Scientific Report

AI Folk – Resource Management for Distributed AI

2022

Contents

1 Introduction

As machine learning (ML) continues to permeate every aspect of daily life, gathering datasets and training state-of-the-art models becomes more and more the privilege of large technology companies. Recent approaches – such as *federated learning* [Li et al., 2020] and *edge AI* [Wang et al., 2019] – propose the decentralization of learning, such that model training happens on devices which are closer to the user, using that which resides on those devices. One of the core challenges of federated learning is heterogeneity of devices and openness of the distributed AI system [Li et al., 2020].

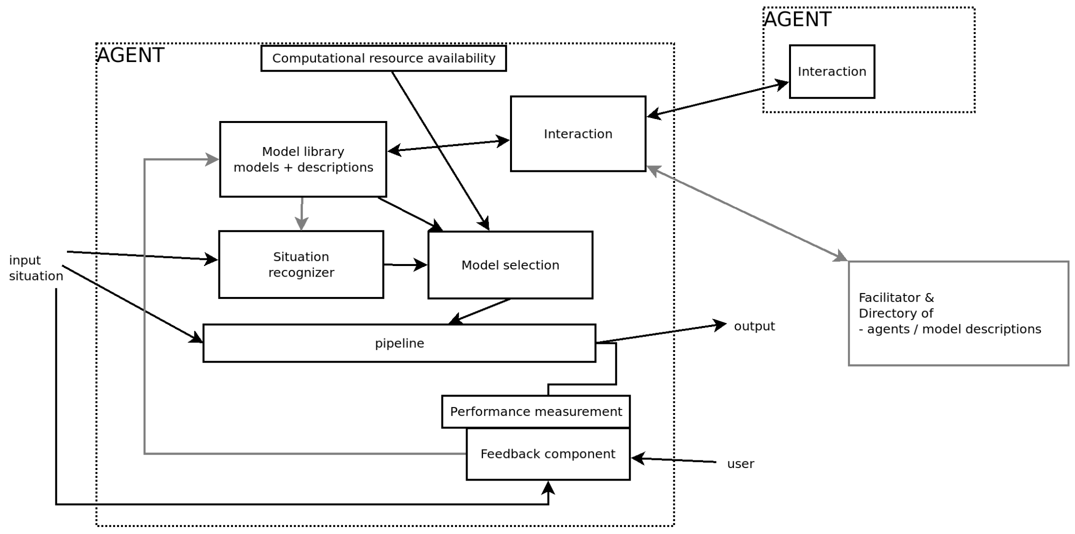

Figure 1: AI Folk entity architecture.

Our research goal is to develop a knowledge model and an interaction protocol which allow a system formed of multiple actors – human, software, or organizational – to find, use and share improvements on machine learning resources. Such resources can be datasets, models, or experiences.

This project aims to develop the AI Folk framework and methodology, at the intersection of machine learning, knowledge management, and multi-agent systems. It comprises tools and methods that allow the management and discovery of ML-related resources in a distributed system.

Concrete objectives of the AI Folk project are development of the AI Folk ontology, the AI Folk interaction protocol, the AI Folk methodology, and the implementation of two proof-of-concept scenarios set in the domains of autonomous driving and disaster response.

The goal of the AI Folk ontology is to describe ML models, training datasets, but also experiences with trained models, with the objective that agents be able to select an appropriate model, depending on their concrete and present situation. This requires that the agent is able to match the description of the model to the description of its own situation.

The AI Folk interaction protocol must enable agents to communicate in order to search for and exchange ML models. It must support the transfer of large amounts of data, as some models can be large. It also must allow for the discovery of other agents and of models that may be useful for the current situation.

Certainly, for each domain of study, the ontology and the interaction protocol may need to have some elements which are domain-specific. The AI Folk methodology must define what parts of the AI Folk resources are generic and how one can adapt AI Folk resources to new domains.

1.1 Concepts and Architecture

A primary approach to the architecture of AI Folk entities is shown in Figure 1. An AI Folk entity holds a *library of models* that it uses or it may potentially use. Each of the models has a description using the AI Folk ontology.

The sum of the inputs that the entity can percieve can be used to detect the *situation* of the entity. For instance, in the autonomous driving scenario, the situation may consist of the time of day (night, sunrise/sunset, daylight), weather conditions (clear, fog, rain, snow), types of objects visible. The situation recognizer itself may use ML models to recognize the situation.

Based on the situation, the entity selects models which are appropriate for the situation and for the computational resources available to the entity. These models are assembled into a pipeline, which produces the useful output of the entity.

The entity may measure the performance of the models that is used, and may able to evaluate, or receive from a human user, or from another entity, a feedback on the quality of the output. The entity records this output in the model library, in the description of the models that it used.

The entity is able to interact with other entities via an interaction module implementing the AI Folk protocol, in order to update its model library and share with others its experience with the models.

2 AI Folk Ontology

2.1 Related Work

The paper [Klampanos et al., 2019] introduces a generic topology that allows the description of neural models, the algorithms used for training and the evaluation metrics. The ontology design proposed by the authors of this paper connects three defining aspects of a solution based on deep learning:

- **Topology** – The connectivity of complex, multi-network, configurations;
- **Training** – The algorithms used for training complex ANN configurations;
- **Evaluation** – Quantitative performance information.

Figure 2 shows the architecture of this ontology which contains 160 classes, 50 object properties and 32 data properties. The architecture of this ontology was designed to allow the description of the topological perspective, of the training or the evaluation characteristics of some complex configurations that may contain multiple Artificial Neural Networks.

