Human-in-the-loop online multi-agent approach to increase trustworthiness in ML models through trust scores and data augmentation

Gusseppe Bravo-Rocca*†, Peini Liu*†, Jordi Guitart*†,
Ajay Dholakia‡, David Ellison‡, Miroslav Hodak ‡
*Barcelona Supercomputing Center, Barcelona, Spain
†Universitat Politècnica de Catalunya, Barcelona, Spain
‡Lenovo Infrastructure Solutions Group, Lenovo, Morrisville, NC, USA
E-mail: {gusseppe.bravo,peini.liu,jordi.guitart}@bsc.es,{adholakia,dellison,mhodak}@lenovo.com

***Abstract*—Increasing a ML model accuracy is not enough, we must also increase its trustworthiness. This is an important step for building resilient AI systems for safety-critical applications such as automotive, finance, and healthcare. For that purpose, we propose a multi-agent system that combines both machine and human agents. In this system, a checker agent calculates a trust score of each instance (which penalizes overconfidence in predictions) using an agreement-based method and ranks it; then an improver agent filters the anomalous instances based on a human rule-based procedure (which is considered safe), gets the human labels, applies geometric data augmentation, and retrains with the augmented data using transfer learning. We evaluate the system on corrupted versions of the MNIST and FashionMNIST datasets. We get an improvement in accuracy and trust score with just few additional labels compared to a baseline approach.**

***Index Terms*—Trust, multi-agents, robust AI, data centric**

I. Introduction

In Machine Learning (ML), the main focus has been given to improving the accuracy of ML models, driven by benchmarks that aim to accelerate the development of new models. However, this is not enough to have reliable models ready to be used in real complex situations. In particular, it is not clear whether one should really trust the predictions of a model in applications where every decision carries some responsibility [1]. For example, a model that assists a doctor in a medical diagnosis based on scanned images can reduce diagnosis times to serve more patients and also reduce human mistakes. However, what if the model provides a wrong diagnosis? Worse yet, what if the wrong prediction delivered has a high percentage of confidence? Clearly, the model trustworthiness is a critical point to be analyzed and taken into account.

To improve trust in ML models, many methods opt to first improve the accuracy of the model, by modifying internally its structure, and consequently improve the trust. While it is true that an improvement in accuracy helps to improve trust, it does not guarantee it (i.e., unbalanced datasets, fraud detection) [2]. A better way is to interact with the model externally in such a way that, through continuous monitoring and evaluation, small gains in trust can be achieved with less human effort.

In this paper, we propose a multi-agent system that combines both machine and human agents to improve ML models trustworthiness. As shown in Fig. 1, a set of agents works together to quantify the trust on the set of predicted samples, so that, given new sets of corrupted data to be inferred, they can detect untrustworthy samples and improve the model (and the training data) over time. The Supervisor agent is in charge of obtaining the initial values of the system, e.g., the current model, the training set, etc. With these values, the Checker agent trains a model that calculates a trust score for each individual instance, and applies a rule to identify anomalous instances. Those are sent to the Improver agent, which is assisted by a human agent to label them, applies data augmentation, and retrains the model using transfer learning from the weights of the previous model and the new training set (adding augmented labeled samples). This process is performed iteratively during the inference stage, so that the model can face situations similar to those ocurring in real life.

The remainder of this paper is as follows: Section II states the background and related work. Section III describes the proposed multi-agent system. In Section IV, we discuss the evaluation results on two datasets. Finally, the conclusions and new ideas for future work are presented in Section V.

II. Background and Related work

A. The role of humans in learning systems

The role of end-users in learning systems is often relegated to their usage, excluding the participation in their continuous improvement. A more interesting alternative arises in interactive ML [3], which allows users with little expertise to explore and adapt the behavior of the model interactively, drastically reducing the intervention of experts (e.g., data scientists). Examples of such users' interaction are given in image segmentation [4], where the user corrects the segmentation made by the system by painting pixels as foreground or background in an image. Our work takes this idea and extrapolates the model-human interaction to an agent-human one.

