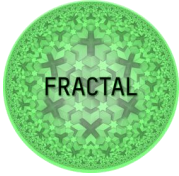
Table 13 – The tools and frameworks considered for AI implementation.

7.14 ONNX and ONNX Runtime

The Open Neural Network Exchange (ONNX) Project (ONNX, 2021) provides an open standard for machine learning interoperability. The project has been adopted as Graduated Project by the Linux AI Foundation (LF AI) and is published under MIT License via Github (Github ONNX, 2021) (<https://github.com/onnx/onnx>).

The ONNX specification describes a Machine Learning model by defining a common representation of the model computation graph and operators. The actual specification forwards a serialized format of a common computational graph and allows for model specific extension e.g., provide own operators. The flow for the ONNX standard is shown in Figure 23

⁵ (Redmon, 2016)

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

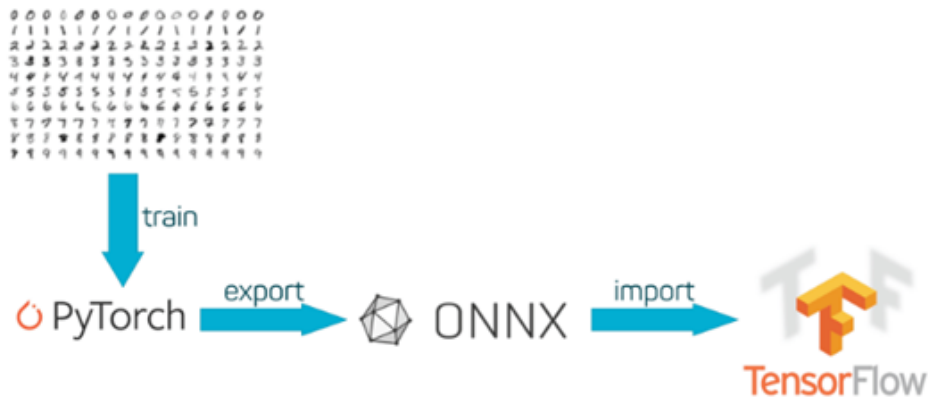


Figure 23 – The ONNX flow

To enable conversion between each ML frameworks' internal representations to and from the ONNX format the project provides converters for a large set of frameworks as available from the source repository. In this way any model from the supported frameworks can be converted into other frameworks through ONNX format as Intermediary Representation (IR). Instead of converting ONNX models to any of the supported ML frameworks for further processing the accompanying project ONNX Runtime (see Figure 24) provides direct deployment of these models for inference on various targets that serve as Execution Providers (EP).

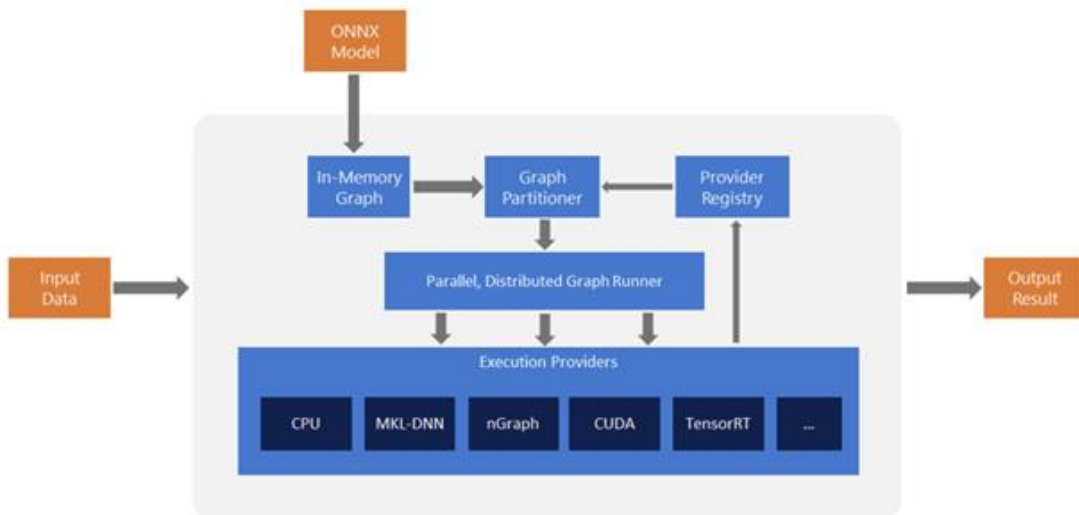
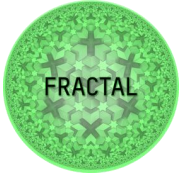
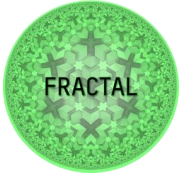


