AI Folk: Sharing Machine Learning Models in a Multi-Agent Community

Andrei Olaru[0000−0002−2718−9195], Alexandru Sorici[0000−0002−6850−0912], Mihai Nan[0000−0002−3260−9705], and David-Traian Iancu[0009−0002−0846−6844]

National University of Science and Technology POLITEHNICA Bucharest,
313 Splaiul Independentei, Bucharest, Romania
{andrei.olaru,alexandru.sorici,mihai.nan,david_traian.iancu}@upb.ro

Abstract. With the rapid growth in the number of available pre-trained machine learning (ML) models for common tasks, with different performance, focus, and capabilities, complex problems can increasingly be solved through adequate choice of model, more than through training or tuning new models. In this paper we introduce the AI Folk methodology to address the challenge of autonomously managing ML models in a community of agents which can use and exchange semantic information about the models that they are using. We present a proof-of-concept implementation in an autonomous driving setting tackling various practical challenges which arise when dealing with this goal.

Keywords: machine learning, multi-agent systems, semantic description.

1 Introduction

Machine learning (ML) models are now used all around us, in various sizes, on various devices, handling tasks of various level of specialization. Libraries and stores of ML models are available online, such as Hugging Face[1] and the recently launched Qualcomm AI Hub[2], allowing the quick download of pre-trained models for a variety of tasks.

However, in complex scenarios, such as, for instance, autonomous driving, in which the activity of an agent lasts for a longer period of time, it may be impossible to say from the start what is the complete set of models that is going to be needed by an agent controlling an autonomous car, or what is the complete set of situations that the agent is going to have to address. Given the ability to use multiple models for the same task over its lifetime, an agent may benefit from information other agents have regarding those models.

The goal of this paper is to illustrate a practical means for agents in a multi-agent system (MAS) to use pre-trained machine learning models as a community, being able to search for and exchange experience information regarding ML models.

[1] Hugging Face https://huggingface.co

[2] Qualcomm AI Hub https://aihub.qualcomm.com/

This goal involves challenges related to formalizing information about the ML models, as well as practical issues regarding loading, using, and switching ML models during execution.

We place examples and a proof-of-concept scenario in the domain of autonomous driving,in cases where unexpected conditions arise, for which a specialized ML model is be more appropriate than what the agent currently uses:

> *An autonomous car travelling through a city enters a university campus, where the streets are shared between cars and pedestrians. The car encounters a large number of pedestrians walking in the streets and the current model for controlling the car stops the car abruptly, very often. Other autonomous cars in the campus use specialized models for navigating the campus. The agent contacts other agents in the local community and transfer a campus-specific model which it is able to use so that it will run slower, closer to pedestrians, and more smoothly.*

The issues in this scenario generalize to being able to describe ML models in order to be able to look for more appropriate models in a given situation; to identify models which match a given set of properties; and to integrate, in a running framework, new ML models and switch to them as appropriate. This is, in a nutshell, the goal of the AI Folk project[3].

We address the issue of dynamically selecting the model to use in situation when the distribution of the input data becomes very different from the training data distribution. This can happen in the case of lighter ML models or when the current input is very niche. Unlike federated learning, where a larger model is trained distributively, we target selecting the most appropriate pre-trained model.

Given our research goal, we have established the following objectives:

O1. Give agents the ability to use pre-trained machine learning models. Models are abstracted as black-box entities which have an input and an output;
O2. Give agents the ability to detect when the ML models currently in use become inappropriate for their current situation. This is the moment when they should attempt to identify a more appropriate model;
O3. Give agents the ability to create descriptions of their current situation and, conversely, to identify models that match a given description. This way, the agent can send and respond to queries regarding ML models;
O4. Give agents the ability to integrate new ML models in their model library and to switch between models as needed by the current situation;
O5. Create a methodology for achieving objectives O1-O4 in any scenario, identifying which aspects are scenario-invariant and which must be developed specifically for each scenario, and how that should be done.