 DOI 10.1109/COMPSAC54236.2022.00014

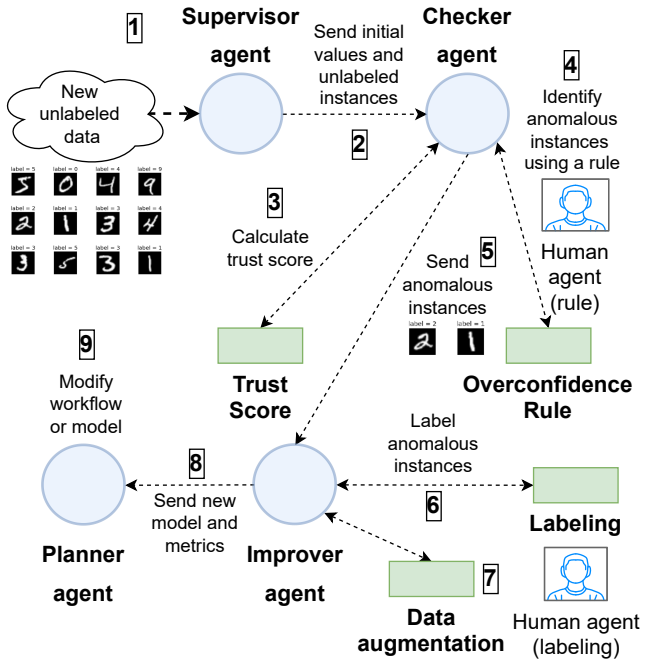

Fig. 1. Overview of the proposed multi-agent system.

B. Thinking fast and slow for machine and human agents

Kahneman's research on reasoning and decision-making [5], describes two ways in which the brain forms thoughts, namely *System 1* and *System 2*. These can be applied to AI as well. *System 1* mimics how ML works and *System 2* mimics symbolic AI. That is, *System 1* is data-driven and *System 2* is knowledge-driven. *System 1* is good at sensing and reading, i.e., perceptual tasks. Conversely, *System 2* is grounded on planning, logic, search, and explicit knowledge [6].

In our work, we use these ideas to develop a hybrid approach to improve a model over time using learning (fast) agents and symbolic (slow) agents. The latter are meant to be more logical, which allows humans to intervene in workflow improvement (enabling a human-in-the-loop approach). The Neuro-Symbolic Concept Learner [7] uses this hybrid approach to learn images and words by reading paired questions and answers that are translated into symbolic programs.

C. Agents in production systems

The agent technology has been used for a while for a myriad of applications such as robotics [8] and automated online assistants [9]. An agent is an entity that lives in an environment and has its own goals, knowledge, and actions [10]. A multi-agent system models agents' goals and actions directly via communication or by using the environment. In our work, we use the structure of a multi-agent system to model a direct communication between different agents.

Khalil et al. [11] have explored the integration of ML and multi-agent systems through information sharing and Q-Learning. However, to maximize the reward, a large space of scenarios must be explored, requiring a lot of data and computation. Our approach is to have pre-established behaviors to reduce the overload of states. To implement such a structure, we take advantage of microservices in contrast to the bunch of tools and languages that were used for such a goal [12]. Microservices features allow building systems at the scale that is needed for creating a multi-agent system, thanks to their commonalities with the characteristics of agents [13].

D. Measuring trust in ML models

To evaluate the overall trustworthiness of a ML model, similarly as done with the accuracy, we need a scalar value to compare with. For that purpose, on our evaluation, we use the *NetTrustScore* [14]. That calculation needs some prerequisites, namely, *question-answer trust*, *trust density*, and *trust spectrum*. Equation (1) quantifies the *question-answer trust*, e.g., the trust when predicting whether a picture shows a cat, by calculating the trustworthiness of an individual instance x (the question) for one label y (answer) based on its behavior under correct and incorrect labels z (oracle answers), that is, where y matches z ($R_{y=z|M}$) or where does not ($R_{y\neq z|M}$) over the space of all possible questions x given a model M. Equation (2) measures the *trust density*, e.g., the distribution of trust when predicting many pictures of a cat, by calculating the density distribution of question-answer trust over a set of instances (e.g., training set), in other words, the trust behavior of a model M for a specific answer scenario y. Finally, Equation (3) measures the *trust spectrum*, e.g., the average overall trust for class 'cat', by calculating the average question-answer trust across all the N instances for a given class z.