Figure 24 – The ONNX runtime

With relevance to the commercial FRACTAL node platform some few Xilinx target platforms are enabled and Vitis AI is available as execution provider. Through this the mapping of models to the Deep Learning Processing Unit (DPU) is available in principle. This DPU is the accelerator kernel to process the AI tasks. ONNX Runtime applies a number of graph optimizations on the ONNX model graph then partitions it into subgraphs based on the accelerator hardware specifics. Optimized computation kernels in core ONNX Runtime provide performance improvements and assigned

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

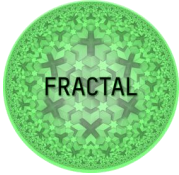
subgraphs benefit from further acceleration from the execution provider implementation.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

8 Conclusions

This deliverable has synthesized the first outcomes from T5.3, in particular as regards the collection of requirements from use cases about AI applications and the selection of methods to be used to satisfy these requirements. The selected methods are described in detail with a particular focus on the applications needed for the implementation of use cases.

It is worth noting that the deliverable should be considered as a first version of the methods description. Actually, the work of T5.3 will continue in the next months of the project, taking benefit from the progress also on other WP, mainly WP2 "Specifications and Methodology" and WP8 "Case Studies, Specification, Benchmarking & Justification File" and on other tasks of WP5, in particular T5.1 "FRACTAL AI Theory". The progress of use case specifications on one side and on the AI theory on the other side will allow a better definition of AI methods that will be included in D5.6 "Final AI methods for use case applications and mechanism for AI transparency interactions".

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

9 Bibliography

Abdulrahman, S., Tout, H., Ould-Slimane, H., Mourad, A., Talhi, C., & Guizani, M. (2021). A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond. *IEEE Internet of Things Journal*, 5476-5497.

Amazon Web Service. (2019). *AWS documentation*. Retrieved from Message Broker for AWS IoT: <https://docs.aws.amazon.com/iot/latest/developerguide/iot-message-broker.html>

Amazon Web Service. (2019). *AWS Documentation*. Retrieved from AWS IoT SDK: <https://docs.aws.amazon.com/iot/latest/developerguide/iot-sdks.html>

Amazon Web Service. (2019). *AWS Documentation*. Retrieved from Security and Identity: <https://docs.aws.amazon.com/iot/latest/developerguide/iot-security-identity.html>

Amazon Web Service. (2019). *AWS IoT Core*. Retrieved from Features: <https://aws.amazon.com/iot-core/features/>

Apache MXNet. (2021).

Apache Software Foundation. (2004). Retrieved from <https://www.apache.org/licenses/>

Apache TVM. (2021). Retrieved from <https://tvm.apache.org/>

Arlen, T. C. (2018, March 1). *Timothy Arlem*. Retrieved from medium.com: <https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3>

Barto, R. S. (2018). *Reinforcement Learning: An Introduction*.

Berkeley Artificial Intelligence Research. (2021). *Caffe*. Retrieved from <https://caffe.berkeleyvision.org/>

Berkeley University. (1990). Retrieved from [opensource.org: https://opensource.org/licenses/bsd-license.php](https://opensource.org/licenses/bsd-license.php)

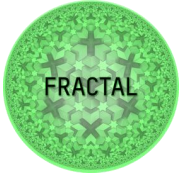
Bochovski, A., Wang, C.-Y., & Mark, H.-Y. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. Retrieved from arXiv:2004.10934

Bosch. (2019). *Gateway software*. Retrieved from <https://www.bosch-si.com/iot-platform/iot-platform/gateway/software.html?ref=ot-2-inst-de-2017h1-sales-contact-forms-iot-platform>

Brownlee, J. (2019, January 11). *Machine Learning Mastery*. Retrieved from How to fix the Vanishing Gradients Problem Using the ReLU: <https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/>

Comer. (2000). Datagram Size, Network MTU, and Fragmentation. In *Sect. 7.7.4* (p. p. 104).