The paper [Ayranci et al., 2022] highlights the benefits that an ontology can bring in terms of the explanation part of the agnostic predictions that a neural model can make. The general architecture of the flow implemented for this approach is presented in Figure 3. To introduce ontology-extractable concepts into this pipeline, the authors introduce a new ontology-based sampling technique. In order to learn the local behavior of the function f in its vicinity, the authors propose the approximation of $L(f, g, \phi_x)$ by extracting samples based on x, from the ontology, with the indicated proximity ϕ_x. They demonstrated the usefulness of this proposed method by introducing two extensive experiments performed on two datasets with real data.

Recently, more and more neural models have appeared that can solve various types of problems. In general, they are based on the concept of supervised learning, which assumes the existence of a data set used in the training process to help the neural network discover various relevant patterns and use in the calculation of the prediction. Thus, it has become a priority to propose a search algorithm able to identify the appropriate neural model for a dataset or a problem. Such a search service was proposed in the paper [Nguyen and Weller, 2019]. Figure 4 presented the pipeline implemented for this search service. The main goal of this work was to support people who use solutions in the field of deep learning, making existing architectures easier to find and reuse. In addition to the proposed search service, they also created a data set, the FAIRnets dataset, which contains more than 500 neural

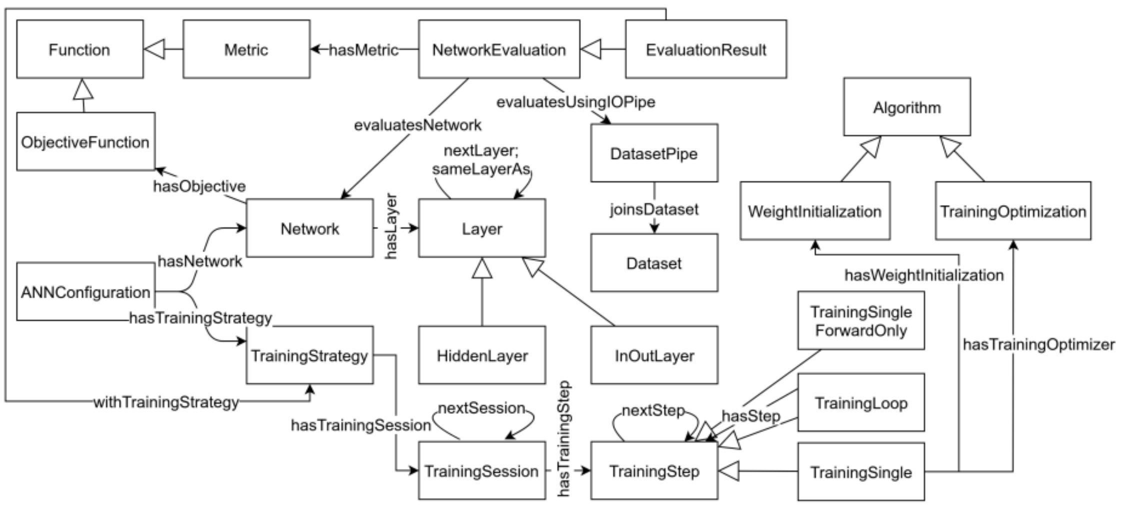

Figure 2: The ANNETT-O ontology architecture proposed by [Klampanos et al., 2019]

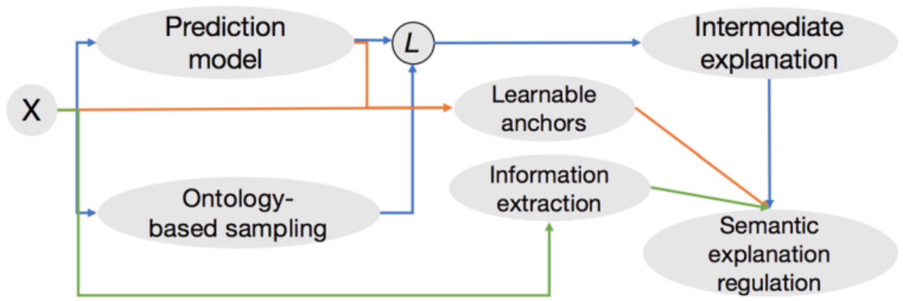

Figure 3: The flow that demonstrates how a semantic explanation can be obtained for the prediction of a neural model starting from an ontology

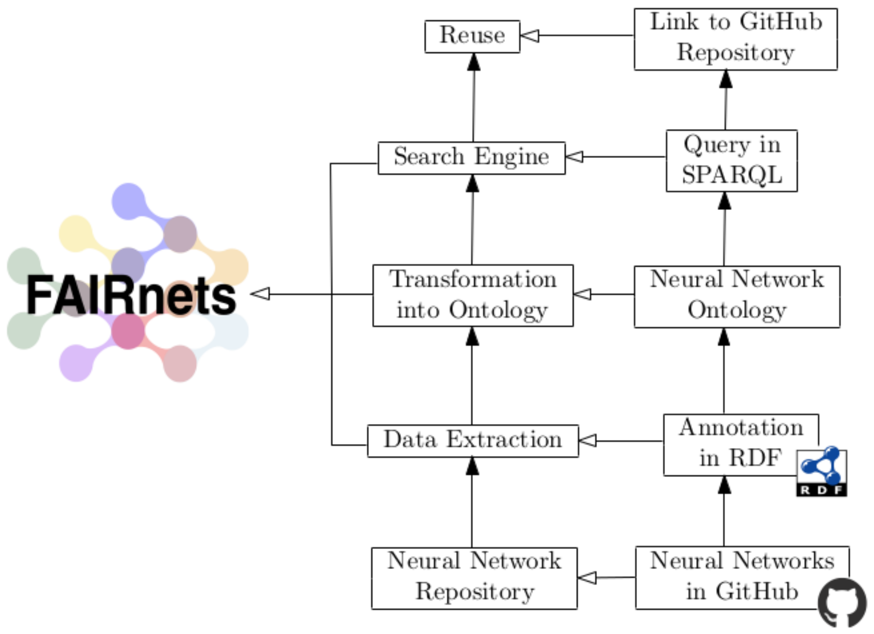
Reuse
Link to GitHub Repository
Search Engine
Query in SPARQL
FAIRnets
Transformation into Ontology
Neural Network Ontology
Data Extraction
Annotation in RDF
RDF
Neural Network Repository
Neural Networks in GitHub

Figure 4: The proposed pipeline for the search algorithm intended to facilitate the reuse of neural models

models.