$$Q_z(x,y) = \begin{cases} C(y \mid x)^{\alpha} & \text{if } x \in R_{y=z|M} \\ (1 - C(y \mid x))^{\beta} & \text{if } x \in R_{y\neq z|M} \end{cases} \quad (1)$$

$$F(Q_z) = \{q \in Q_z(x,y), \forall x \in X, z : answer\} \quad (2)$$

$$T_M(z) = \frac{1}{N}\sum_{x=1}^{N} Q_z(x) \quad (3)$$

Given those, the *NetTrustScore*, which measures the trust spectrum over all the L classes (e.g., the overall trust for classes 'cat', 'dog', ...), is formalized as follows:

$$NetTrustScore = \frac{1}{L}\sum_{z=1}^{L} T_M(z) \quad (4)$$

Other related works explore how to obtain precise probability estimates for classification problems using calibrated binary probability estimates [15]. However, this method is used mainly for classical ML models. Other works have targeted modern deep networks, like ensembles of neural networks [16], and proposed to use an Average Early Stopping (AES) algorithm. However, these approaches are computationally expensive and yield lower interpretation.

III. Multi-agent system for trustworthy models

This section describes our approach to increase trustworthiness in ML models as a multi-agent microservice system that combines learning and symbolic agents.

A. Agent properties

In our current design, each agent is reactive with some limited internal state that is saved in a tracking repository. This means that the agents lack the deliberation to devise a long-term plan by themselves; instead, they keep an ongoing interaction with the environment (new input data and other agents) to act upon any change in time and respond timely. Additionally, each agent is rational because it is programmed to always perform the correct actions, the most appropriate to reach its objective, as a response to the information perceived by its sensors (new data and messages from other agents).

Some agents have a learning behavior, for instance, the Checker, whose mechanism of detecting anomalies (whose trust score is considered anomalous) requires learning historical information about training data. Other agents, such as the Improver, have a symbolic behavior in the form of heuristics (rules that allow establishing the thresholds of the model trust) and in the use of human labeling to pick the instances with high uncertainty (anomalous cases of overconfidence).

Moreover, the agents have a temporal continuity, that is, the agents are available 24/7 to answer requests. However, the most important property is the agents' social ability to interact with others via some kind of agent-communication protocol that fulfills the following properties. First, the agents are cooperative and respect the benevolence assumption. All the agents share a common goal and there is no conflict between them. This implies that all that matters is the overall goal and not the individual one. For instance, in our system, the global goal is to improve the model trust. Second, the agents have explicit communication and have the intention to exchange messages. In our system, they only react to the performative messages, such as QUERY, INFORM, PROPOSE, etc. Finally, the agent's interaction is driven by certain meta-negotiation. Our system implements some limited features from the Contract Net Protocol (CNP) [17]. Given that we have some assumptions about each agent's beliefs (e.g., the Supervisor agent knows which agents should do the checking), we can perform direct contracts. As a result, our system sends simple messages, reducing response times.

B. Agent definition

We assume that the training data increases or improves in quality, sequentially and in discrete time, as the model receives requests. Additionally, the agent's actions, i.e., its conclusions, influence the future performance of the model. Given those premises, let s_t^i denote the state of agent $i \in \mathcal{A}$ (set of agents) at time t, a_t^i denote the action taken by agent i at time t, F_t^i denote the external flow of information (represented by messages e that refer to requests τ) sent by other agents that affects the behavior of agent i at time t (in particular, its decision about a_t^i) ($F_t^i = (e_t^{m,i}(\tau^m)), \forall m \in \mathcal{A} \setminus \{i\}$), s_{t+1}^i denote the next state at time $t+1$ and r_t^i the reward (in our case, the net-trust score) at time t, both obtained when agent i applies a_t^i using the information provided by s_t^i and F_t^i at each time t ($((s_t^i, a_t^i, F_t^i) \implies s_{t+1}^i)$, $((s_t^i, a_t^i, F_t^i) \implies r_t^i)$). Then, the goal is to find a sequence of actions $\{a_t^i\}$ to maximize the cumulative reward $\sum_{t=0}^{T} r_t^i$ given an initial state s_0^i. For instance, a_t^i can be updating the model, training a Checker to detect distribution drift, or asking for human labels.