Cortes, C., & Vapnik, V. (1995). Support Vector Machine. *Machine Learning*, 20(3), 273-297.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

D. Vidhate, P. K. (2017). Cooperative Multi-agent Reinforcement Learning models (CMRLM) for intelligent traffic control. *1st International Conference on Intelligent Systems and Information Management (ICISIM)*.

Dey, A. K. (2000). *Providing Architectural Support for Building Context-Aware Applications*. Ph.D. Thesis, College of Computing, Georgia Institute of Technology.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. Irvine: University of California.

Github ONNX. (2021). Retrieved from <https://github.com/onnx/onnx>

Graf, S., Herbig, T., Buck, M., & Schmidt, G. (2015). Features for voice activity detection: a comparative analysis. *EURASIP Journal on Advances in Signal Processing*, 91. doi:<https://doi.org/10.1186/s13634-015-0277-z>

Gupta, M. (2020). *YOLO — You Only Look Once*. Retrieved from [towardsdatascience.com: https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4](https://towardsdatascience.com/https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4)

Gym OpenAI. (2021). Retrieved from <https://gym.openai.com/>

Hui, J. (2018, March 18). *Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3*. Retrieved from <https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088>

Jordan, J. (2017, July 18). *Federated Learning: Challenges, Methods, and Future Directions*. Retrieved from [jeremyjordan.me: https://www.jeremyjordan.me/neural-networks-training/](https://www.jeremyjordan.me/neural-networks-training/)

Kaiqing Zhang, Z. Y. (2021). *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. Springer.

Kamal, A. (2019, August 3). *Amro Kamal*. Retrieved from [medium.com: https://amrokamal-47691.medium.com/yolo-yolov2-and-yolov3-all-you-want-to-know-7e3e92dc4899](https://amrokamal-47691.medium.com/yolo-yolov2-and-yolov3-all-you-want-to-know-7e3e92dc4899)

Keras. (2021). Retrieved from <https://keras.io/>

Lampkin, V., Tat Leong, W., Olivera, L., Rawat, S., Subrahmanyam, N., Xiang, R., . . . Locke, D. (2012). *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*. IMB Redbooks.

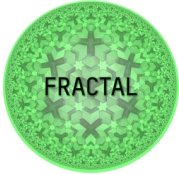
Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*, 37, 50-60.

Marco Wiering, R. S. (1999, January). Reinforcement Learning Soccer Teams with Incomplete World Models. *Autonomous Robots*.

MIT. (1988). Retrieved from <https://opensource.org/licenses/mit-license.php>

ONNX. (2021). Retrieved from onnx.ai

OpenCV. (2021). Retrieved from <https://opencv.org/>

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Paley, A., Urma, R.-G., & Lawrence, N. D. (2020). *Challenges in Deploying Machine Learning: a Survey*.

Pandas. (2021). Retrieved from <https://pandas.pydata.org/>

PyTorch. (2021). Retrieved from <https://pytorch.org/>

Qingrui Zhang, H. D. (2020, 11 25). Lyapunov-Based Reinforcement Learning for Decentralized Multi-agent Control. *International Conference on Distributed Artificial Intelligence*.

Redmon, J. (2016). Retrieved from <https://github.com/pjreddie/darknet/blob/master/LICENSE>

Redmon, J. (2016). You only look once: Unified, real-time object detection. *IEEE Conference on Computer Vision and Pattern Recognition*. Retrieved from arXiv:1506.02640. Bibcode:2015arXiv150602640R.

Redmon, J. (2021). *Darknet*. Retrieved from <https://pjreddie.com/darknet/>

Redmon, J., & Farhadi, A. (2016). *YOLO9000: Better, Faster, Stronger*. Retrieved from <https://arxiv.org/pdf/1612.08242.pdf>

Redmon, J., & Farhadi, A. (2017). *YOLO9000: Better, Faster, Stronger*. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 7263-7271).

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. Retrieved from arXiv:1804.02767

Sabra, A. (2021, February 16). Retrieved from Towards Data Science: <https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8>

Saint-Andre, P., Smith, K., & Tronçon, R. (2009). *XMPP: The Definitive Guide. Building Real-Time Applications with Jabber*. O'Reilly Media, Inc.