2.2 Challenges and Questions

Recently, researchers have started to use more and more types of neural networks to solve various problems with practical applicability. In general, the proposed approaches are focused on specific problems, and the results obtained are closely correlated with the existing data sets. However, it is not easy to identify which is the most appropriate approach that could be used for a problem depending on the quality and type of data used in the solving process. We propose that by designing an ontology that models the properties of a neural model, we can develop a search strategy for the most suitable type of network.

The construction of such an ontology is challenging because several aspects must be taken into account. Among the most important are those related to the architectural properties of the neural network, the statistics that can describe the data set used for training and the hyperparameters used in the model training process.

In order to develop the AI Folk ontology, we have tried to understand which are the specific challenges in deployments such as the ones envisoned by the AI Folk project, as well as in autonomous driving scenarios and most specifically semantic image segmentation tasks, which are among the most challenging in such scenarios. We have hence developed a list of *competency questions* which the description of a machine learning model *is supposed to answer*. The list of questions is the following:

- General Dataset Information Questions
  1. What is the dataset name?
  2. What is the dataset access URL?

3. What is the dataset human readable description?
4. When was the dataset created?
5. When was the dataset last updated?

- General Domain description Questions
  The *domain* is defined as an *intended principal / high-level* usage of the dataset (e.g. Autonomous Driving, Search and Rescue, Activity Recognition).
  A *Task* is defined as a *learnable objective* that can be addressed using the dataset (e.g. Semantic Segmentation of drivable road and driving obstacles, Semantic Segmentation of terrain types)
  1. What is the *domain* of the dataset?
  2. What *tasks* can be learned from the dataset?
- Semantic Segmentation Task Questions
  1. What *type* (e.g. image, 3D point-cloud) of input does the segmentation task have?
  2. What is the *size* (as tensor dimensions) of the segmentation task input?
  3. How many target classes does the segmentation task have? (e.g. drivable lane, opposite lane, traffic participant, pedestrian, other)
  4. What are the target classes for the segmentation task?
- Semantic Segmentation Task for Autonomous Driving
  Scene Categories define a highlevel classification of the driving scene (e.g. residential house-only, residential block-only, residential mixed, urban high-rise, single lane highway, multi lane highway, country side, mountain road)
  1. What are the illumination conditions for driving scenes present in the dataset? What is the percent-wise coverage of each illumination condition in the dataset?
  2. What are the weather conditions for driving scenes present in the dataset? What is the percent-wise coverage of each weather condition in the dataset?
  3. What are the *scene categories* present in the dataset? What is the percent-wise coverage of each *scene category* in the dataset?
  4. How many lanes per driving direction do roads in the dataset have?
  5. What is the min / max / avg number of traffic participant obstacles per frame?
  6. What is the min / max / avg number of pedestrians per frame?
  7. What is the min / max / avg number of cross intersections per frame?
  8. What is the min / max / avg number of T-shaped intersection per frame?
  9. What percentage of images in the dataset contain road section without any lane markings?
  10. What percentage of images in the dataset contain parking lot scenes?
  11. What is the min / max / avg *ratio* of the drivable road segmentation mask with respect to the input image resolution?
  12. What is the min / max / avg *ratio* of the perdestrian segmentation mask with respect to the input image resolution?
  13. What is the min / max / avg *ratio* of the traffic participant segmentation mask with respect to the input image resolution?

2.3 A first version for the ontology

The ontology proposed by us contains concepts grouped into 5 essential categories:

1. **Problem description** – represents the description of the problem to be solved using a neural model. For each problem, we will define the category it belongs to, information about the evaluation metric used to evaluate the solution of such a problem and characteristics of the data that we can use as input.
2. **Model description** – We can describe the neural model both from an architectural point of view and from the perspective of the input used and the output generated.
3. **Dataset description** – For problems that can be solved using neural networks, we need to

Table 1: Elements in the AI Folk ontology (continued in Table 2)

Problem description

1. **Category**
   (a) Classification
   (b) Regression
   (c) Segmentation
   (d) ObjectDetection
   (e) Clustering
   (f) Adversarial
   (g) Discrimination
   (h) Generation
   (i) Prediction
   (j) Estimation
   (k) Reconstruction
2. **EvaluationMetric**
3. **DataModality**

Dataset description

1. Name
2. URL
3. CreationTime
4. Domain
5. Application/ possible tasks
6. Number of images
7. Input type (video, image, lidar etc)
8. Number of classes (if applicable)
9. Light conditions
10. Weather conditions
11. Area recorded
12. Number of traffic participants
13. Number of cars
14. Number of pedestrians
15. Keywords
16. Citation
17. Size

Model description (A)

ANNArchitecture:
1. PretrainedNetwork
2. InputSize
3. OutputSize
4. Layer
   (a) ConvolutionLayer
      i. Conv1d
      ii. Conv2d
      iii. Conv3d
      iv. ConvTranspose1d
      v. ConvTranspose2d
      vi. ConvTranspose3d
      vii. Unfold
      viii. Fold
   (b) Poolinglayer
      i. MaxPool1d
      ii. MaxPool2d
      iii. MaxPool3d
      iv. MaxUnpool1d
      v. MaxUnpool2d
      vi. MaxUnpool3d
      vii. AvgPool1d
      viii. AvgPool2d
      ix. AvgPool3d
      x. AdaptiveMaxPool1d
      xi. AdaptiveMaxPool2d
      xii. AdaptiveMaxPool3d
   (c) PaddingLayer
      i. ReflectionPad1d
      ii. ReflectionPad2d
      iii. ReflectionPad3d
      iv. ReplicationPad1d
      v. ReplicationPad2d
      vi. ReplicationPad3d
      vii. ZeroPad2d
      viii. ConstantPad1d
      ix. ConstantPad2d
      x. ConstantPad3d

use a data set. It is necessary to have a clear description of this dataset to establish how its characteristics influence the results of the neural model.