C. Architecture of the multi-agent system

The internal behavior of the agents in this paper builds on a previous work [18] that considers model debugging through static nodes in a graph. In a nutshell, we propose to convert those static nodes into dynamic agents (the Checker and Improver) and add two agents with new responsibilities (the Supervisor and Planner). A detailed description of each of these agents is provided in the following sections.

1) Supervisor (Initializing and starting the MAS): This agent is in charge of getting the initial metadata, such as latest model, training data, etc., which are needed to proceed further in the debugging process. It starts the multi-agent system by sensing a stimulus (for instance, scheduling time or amount of inference samples) and exposes the metadata to the Checker.

2) Checker (Untrustworthy points and rule-based threshold): As shown in Fig. 2, the Checker agent performs three steps, namely reducing the dimensionality of the input data using an autoencoder (the trust score estimator works best for low to medium feature space size), modeling a trust estimator to measure the agreement in the predictions (in the form of a trust score) between the model and a modified version of a nearest-neighbor (k-NN) classifier [19], and applying a rule-based procedure to assess what ranges of trust score values and prediction probabilities are considered not trustworthy.

As the labels are unknown in the inference stage, the Checker agent models a complementary unsupervised trust score estimator. Let be h the model, $(X, Y) = (x_1, y_1), ..., (x_n, y_n)$ the training data, x' the inference sample, and K a k-NN estimator trained on (X, Y) that estimates a density set (k-nearest neighbors of a class without outliers), the trust score of x' is defined as shown in (5), that is, it is the ratio between the distance from the inference sample to the density set of the nearest class different from the predicted class ($\hat{K}$), and the distance from the inference sample to the density set of the class predicted by h.

$$T(h, x') = d(x', \hat{K}(h(x'))) / d(x', K(h(x'))) \tag{5}$$

This trust score and the confidence of the predictions are then used to rank them according to a rule-based procedure (provided by a human agent) that tries to identify anomalous predictions that are considered not safe. If the model yields a prediction with high confidence (e.g., probability between 0.65 and 0.95) but with a low trust score (e.g., lower than 1), then this behavior is considered as overconfident, and the model should not be trusted despite its high confidence.

3) Improver (Human labeling and data augmentation): A form of human intervention that has worked very well in supervised learning problems is labeling. This technique allows business experts or product owners to label data to create datasets ready for the creation of ML models. Similarly, data

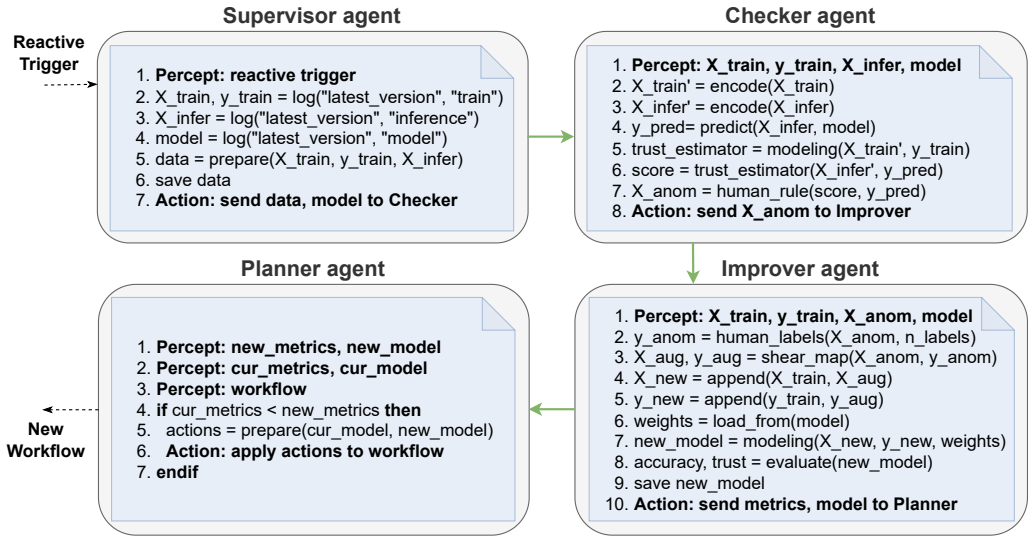

Fig. 2. High level multi-agent execution. Figure shows the interaction between Supervisor, Checker, Improver, and Planner agents to enhance the model trust.