Sato, D., Wider, A., & Windheuser, C. (2019). *Continuous delivery for machine learning*. Retrieved from <https://martinfowler.com/articles/cd4ml.html>

Scikit Learn. (2021). Retrieved from <https://scikit-learn.org/stable/>

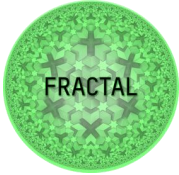
Shelby, Z. a. (2011). *6LoWPAN: The wireless embedded Internet* (Vol. 43). John Wiley & Sons.

Siemens. (2018). <https://www.plm.automation.siemens.com>. Retrieved from https://www.plm.automation.siemens.com/media/global/en/Siemens-MindSphere-Whitepaper-69993_tcm27-29087.pdf

Stable Baselines. (2021). Retrieved from <https://stable-baselines.readthedocs.io/en/master/>

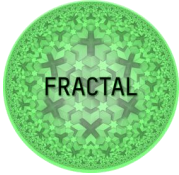
TensorFlow. (2021). Retrieved from <https://www.tensorflow.org/>

UbuntuPit. (2018). <https://www.ubuntupit.com/>. Retrieved from <https://www.ubuntupit.com/choose-the-right-iot-platform-top-20-iot-cloud-platforms-reviewed/>

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Uhle, C., & Bäckström, T. (2017). Voice Activity Detection. In *Speech Coding. Signals and Communication Technology*. Springer.

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., & Gao, Y. (2021). A survey on federated learning. *Knowledge-Based Systems*.

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

10 List of Figures

Figure 1 – Functioning of the online training for the FRACTAL node	11
Figure 2 – Learning Approaches in FRACTAL: (left) Centralized Learning; (centre) Decentralized Learning; (right) Federated Learning.....	12
Figure 3 – Common functional silos in large organizations can create barriers, stifling the ability to automate the end-to-end process of deploying ML applications to production. Picture taken from (Sato, Wider, & Windheuser, 2019).....	22
Figure 4 – Continuous Delivery for Machine Learning end-to-end process (Sato, Wider, & Windheuser, 2019).	23
Figure 5 – Classic structure of a CNN for image recognition	39
Figure 6 – Action of a convolutional kernel over a 4x4 input: this is how a convolutional layer works inside a CNN.	40
Figure 7 – Convolution between an input matrix 8x8 with a kernel filter 3x3. The output is a matrix 6x6.....	40
Figure 8 – Example of pooling	41
Figure 9 – Forward and backward pass in backpropagationKeep in mind that the forward propagation computes the result of an operation and save any intermediates needed for gradient computation in memory. Backward: apply the chain rule to compute the gradient of the loss function with respect to the inputs.	42
Figure 10 – The forward pass on the left calculates z as a function $f(x,y)$ using the input variables x and y. The right side of the figures shows the backward pass. Receiving dL/dz , the gradient of the loss function with respect to z from above, the gradients of x.....	42
Figure 11 – Bounding boxes and class probability map. Source: (Redmon, You only look once: Unified, real-time object detection, 2016)	44
Figure 12 – IoU operation	44
Figure 13 – The effect of non-max suppression in bounding box identification. Source: (Gupta, 2020)	45
Figure 14 – YOLO Architecture (source: You Only Look Once: Unified, Real-Time Object detection).....	45
Figure 15 – Confusion matrix structure for binary classification problems	46
Figure 16 – Some theoretical precision-recall curves	47
Figure 17 – Differences between YOLO and YOLO v2 (source (Redmon & Farhadi, YOLO9000: Better, Faster, Stronger, 2017)	48
Figure 18 – YOLO with Darknet-19.....	48
Figure 19 – The performances of the different YOLO versions on the MS COCO benchmark. Source (Bochovski, Wang, & Mark, 2020).....	49
Figure 20 – The agent–environment interaction in a Markov decision process. (Barto, 2018).....	53
Figure 21 – The multi-agent interaction with a shared environment in a Markov game setup. (Kaiqing Zhang, 2021)	54
Figure 22 – Apache TVM Blog - Bringing AMD GPUs to TVM Stack and NNVM Compiler with ROCm.	58
Figure 23 – The ONNX flow	65