4. **Training description** – Training the neural model is the most important process, because it decisively influences the results of the model.
5. **Requirements** – When we want to take over a pre-trained neural model to use it for another problem or another set of data, it is important to know the existing requirements from a hardware perspective and the software one: libraries and frameworks.

Figure 5 highlights the relationships that exist between these 5 categories of the proposed ontology. The ontology contains the concepts enumerated in the remainder of this section.

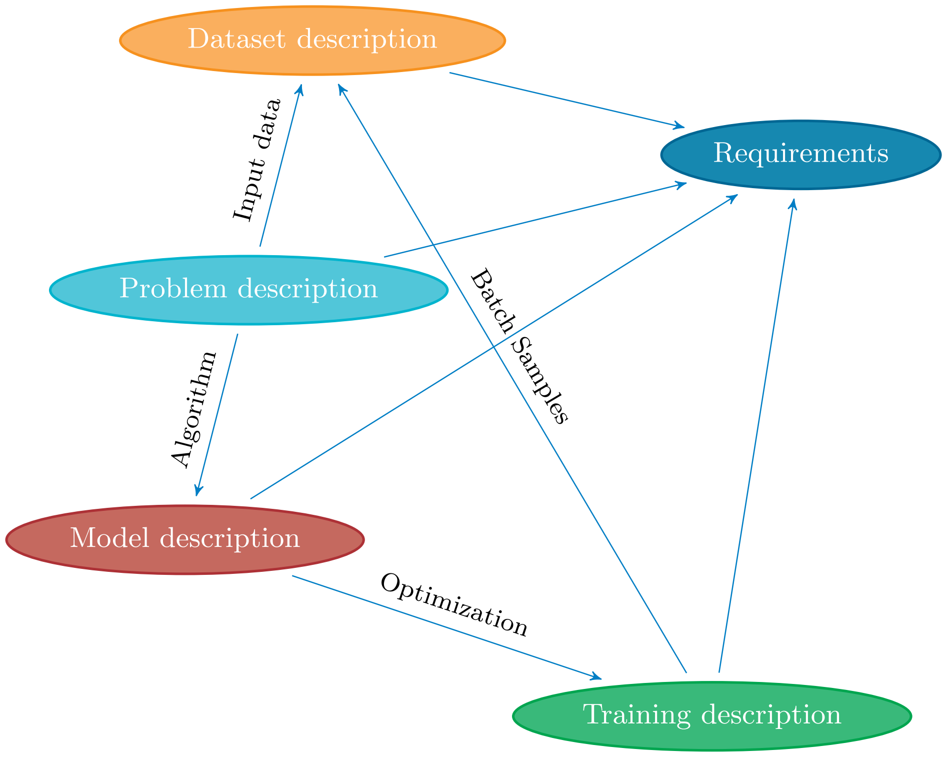

Figure 5: The structure of the ontology from the perspective of the proposed categories

Table 2: Elements in the AI Folk ontology (continued from Table 1)

Requirements	Model description (B)
1. Operating System	(d). NormalizationLayer
2. Language	(i) BatchNorm1d
3. Has Installation Script (to be executed instantly)	(ii) BatchNorm2d
	(iii) BatchNorm3d
4. Dependent libraries	(iv) GroupNorm
5. MLFramework	(v) InstanceNorm1d
6. ModelSize	(vi) InstanceNorm2d
7. GPU	(vii) InstanceNorm3d
8. RAM size	(e). RecurrentLayer
9. Memory size	(i) RNN
	(ii) LSTM
	(iii) GRU
	(iv) RNNCell
	(v) LSTMCell
	(vi) GRUCell
	(f). TransformerLayer
	(i) Transformer
	(ii) TransformerEncoder
	(iii) TransformerDecoder
	(iv) TransformerEncoderLayer
	(g). LinearLayer
	(i) Identity
	(ii) Linear
	(iii) Bilinear
	(h). DropoutLayer
	(i) Dropout
	(ii) Dropout1d
	(iii) Dropout2d
	(iv) Dropout3d
	(i). SparseLayer
	(i) Embedding
	(ii) EmbeddingBag
	(j). VisionLayer
	(i) PixelShuffle
	(ii) PixelUnshuffle
	(iii) Upsample
	(iv) UpsamplingNearest2d
	(v) UpsamplingBilinear2d

3 AI Folk Interaction Protocol

One of our goals for the project is to develop the specification of a high-level interaction protocol, that allows actors in the system to search for resources, transfer resources (datasets, models) locally in a manner that is efficient (using compression and incremental updates) and customized (e.g. dimensionality reduction), subscribe to updates, and share experiences with the resources (especially learning models).

3.1 Software and communication infrastructure

In order to implement and test a proof-of-concept version of the AI Folk framework implementing the elements of the AI Folk methodology, we will use an existing infrastructure for the deployment of multi-agent and multi-entity systems, called Flash-mas [Olaru et al., 2019]. Flash-mas is developed in University Politehnica of Bucharest and is a capable MAS deployment framework with the end goal of being comparable and surpassing the possibilities offered by the current most popular MAS framework, Jade.

Flash-mas defines a multi-entity system as a set of *entities* – computing processes with the property that they are persistent over a longer amount of time, they make decisions autonomously, and they interact in order to fulfil their goals. Entities interact wmong each other via messages, each message having a source, a destination, and one or serveral pieces of content.