augmentation came up as a way to artificially augment data in cases where datasets are small. ML models are susceptible to the quality of labeling and the number of samples. Therefore, applying the correct, unambiguous, and consistent labeling [20] is as important as augmenting data that follows the same training distribution, without adding noise [21].

As shown in Fig. 2, the Improver agent obtains the anomalous instances from the Checker, which are only a subset of the many instances that arrive in the inference stage. A human agent intervenes and labels a given amount of these instances (n_labels). Then, the Improver applies geometric data augmentation (shear mapping), which is a non-intrusive technique that maintains the same semantics of the training distribution. The shear mapping is a linear transformation of $\mathbb{R}^n$ that distorts the shape but keeps the same n-dimensional measure (hypervolume) of any figure. We applied this transformation to augment new instances that were previously labeled by a human, allowing the model to have more variety of examples to be trained, always keeping the same semantics.

At the end, the Improver applies a simple transfer learning from the current model's weights (new models fit using past weights) to retrain a new model after appending the new augmented data to the training set, thus emerging a more robust model in each human intervention. Transfer learning improves a target model on a target domain by transferring knowledge from a different source domain in order to reduce the dependence on large amounts of target domain data [22].

4) Planner (Modify the current model in production): This agent is in charge of getting information from the Improver agent so that it can apply the required actions to the model (or workflow) if some conditions are met, for instance, if the model trustworthiness is greater than the previous one.

IV. Experiments and Results

This section evaluates the effectiveness of using both human and machine agents to improve the model's trustworthiness by comparing two scenarios. Scenario 1 ('Agents') corresponds to our proposal to retrain the model using information coming from the Supervisor, Checker, and Improver agents (the Planner agent is meant for production systems) as described in Fig. 1. Scenario 2 ('Random'), which is fairly comparable to the first one, represents a common form of retraining a model. As shown in Fig. 3, it only performs labeling of random samples. That is, performs a random choice of the new instances, asks the human to label them, and retrains with (X_train + X_random, y_train + y_labels_random).

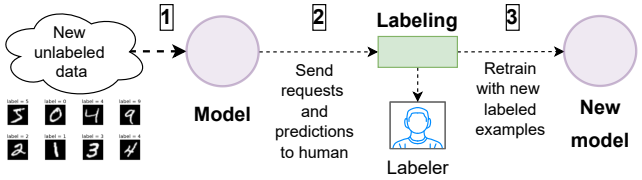

Fig. 3. Pipeline for Scenario 2 (Random approach).

The methodology to compare these scenarios is as follows. First, we select a testing set from the original dataset and an inference set with noise for both scenarios. The model is evaluated on these two sets in order to make accuracy and NetTrustScore comparable. We also set the number of instances to be labeled (only 15 instances, even though the number of anomalous instances can be larger) and the seed number to have a fair comparison. Second, for each iteration (time step), we issue a batch of new requests from the inference set to feed the model and evaluate how scenarios 1 and 2 perform on these new data. We repeat for a number of iterations while tracking the model improvement.

The testbed used in the experiments is as follows. Platform: Fedora Linux 35 (64 bits). Hardware: AMD Ryzen 9 5900HS, NVIDIA RTX 3060, 40 GB RAM. Software: Anaconda3 Python 3.8, Tensorflow 2.5.0, and Pytorch 1.9.0. Datasets:

MNIST [23] (60,000 28×28 pixel grayscale images of handwritten digits from 0 to 9), MNIST-C [24] (corrupted MNIST with 4 corruptions: 'contrast', 'impulse', 'shot', and 'gaussian'), FashionMNIST [25] (60,000 28x28 pixel grayscale images of 10 types of clothing), and FashionMNIST-C [24] (corrupted FashionMNIST with the same 4 corruptions).