We have instantiated the *AI Folk Methodology* – Objective O5 – on a proof-of-concept scenario similar to the one described above, implementing all the objectives for agents and separating generic and scenario-specific implementations, thus demonstrating how the methodology can be applied.

[3] The AI Folk project https://aifolk.upb.ro

After discussing related work, we present the AI Folk architectural elements and the methodology in Section 3, followed by practical challenges involved by the implementation in Section 4. The design of the experimentation protocol 5 is followed by the conclusions.

2 Related Work

The focus of AI Folk is on leveraging crowd-based reasoning to increase the efficacy and efficiency of individual agent decision making in the face of dynamic changes in the *characteristics* of the environment in which they operate. To do so, it is necessary for agents to be able to *describe*: (i) task-specific characteristics of the data they perceive, and (ii) characteristics of the *models* they use in prediction and decision making, including the data on which those models have been trained. When there is a mismatch between observed data characteristics and used model characteristics (a domain shift), an agent will seek to update its decision making model using *experiences* of other agents.

ML Model sharing is an increasingly common practice under the paradigm of Federated Learning (FL). However, the aspects and challenges that are most frequently addressed in the FL literature refer to federation architectures, optimization of the communication mechanisms or security and privacy [2]. In [1], for example, the focus is on using ideas from MAS coalition formation, to address challenges in FL based on non-IID data, where individual agents that train models are assumed to be self-interested. The authors of [6] use similar mechanisms of auctioning and coalition formation to enable communication-efficient FL under a collaborative Internet of Vehicles setup.

However, in such works the *learning task* is already known before hand and the focus is on improving ML models, which are already known to be compatible, based on distributed local learning and a periodic synchronization with a global ML model. In contrast, the focus of the AI Folk approach is on being able to *identify* the most relevant ML models among peer agents, based on a description centred on the *data and model characteristics* of the learning task. It therefore becomes important to look at structured methods that enable such descriptions.

Describing ML Models. In AI Folk, the ML models we focus on are based on deep neural networks (DNN). The vast architectural space of DNNs makes the selection of an appropriate architecture for a task challenging. Ontologies emerge as a promising solution to describe DNNs in a structured manner, enabling support for querying, analysis and comparisons of various DNN architectures, as well as their configuration, training methodologies and evaluation procedures.

FAIRnets [7], for example, is an RDF dataset that stores information about existing neural networks. Using RDF and OWL, the system enables SPARQL querying for desired characteristics of DNNs. Additionally, detailed information and further links about network architectures are provided.

In AI Folk we choose to extend the ANNETT-O ontology [3], which offers a structured and computationally accessible vocabulary, enabling systematic documentation and analysis of DNN design and experimentation, with emphasis on aspects of evaluation, topology, and training.

Semantic Data Description. The other need in AI Folk is to use knowledge graphs (KG) to *semantically describe the data* on which a model is trained and operates. KG descriptions allow inclusion of existing *domain knowledge* into the desired functionality of data-driven prediction models. Consequently, ML models get better at predicting under setups of high data variability (e.g. HAR in videos on the Kinetics dataset [5]), and knowledge-based *explainability* can be achieved in multiple domains (e.g. image recognition, recommender systems, natural language applications) [11]. Closest to the AI Folk interest is the use of KG to index both general and task-specific *characteristics* of the data on which a model operates. OntoDM is a suite of ontologies which is extended to provide dataset-, task- and algorithm-level semantic descriptions for applications in Data Mining [4], with use cases ranging from neurodegenerative disease to earth observation datasets, to autonomous driving and scene explanation. The work highlights the nuScenes-QA dataset [9] as an enabler for improved driving scene understanding, since DNN models can setup learning tasks that answer to queries regarding the *type*, *number* and *status* of encountered traffic participants. Such works show that there is a need and a benefit in terms of understanding, performance and explainability in semantically describing data. In AI Folk we extended ANNET-O in a modular fashion, creating a *core*[4] and a *domain specific* vocabulary that facilitates describing learning tasks and their data characteristics. The resulting ontology allows agents to identify *changes* in data characteristics which are relevant for a specific learning task.