One advantage that Flash-mas has over other other similar frameworks is that all elements in the framework are explicitely represented as entities in the model of the system, whereas in other frameworks only some of them are, e.g. agents, or agents and artifacts. As such, in Flash-mas there are several standard entities: *nodes* manage the deployment of the framework on every machine; *support infrastructures* are virtual entities that span multiple nodes and offer interaction services to other entities, and are represented on nodes by *pylons*; *pylons* represent support infrastructures on nodes, and entities can interact directly with the pylons; *agents* are autnonomous entities which act in the environment in order to fulfil their goals; *artifacts* offer to other entities specific tools and means to interact with the physical environment; and *shards* are sub-agent entities which implement specific behaviors and functionalities and can be integrated in agents, as well as in other entities. But these are not the only types of entities that can exist in a Flash-mas deployment; on the contrary, any type of entity can be added to a system, and any entity can interact with any other, based on the identifiers of entities.

The main advantage of Flash-mas is its flexibility in modeling and deployment. Even for standard entities, the developer can choose from multiple implementations or can develop new ones, provided a small set of methods are available in the implementing class. This is especially useful when it comes to interaction. There are already several support infrastructures implemented in Flash-mas which offer communication services, among which Websocket-based, webservice-based, using ROS, MPI and others. The modularity of the framework makes it that the code of the entities doesn't need to change in any way when a different support infrastructure is used, and implementations can even be changed at runtime.

The most appropriate means of communication between entities for the AI Folk approach is interaction via web services, where on each node in the network there is a lightweight web server which responds to http requests. This allows for a RESTful approach to communication.

Flash-mas is implemented in Java, but work has already been done in integrating it with entities written in Python, to serve for intarction with Machine Learning frameworks such as PyTorch[1] and Keras[2].

3.2 Names and Identifiers

In any multi-entity system, there is an important need for being able to identify things. In AI Folk there are two elements that need to be idenfiable accross the system: entities and models. At this

[1]PyTorch `https://pytorch.org/`
[2]Keras: the Python deep learning API `https://keras.io/`

point we assume datasets as not chainging and, moreover, we consider they are not stored inside the AI Folk system. Hence, their identification is trivial and can be performed, for instance, by means of their URI.

Entities need identifiers so that they can be addressed as destinations of messages. Identification must work in the context of entities which are bound to mobile devices (cars, mobile robots, smartphones), and in the context of a changing network topology. Considering our work in developing the webservice-based support infrastructure for FLASH-MAS, it would be easiest to identify entities using their home server URI, concatenated with an identifier specific to the entity and unique to the home server. The home server paradigm [Wojciechowski, 2001] enables a distrubted approach, while managing parts of the system in a centralized manner – all entities are assigned to a home server which knows where the entity is located. If a device is routable (from an Internet Protocol point of view), then it can serve as a home server for all entities that execute on that device.

Machine learning models also need to be identified. In a deployment in which entities search for, exchange and share ML models, entities must be able to refer to those models. An additional challenge comes from the fact that models can be tweaked by individual entities, leading to slight variations in the model. We propose the following identification mechanism.

Every model is identified by a *hash number*, providing a unique identifier. Moreover, in order to be able to differentiate between models which are completely different and models which are only slight varations of other models, we build a conceptual *model forest* of hashes, as follows:

- there is one *tree* of hashes in the forest for each major model existing in the system (including its variations and tweaks);
- there is a *node* in the tree of a model for each concrete implementation of the model. In the tree structure, the node only contains the hash of the model. For a node corresponding to a specific model, the children of that node correspond to models which are directly derived from that model. Edges optionally contain summary information about the variation which has been applied.

Of course, in a large system, there will be no single entity which holds the *entire* model forest of hashes. However, individual entities using ML models will find useful to store, for each of the models they use, the ancestor line and potentially other parts of the hash tree. Model libraries will also store complete or almost-complete trees for the models they store.

3.3 Protocol Specification

In order to allow entities in an AI Folk ecosystem to discover and exchange ML-related resources, including ML models, we developed the protocol presented in this section. The protocol contains three segments: discovery, transfer, and update. The protocol is high-level and does not cover issues such as message routing or conversation control.

Discovery is the process through which an entity can search for (1) ML models adequate for the situation and needs of the entity and (2) information on the experiences other entities had with using specific models in specific situation.

An entity will be able to search for a model according to a given SPARQL query, and in a given area around its location (in terms of phyisical space or in term of in terms of network space). An entity will also be able to query the experience with a particular ML model; there the variables are the ara to search (in physical or network space), but also the maximum *subtree distance* in the tree of models – that is, the entity may search for experience data not only with a specific model, but also with closely

Message type	Message arguments		
DISCOVERY segment			
SEARCH	Query	Area	
EXPERIENCE SEARCH	#modelHash	Area	Subtree distance
LISTING	Query id / #modelHash		
TRANSFER segment			
REQUEST transfer	#modelHash	[chunk size]	[chunk no]
TRANSFER	#modelHash	chunk size	chunk no
UPDATE segment			
SUBSCRIBE	#modelHash	Subtree distance	
SUBSCRIBE	Query		
UNSUBSCRIBE	#modelHash / Query id		
MODEL UPDATE	#modelHash	#newModelHash	update description
REQUEST diff	#modelHash	#newModelHash	
TRANSFER diff	#modelHash	#newModelHash	

Figure 6: Types of interactions in the AI Folk protocol.

related models. Here, by experience we signify performance metrics related to the use of models in a particular situation. The destination of the search will reply with a listing of results.

Transfer is the process through which entities can transfer among each other entire ML models, based on a peer-to-peer interaction. Since some models may be large, they may be transferred in chunks, and the transfer of a particular chunk can be (re-)requested individually, which may be useful especially in scenarios with poor network reliability.