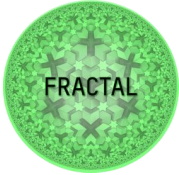
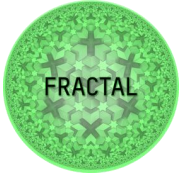
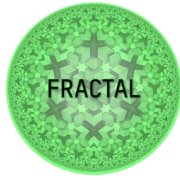
	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

Figure 24 – The ONNX runtime.....65

	Project	FRACTAL		
	Title	Specification of AI methods for use case applications		
	Del. Code	D5.1		

11 List of Tables

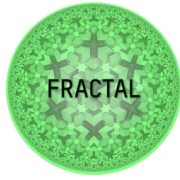
Table 1 – Document History	6
Table 2 – Different categories of algorithms in Machine Learning	17
Table 3 – Considerations, issues and concerns explored in (Paleyes, Urma, & Lawrence, 2020).	21
Table 4 – Requirements for Use Case 1 – Demonstrator 1	25
Table 5 – Requirements for Use Case 1 - Demonstrator 2	26
Table 6 – Requirements for Use Case 2	28
Table 7 – Requirements for Use Case 3	29
Table 8 – Requirements for Use Case 4	30
Table 9 – Requirements for Use Case 5	33
Table 10 – Requirements for Use Case 6	34
Table 11 – Requirements for Use Case 7	36
Table 12 – Requirements for Use Case 8	37
Table 13 – The tools and frameworks considered for AI implementation.	64



Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

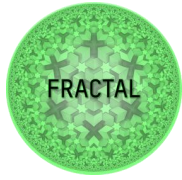
12 List of Abbreviations

A2C	Actor-Critic
ACKTR	Actor-Critic using Kronecker-Factored Trust Region
BPF	Band Pass Filter
BSD	Berkeley Software Distribution
CIoU	Complete Intersection over Union
CmBN	Cross mini Batch Normalization
CMRLM	Cooperative Multi-Agent Reinforcement Learning Models
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSP	Cross Stage Partial connections
CSV	Comma Separated Values
DDPG	Deep Deterministic Policy Gradients
DDR	Double Data Rate
DeepRL	Deep Reinforcement Learning
DevOps	Development Operations
DNN	Deep Neural Network
DQN	Deep Q-Networks
EDDL	European Distributed Deep Learning
EP	Execution Provider
ERI	Energy Ratio Index
FN	False Negative
FCNN	Fully Connected Neural Network
FLOPS	Floating point Operation Per Second



Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

fps	frames per second
Gbit	Gigabit
GFLOPS	Giga Floating-Point Operations Per Second
GPU	Graphical Processing Unit
HDMI	High-Definition Multimedia Interface
HW	Hardware
IoT	Internet of Things
IoU	Intersection over Union
IR	Intermediary Representation
LF AI	Linux AI Foundation
LRCM	Long-Term Recurrent Convolutional Network
LSTM	Long Short-Term Memory
mAP	mean Average Precision
MARL	Multi-Agent Reinforcement Learning
MASAC	Multi-Agent Soft-Actor Critic
MDP	Markov Decision Process
MFCC	Mel-Frequency Cepstral Coefficient
MIT	Massachusetts Institute of Technology
ML	Machine Learning
MLI	Maximum Likelihood Index
MLOps	Machine Learning Operations
ms	millisecond
NLP	Natural Language Processing
OAA	One-Against-All
OAO	One-Against-One



Project	FRACTAL		
Title	Specification of AI methods for use case applications		
Del. Code	D5.1		

ONNX	Open Neural Network Exchange
OpenCV	Open-Source Computer Vision Library
OS	Operating System
PM	Probability Matrix
PPO	Proximal Policy Optimization
RAM	Random Access Memory
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SAT	Self-Adversarial Training
SFI	Smart Flatness Index
TP	True Positive
TRPO	Trust Region Policy Optimization
TPU	Tensor Processing Unit
TSV	Tab Separated Values
URL	Uniform Resource Locator
USB	Universal Serial Bus
VAD	Voice Activity Detection
WRC	Weighted Residual Connections
YOLO	You Look Only Once