3 Architecture and methodology

The main goal of the project is to create a generic architecture for a variety of scenarios in which a number of agents have to make decisions using machine learning models and agents have the possibility to exchange information about the models they are using. Differently from a distributed system in which entities collaborate on the same task, in a multi-agent system agents may have different goals, but cooperation my lead to better results for the agents involved.

During this research, we have addressed two application scenarios – autonomous driving and disaster response. In both of them, the global outcome can be improved when agents are able to switch the ML models that they are using. While this process could be useful for a greater variety of scenarios, we have defined several invariant elements and the AI Folk methodology specifies how they can be applied and adjusted for other types of scenarios.

Achieving the objectives stated in the Introduction requires several elements:

[4] The core ontology can be found on the project website.

- an agent deployment framework allowing communication between agents (O1 and O4);
- a means to work with ontologies (written in OWL-DL, for instance) (O2 and O3);
- a means to dynamically load and run machine learning models (O1 and O4);

In addition, development is substantially aided by means to validate designed functionality. Therefore, we have developed a method to run scenarios, that is, to provide agents with specific inputs at specific times and to get the output for verification.

For our proof-of-concept implementation, we have relied on a multi-agent system deployed using the Flash-mas framework[5], a Java-based modular platform built around the generic concept of *entities*, which can be agents, but also other persistent first-class entities in a MAS, such as nodes and communication infrastructures [8]. Although not mandatory, agents can be implemented as collections of *agent shards*, each shard implementing a specific functionality, with shards of an agent interacting by means of an event queue. We have chosen Flash-mas for its significant modularity and flexibility, not least because of the ability to implement non-agent components as first-class entities of the framework (as opposed to Jade, for instance).

Machine learning models are generally implemented in Python, so we have integrated in Flash-mas an interface to a local RESTful web service which can load and run ML models (see more in Section 4).

3.1 General Methodology Principles

The AI Folk approach applies to agent systems in which participants have to execute their predictive tasks in environments in which the degree of input variability is too large to be covered by a single prediction model. Such situations require agents to form *experience sharing communities*, in which agents collaborate to exchange appropriate prediction models with each other. The mentioned operating conditions necessitate the support of a framework that is based on the following design principles and requirements.

1. Formation of experience sharing communities: refers to giving support to agents to dynamically identify and communicate with the set of peers that have a *relevant experience*, in terms of similarity of task and data descriptions on which a prediction model has been successfully employed (where success is measured based upon standard or mutually agreed upon task-specific metrics). Thus, the agent community building principle is driven foremost by an information retrieval concern, as opposed to one related to logical / physical vicinity or speed of communication.

2. Identification of salient prediction model, task and data characteristics: refers to enabling to agents to describe *what* is changing in the task for which they have to use prediction models. AI Folk enlists the use of semantic web technologies, such as ontologies, to describe task types, general and task-specific data

[5] Fast and Lightweight Agent Shell https://github.com/andreiolaru-ro/FLASH-MAS

attributes, DNN model architectures, DNN model evaluation datasets and evaluation performance. Note that we currently focus on the case where data characteristics that agents should pay attention to (because they can cause distribution-shifts) are defined based on human domain knowledge. The methodology can be extended in future work to include approaches where a data-driven model itself is responsible to highlight and semantically map the data characteristics which it considers are relevant in the prediction and which have changed.

3. Transfer, loading and running of prediction models: refers to giving support to agents to reference shared models, transfer model parameters, dynamically configure a model deployment environment and run the inferences required by their prediction task. AI Folk envisions a system that supports both (i) cloud-based setups, where agents run predictions in an already configured cloud environment for which a web-based entry point and access credentials are provided when sharing it with other agents, and (ii) an edge-based setup that enables local running of prediction models and which requires methods to efficiently serialize and transfer model parameters and deployment configurations (in a manner similar to Federated Learning setups). Note that in AI Folk we currently focus on building support for cloud-based setups.