Update is the process through which entities may subscribe for and receive updates on models of interest or queries of interest. Since some changes to models may be applied incrementally, an entity is able to request information on how to update its own model in order to obtain another version, without the need to transfer the entire new version.

4 Autonomous Driving Scenario

The first scenario that we will consider in this project is regarding an autonomous car that drives in an autonomous environment, with many autonomous cars that can be considered as agents in a distributed system. We will describe the scenario in detail, we will analyze the components and the associated models and finally we will describe the validation design made for this scenario.

4.1 Scenario description

The autonomous driving scenario works as following:

1. An autonomous car drives in a familiar environment, without having to worry about exchanging information.

2. At one point, some variables change - for example, the weather (from sunny to rain), the light (from day to dusk or even night conditions, in a tunnel), the city, the road (from highway to national road or vice versa), the driving scene (a parking lot, an university campus). Because many autonomous driving components are based on trained models with various data, that can contain only images recorded during the day or in a specific city, the autonomous car should update its algorithms.
3. The autonomous car will try to connect other cars that are connected in the environment, in a peer-to-peer fashion.
4. The autonomous car will exchange information about the desired models and if the hardware and software requirements are met, it will update the algorithm accordingly.

4.2 Description of Models

An autonomous car contains both specialized hardware (GPS, video cameras, LiDAR, radar, etc) and specialized software. There are many components that have to be considered when talking about an autonomous cars:

1. localization
2. Perception - this is a crucial component, which involve both the software and the hardware and has many subcomponents - object detection and tracking, lane detection, semantic segmentation and depth estimation
3. Prediction - trajectory prediction for the surrounding cars and pedestrians
4. Planning - first, a global planning for the path from A to B, then a local planning to navigate in the current situation (a maneuver planning - continue straight, turn left, turn right, break, accelerate, stop)
5. Motion control - the final layer, which involve the actual control of the car

An example of a typical car can be seen in Figure 7.

Each of the mentioned components will be further analyzed in the next pages and on each layer we will describe the information and the algorithms that can be communicated between the autonomous agents (the cars).

4.2.1 Localization

The localization task consists of precisely detect both the car position and the orientation regarding the real space and mapping the position and orientation to a virtual model of the environment. This task is usually done mostly based on different sensors (GPS, inertial sensors) but also with the help of different algorithms like optical flow, ego motion or even neural networks [Han et al., 2018, Zhao et al., 2021]. This task is also called visual odometry. Generally, it can be considered that the current models work good enough if you have the necessary hardware, but we can think about a scenario where the current hardware specification of a car is not good enough and maybe an exchange of a visual odometry algorithm can increase the localization precision.

4.2.2 Object detection

The object detection is one of the most crucial tasks regarding autonomous driving, not only for the detection itself but also because almost all the other components will be based on the detected position of the cars and pedestrians, for example the trajectory prediction, the depth estimation and even the

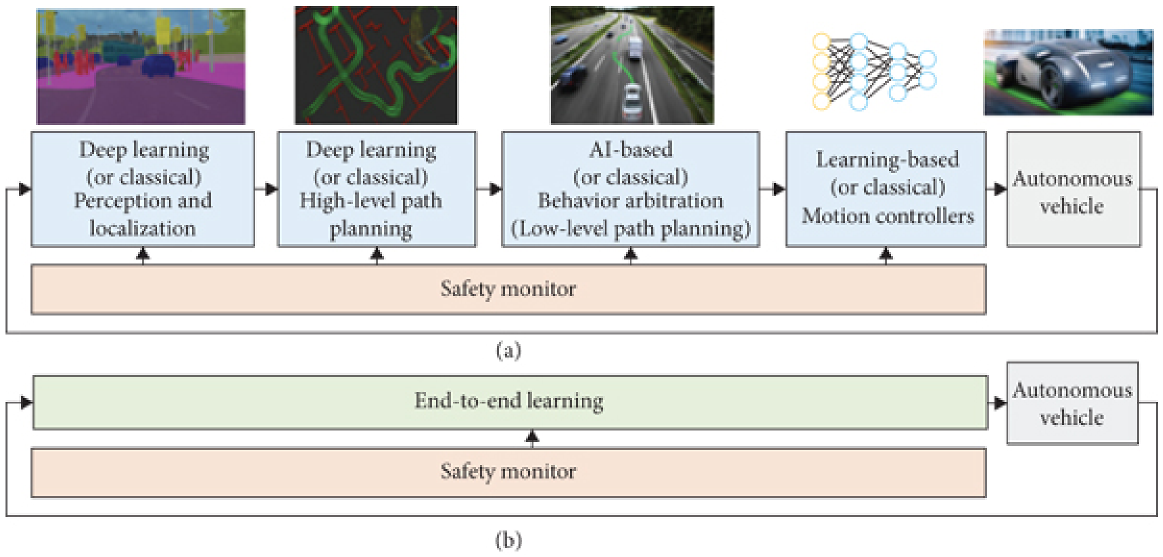

Figure 7: Typical autonomous car [Peng et al., 2020]

planning, where you have to know the position of a car if you want to overpass it. The object detection consists of the estimation of some bounding boxes for every object in the video. Typically, there are only a few classes that are relevant - car, people, bicycle, motorcycle, bus, truck, tram and also, very important, the traffic signs. The traffic sign detection is a crucial component, because of the big importance of detecting stop signs and traffic lights. The detection algorithm is typically made with pretrained neural networks and an exchange of information for a different environment can be safely done. Another scenario is where the classes for the detection should be increased.

4.2.3 Object tracking

The object tracking algorithm in an autonomous driving environment have an important role - to maintain the position of different object even when they are occluded and the objects cannot be detected, but can appear in the next frames. This task can be combined or even based on trajectory prediction, but an object tracking algorithm can be easier to make and will have faster inference times. An object tracking algorithm will label each object and maintain the label for the duration of the appearance of the specific object in the image. The object tracking can be done with or without neural networks. If neural networks are used, an exchange of information can be made for better results.

4.2.4 Lane detection

Another crucial task for autonomous driving is the lane detection. Even if many algorithms use classical AI approaches, more recent research uses convolutional neural networks for the lane detection task, sometimes combined with instance segmentation.Again, exchanging models for this task can be a good idea, especially for roads without good lane markings, as someone can find in many Romanian villages.

4.2.5 Semantic segmentation

The semantic segmentation task is especially important for the road detection. Even if the object detection boundary box is precisely enough, without the need of knowing the exact contour of an object, it lacks the capability of detection the drivable road, thus semantic segmentation is a crucial task. Some recent system try to do object detection, road segmentation and lane detection in the same task. [Wu et al., 2022]. Made especially with convolutional neural networks, exchanging information about segmentation is very important.

Because the relevance of this task, we will make our first task to examine in our ontology, because it is also relevant for the disaster response scenario. An example of labeling the road and the objects can be found in figure 8.

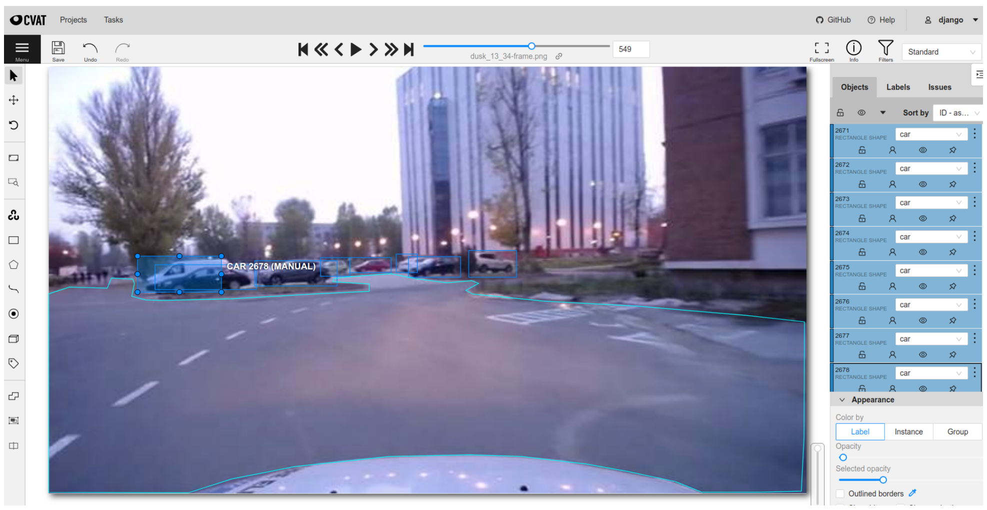

Figure 8: Road segmentation and car labeling

4.2.6 Depth estimation

The depth estimation task will try to estimate the distance to all the points in an image. The most important points are those with an object, so the most important task is to estimate the distance to the surrounding objects. This can be done via expensive sensors like the LiDAR and also with neural networks with pretrained models. If the car lacks the necessarily hardware, an exchange for depth estimation algorithms can be useful.

4.2.7 Trajectory prediction

The trajectory prediction task starts from the object detection and tries to estimate the future trajectories of the traffic agents (vehicles, people, bicycles). Typically is now done with neural networks and it is a crucial task necessary for braking (in case of a possible collision) and car passing. This component can also benefit of exchanging different models, if the data used for the current model consists only of images recorded during the day, for example.

4.2.8 Local and global planning

The planning component is typically done with classical algorithms, which don't involve neural networks. For example, the global planning can be done with a Dijkstra-like algorithm, or an A* algorithm. The local planning or the maneuver detection can be made only based on the prediction (vision) layer and otherwise use only classical AI algorithms (for example Monte Carlo Tree Search [Albrecht et al., 2021]) or can also use the results from vision and neural networks in some cases [Breuer et al., 2019]. However, the usage of especially designed neural networks is not so common on this layer, with limited application for our scenario.

4.2.9 Motion control

The final layer of an autonomous car consists of the actual movement of the car, by putting all the previous layers together and send a command to the car based on the local planning - move the steering angle, break or accelerate. However, for this layer there is only limited usage of deep learning algorithms. There are some end-to-end algorithms that try to estimate the steering angle and the acceleration only from the vision, but with worse results [Xiao et al., 2020] and we will not consider exchanging information for this layer with end-to-end algorithms.

4.3 Validation Design

Considering the previous detailed description of an autonomous car, our validation design will include, for the moment, only the perception and the prediction layers. We will focus especially on the tasks regarding the panoptic perception (object detection, road segmentation and lane detection), which are the most crucial tasks regarding autonomous driving. The depth estimation can be made with sensors and the object tracking can be made without deep learning. If the time will allow us, we will also integrate the task of trajectory prediction. The car discovery can be initially made considering the existence of a centralized server which will allow the agents to find information about other connected agents at a current time. Also, for validation purposes, we will not initially use a car simulator, only a predefined scenario which will include some active agents (cars) that are in different locations, have different algorithms for the previous tasks trained on different datasets and the environment will change according to the scenario (various changes regarding light, weather, road type, location and environment will be considered).

5 Progress on objectives

At the time of this report having completed the first 6 months of the project, we have achieved the following progress:

WP1: significant progress in creating the AI Folk ontology. The general model of the ontology has been created and a detailed description of the ontology has been done, also containing some elements for the autonomous driving scenario, but otherwise general to machine learning problems.
Milestone **M1.1** has been reached.

WP2: significant progress has been don in developing the AI Folk interaction protocol. A specification for the entire protocol has been created and an approipriate underlying implementation for communication between entities in an AI Folk deployment has been chosen.
Milestones **M2.1** and **M2.2** have been reached in terms of specification. The underlying implementation of the communication mechanism is implemented and what is left to do for both milestones is the integration of the AI Folk communication primitives.