3.2 Scenario specific design

Our reference scenario presents the case of autonomous driving, where agents controlling the vehicles can encounter frequently changing *driving scenes* in an extended urban setting (involving both main street, as well as residential area locations). The specific prediction task we focus on is that of semantic segmentation of the drivable path based on video input. The community of experience sharing agents is that of the extended urban environment and agents use the cloud-based setup to run and exchange segmentation models.

The most important concern to address from the principles outlined in Section 3.1 is the definition of the semantic segmentation task specific vocabulary to describe the conditions under which the task is being performed. Informed by domain expertise, we adopt a competency question based procedure [10] to extend the general AI Folk ontology vocabulary with one that describes the specifics of data involved in a semantic segmentation task for the autonomous driving domain. By answering questions meant to qualify which aspects in a driving scene should be accounted for when analyzing the performance of a semantic segmentation model, the resulting vocabulary addresses descriptions of: (i) the type of driving scene (e.g. parking lot, residential area, main city street, rural area), (ii) the weather and illumination conditions (e.g. in daylight, at dusk or at night, in sunny, overcast or rainy conditions), (iii) properties giving statistics of the type of roads (e.g. with cross or T-shaped intersections) or the min/avg/max counts of different types of traffic participants (e.g. pedestrians, other vehicles). The extension of the core AI Folk ontology[6] to the needs of the reference scenario is used by agents to both describe their current task, as well

[6] The ontology for the auto driving scenario can be found on the project website.

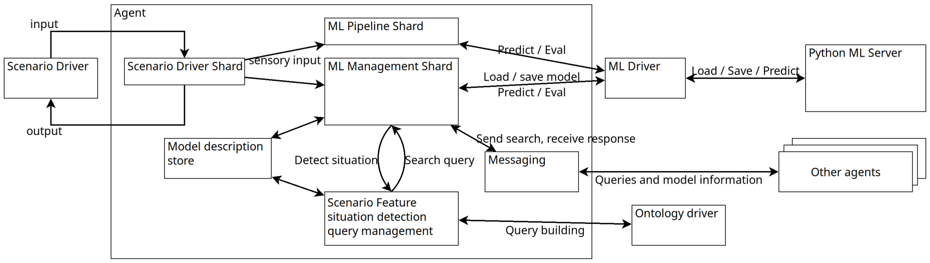

Fig. 1. The AI Folk architecture, showing the interaction between components inside and outside the agent.

as the datasets on which semantic segmentation models have been trained and evaluated on. The common description vocabulary is what enables search and fit of existing segmentation models to the current data characteristics of the task. The showcasing of agent communication, task description, model search and model use, as enabled by the AI Folk approach for the reference scenario, is given in Section 5.

4 Practical Challenges

Implementing the AI Folk methodology raised several relevant challenges. In terms of agentification, the challenge was to separate scenario-independent and scenario-specific components, and to model non-agent and sub-agent entities supporting the implementation of the methodology.

For support functionality, the open and modular nature of FLASH-MAS allowed us to create a new type of entity in – the *Driver*, which is local to a node (a machine) and is available to all agents on that node. We have implemented 3 drivers: the `ML Driver` interfaces with the Python server to relay operations related to ML models; the `Ontology Driver` interfaces with an OWL-DL ontology that integrates the description of models and driving situations; and the `Scenario Driver` is used in validation to feed input to agents and to verify their output, according to a scenario file.

Scenario-specific functionality – such as the logic for recognizing situations and choosing appropriate model – has been isolated in instances of a *Feature* sub-agent entity which is used by each agent in a scenario.

We have encapsulated AI Folk functionality inside the agent in three types of shards; the `ML Management Shard` keeps a list of the models currently usable by the agent, as well as the models currently in the pipeline, and it evaluates, via the scenario specific *feature*, whether there is a need to switch to a different model; the `ML Pipeline Shard` picks the input from the event queue and sends it to the ML model which is currently in use, posting the output back into the agent's event queue; the `Scenario Driver Shard` supports validation by feeding input data from the `Scenario Driver` to the agent's event queue, and by picking the corresponding output to send back to the driver for validation.

A challenge in deploying the AI Folk scenarios was the FLASH-MAS framework is implemented in Java, but most machine learning models are implemented in Python, and using dedicated ML libraries such as PyTorchand Tensorflow. We have solved this challenge by interfacing the `ML Driver` to a local Flash web server started automatically by the driver, offering a RESTful API supporting several operations: *listing* configured models, *adding* a new model, adding information about a *new dataset*, *exporting* a model, and using a model to *predict* an output based on the input.

ML models are not standardized. Each model comes with is own set of prerequisites, instantiation procedure and configuration parameters, input format and processing, and output processing. Therefor, to each model we have attached (1) a YAML file which contains configuration options, and (2) a Python file containing the methods to process the input and the output of the model according to its specific requirements. These transformations can be reused for multiple models, if possible. Hence, a model is fully defined by the model file (e.g. a `.pth` file), a YAML configuration, and a Python file with transformation code.

5 Design of AI Folk Evaluation Protocol

To apply the methodology presented in Section 3.1 we design an experimentation protocol, describing the application conditions, the experiment steps and the evaluation method.

The AI Folk approach is highly relevant in situations where in the environment of an agent a *data distribution shift* is expected to occur sufficiently often. Our reference scenario of semantic segmentation in autonomous driving, implying a large mobility of the self-driving agent and the possibility to experience significant changes of driving scenes, is a good example of this application condition. To simulate such conditions from existing datasets, the following steps are followed.

First, a domain of application (e.g. autonomous driving) and a main task (e.g. drivable road segmentation) are specified. For the main task, attributes which can influence it are identified (e.g. min/average/max number of pedestrians encountered over a recent time window). This leads to an instance of the core AI Folk ontology expressing the data characteristics which are deemed to influence an ML algorithm by existing human domain knowledge (see also Section 3.2).

Second, for the identified data attributes, a set of domain-knowledge informed *change conditions* are defined, whose objective it is to determine whether there is a *mismatch* between the characteristics of the data on which the currently employed ML model (which is solving the main task) was trained and the characteristics of the data from the currently experienced scenario (e.g. the minimum number of pedestrians detected over frames collected for the past minute is higher than the maximum number of pedestrians detected on any frame of the training set for the currently used segmentation model).

A detected mismatch signals the need to search for a replacement model, obtained from the *experience* of other agents in the community (see Section

3.1). Step three involves setting filtering criteria for searched models. New models need to have been trained on data that has *a suitable similarity* with respect to the characteristics of the attribute that caused the original mismatch. The filter is set up as a domain-knowledge informed reasoning rule that looks at the suitability of the data characteristics (e.g. the difference in average number of pedestrians is below a threshold), as well as the *performance* of the model on the main task (e.g. the pixel-wise accuracy for the segmentation of the drivable road class is above 95%).

The next step involves simulating the condition mismatch using one or more existing dataset(s) (e.g. Cityscapes or KITTI-360 for our segmentation task). A ML model (e.g. a DeepLab-v3 model) is finetuned on the main task on splits of data from a first selected dataset, which correspond to mismatching conditions on the identified attribute (e.g. number of pedestrians). A second ML model (e.g. YOLOv8) is finetuned on the same main task on a second dataset, on a split of data respecting the characteristics of the identified data attribute. Variation can be created in this case, by creating dataset splits that correspond to different characteristics of the selected data attribute (e.g. splits of data corresponding to different average pedestrian detection per 100 frames). Each finetuned model is given to an agent in the modeled *experience sharing* communities. Each agent has an evaluation holdout dataset.