WP3: explicit work on the AI Folk methodology has not been started yet, but numerous lessons have been learned while developing the autonomous driving scenario.

WP4: work is underway for developing the autonomous driving scenario, in terms of describing several models for use in such a scenario and implementing a scenario using the FLASH-MAS framework in which agents can search for such models in an AI Folk-type environment. Milestone 4.1 has not yet been reached, but progress towards it exists.

WP5: work has not started yet, due to delays in the initial contracting phase and reductions in funding.

WP6: a paper has been written and will be sent to a Q2 journal in December 2022. The paper reports on progress in the underlaying interction infrastructure that is being used in the AI Folk project and is titled "Multi-modal distributed interaction in multi-entity systems".

6 Executive report

As machine learning (ML) continues to permeate every aspect of daily life, gathering datasets and training state-of-the-art models becomes more and more the privilege of large technology companies. Recent approaches propose the decentralization of learning, such that model training happens on devices which are closer to the user, using that which resides on those devices. One of the core challenges of such an approach is heterogeneity of devices and openness of the distributed AI system.

Our research goal is to develop a knowledge model and an interaction protocol which allow a system formed of multiple actors – human, software, or organizational – to find, use and share improvements on machine learning resources. Such resources can be datasets, models, or experiences.

This project aims to develop the AI Folk framework and methodology, at the intersection of machine learning, knowledge management, and multi-agent systems. It comprises tools and methods that allow the management and discovery of ML-related resources in a distributed system.

In this first phase of the project, we have developed the conceptual architecture for the entities in the AI Folk ecosystem, defining the relation between input and output, the concept of situation recognition, and the elements related to model selection and searching for models in the ecosystem.

We have designed and developed a first version of the AI Folk ontology, which will be used for the description of ML-related resources (especially machine learning models), and which is currently biased towards problems and processes that occur in autonomous driving scenarios. The ontology has been developed together with a series of competency questions that help us identify the needs for the ontology, currently geared towards the autonomous driving scenario.

We have advanced on the aspect of entity interaction, learning from and using tools from the field of open multi-agent systems, identifying tools that are appropriate for cases of highly mobile entities, potential communication failure, open entity ecosystems (entities can join and leave the system at any time during execution). We have developed a series of communication primitives to serve as a basis for a first implementation of the AI Folk protocol.

We have made progress in the design of the software application that will constitute the proof-of-concept implementation of the autonomous driving scenario. We have performed an initial selection of the processes that we will include in the scenario and the types of models which the entities in the deployment will be able to search for and exchange.

Project coordinator

Andrei Olaru

References

[Albrecht et al., 2021] Albrecht, S. V., Brewitt, C., Wilhelm, J., Gyevnar, B., Eiras, F., Dobre, M., and Ramamoorthy, S. (2021). Interpretable goal-based prediction and planning for autonomous driving. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1043–1049. IEEE.

[Ayranci et al., 2022] Ayranci, P., Lai, P., Phan, N., Hu, H., Kolinowski, A., Newman, D., and Dou, D. (2022). Onml: an ontology-based approach for interpretable machine learning. *Journal of Combinatorial Optimization*, pages 1–24.

[Breuer et al., 2019] Breuer, A., Kirschner, J., Homoceanu, S., and Fingscheidt, T. (2019). Towards tactical maneuver detection for autonomous driving based on vision only. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 941–948. IEEE.

[Han et al., 2018] Han, S.-J., Kang, J., Jo, Y., Lee, D., and Choi, J. (2018). Robust ego-motion estimation and map matching technique for autonomous vehicle localization with high definition digital map. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 630–635.

[Klampanos et al., 2019] Klampanos, I. A., Davvetas, A., Koukourikos, A., and Karkaletsis, V. (2019). Annett-o: an ontology for describing artificial neural network evaluation, topology and training. *International Journal of Metadata, Semantics and Ontologies*, 13(3):179–190.

[Li et al., 2020] Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60.

[Nguyen and Weller, 2019] Nguyen, A. and Weller, T. (2019). Fairnets search-a prototype search service to find neural networks. In *SEMANTiCS Posters&Demos*.

[Olaru et al., 2019] Olaru, A., Sorici, A., and Florea, A. M. (2019). A flexible and lightweight agent deployment architecture. In *2019 22nd International Conference on Control Systems and Computer Science (CSCS), Bucharest, Romania, 28-30 May 2019*, pages 251–258. IEEE.

[Peng et al., 2020] Peng, M., Quek, T. Q., Mao, G., Ding, Z., and Wang, C. (2020). Artificial-intelligence-driven fog radio access networks: recent advances and future trends. *IEEE Wireless Communications*, 27(2):12–13.

[Wang et al., 2019] Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., and Chen, M. (2019). In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165.

[Wojciechowski, 2001] Wojciechowski, P. T. (2001). Algorithms for location-independent communication between mobile agents. In *Proc. of AISB'01 Symposium on Software Mobility and Adaptive Behaviour.*

[Wu et al., 2022] Wu, D., Liao, M.-W., Zhang, W.-T., Wang, X.-G., Bai, X., Cheng, W.-Q., and Liu, W.-Y. (2022). Yolop: You only look once for panoptic driving perception. *Machine Intelligence Research*, pages 1–13.

[Xiao et al., 2020] Xiao, Y., Codevilla, F., Gurram, A., Urfalioglu, O., and López, A. M. (2020). Multimodal end-to-end autonomous driving. *IEEE Transactions on Intelligent Transportation Systems.*

[Zhao et al., 2021] Zhao, B., Huang, Y., Wei, H., and Hu, X. (2021). Ego-motion estimation using recurrent convolutional neural networks through optical flow learning. *Electronics*, 10(3):222.