To evaluate the experiment, agents run the mismatch identification rules and the corresponding search for a replacement model from the *experience* of other agents. The agent receives the most appropriate replacement model, based on the infrastructure and interactions described in Section 4. The current model and the replacement model are evaluated on the evaluation holdout dataset and if the percentage difference in the evaluation metric for the performance on the main task (e.g. pixel-wise accuracy / Dice-score for drivable road segmentation) shows, for all agents, a net increase in performance on the main task, then this constitutes an indication that reasoning on *domain knowledge informed mismatch conditions for key data characteristics* is a key proxy for anticipating performance improvement if a ML model were to be retrained on the current distribution shift. The main advantage of the AI Folk methodology is that, under the assumption of high data distribution shifts for high mobility of agents in their environment, reasoning over similar experiences in terms of domain-knowledge informed data characteristics is faster and more convenient than retraining or updating of the local ML model (as is typical in a federated learning setup).

6 Conclusion and Future Work

We have approached the issue of managing pre-trained ML models in a multi-agent community via semantic descriptions of the model and the ability to dynamically, seamlessly switch between models at runtime.

As future work, the AI Folk methodology will be applied to more complex scenarios and more effort will go into evaluating the performance of models such that it can be factored into the decision process.

Acknowledgments. This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS - UEFISCDI, project number PN-III-P1-1.1-TE-2021-1422, within PNCDI III.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Arisdakessian, S., Wahab, O.A., Mourad, A., Otrok, H.: Coalitional federated learning: Improving communication and training on non-iid data with selfish clients. IEEE Transactions on Services Computing (2023)
2. Beltrán, E.T.M., Pérez, M.Q., Sánchez, P.M.S., Bernal, S.L., Bovet, G., Pérez, M.G., Pérez, G.M., Celdrán, A.H.: Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. IEEE Comm Surveys & Tutorials (2023)
3. Klampanos, I.A., Davvetas, A., Koukourikos, A., Karkaletsis, V.: ANNETT-O: an ontology for describing artificial neural network evaluation, topology and training. Intl Journ of Metadata, Semantics and Ontologies **13**(3), 179–190 (2019)
4. Kostovska, A., Džeroski, S., Panov, P.: Semantic description of data mining datasets: An ontology-based annotation schema. In: International Conference on Discovery Science. pp. 140–155. Springer (2020)
5. Ma, Y., Wang, Y., Wu, Y., Lyu, Z., Chen, S., Li, X., Qiao, Y.: Visual knowledge graph for human action reasoning in videos. In: Proceedings of the 30th ACM International Conference on Multimedia. pp. 4132–4141 (2022)
6. Ng, J.S., Lim, W.Y.B., Dai, H.N., Xiong, Z., Huang, J., Niyato, D., Hua, X.S., Leung, C., Miao, C.: Joint auction-coalition formation framework for communication-efficient federated learning in uav-enabled internet of vehicles. IEEE Transactions on Intelligent Transportation Systems **22**(4), 2326–2344 (2020)
7. Nguyen, A., Weller, T.: Fairnets search-a prototype search service to find neural networks. In: SEMANTiCS (Posters & Demos) (2019)
8. Olaru, A., Sorici, A., Florea, A.M.: A flexible and lightweight agent deployment architecture. In: 22nd Intl Conf on Control Systems and Computer Science (CSCS), Bucharest, Romania, 28-30 May 2019. pp. 251–258. IEEE (2019)
9. Qian, T., Chen, J., Zhuo, L., Jiao, Y., Jiang, Y.G.: nuScenes-QA: A multi-modal visual question answering benchmark for autonomous driving scenario. In: AAAI Conference on Artificial Intelligence. vol. 38, pp. 4542–4550 (2024)
10. Ren, Y., Parvizi, A., Mellish, C., Pan, J.Z., Van Deemter, K., Stevens, R.: Towards competency question-driven ontology authoring. In: The Semantic Web: Trends and Challenges: 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings 11. pp. 752–767. Springer (2014)
11. Tiddi, I., Schlobach, S.: Knowledge graphs as tools for explainable machine learning: A survey. Artificial Intelligence **302**, 103627 (2022)