As for the results on the MNIST dataset, Fig. 4 shows the accuracy on the two scenarios over 20 iterations. We can see that the model accuracy improves on both scenarios over time, being the 'Agents' approach the one that performed better. Similarly, as shown in Fig. 5, which compares the trust on both scenarios, the 'Agents' approach outperforms the 'Random' method, basically because the set of instances used to retrain the model (anomalous instances) were more significant in this scenario, hence, the model generalized better. Moreover, the 'Agents' approach also behaves better on the FashionMNIST dataset. As shown in Fig. 6, the accuracy also gets improved on each iteration. Similarly, as shown in Fig. 7, the trust score also increases more than in the 'Random' approach.

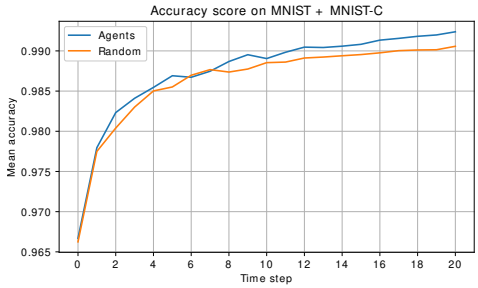

Fig. 4. Accuracy evaluation on MNIST+MNIST-C dataset

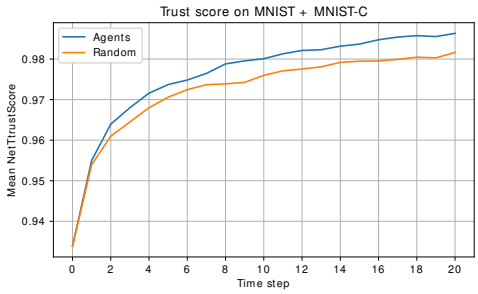

Fig. 5. Trust evaluation on MNIST+MNIST-C dataset

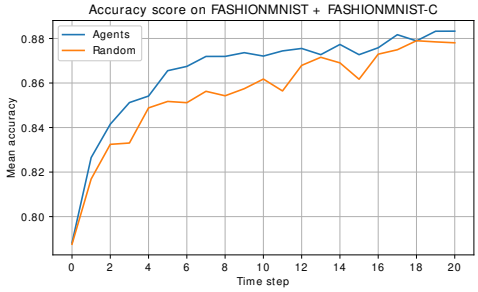

Fig. 6. Accuracy evaluation on FashionMNIST+FashionMNIST-C dataset

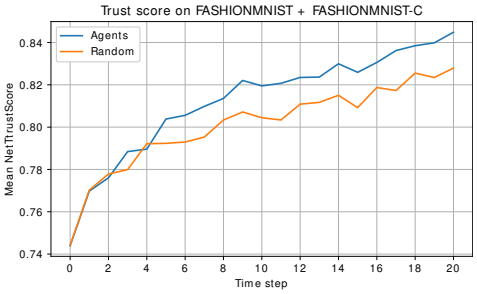

Fig. 7. Trust evaluation on FashionMNIST+FashionMNIST-C dataset

It is also relevant to analyze the improvement in disaggregated form, that is, how each new model behaves in each class (label). Figs. 8 and 9 show the trust spectrum for the 'Agents' approach (blue area) and the method using random choice (brown area). Regarding MNIST dataset, Fig. 8 shows that the model trust improves slightly in almost all the classes. Although on average the increase in trust is not that much, it indicates that certain new instances were more significant (anomalous instances) and the model can continuously improve to reduce its overconfidence.

Regarding FashionMNIST dataset, Fig. 9 shows that the improvement is greater in general, and especially in class 3 and class 5. So, we can say that not only the 'Agents' approach provides a higher number of correct answers (accuracy) but also is very confident about these answers. Due to the higher complexity of the dataset, the trust could be improved considerably, indicating that it is not the quantity of instances that matters but the quality of the instances to be labeled. However, we observe that both methods have low trust in class 6, indicating that further research should be done on this point for the production deployment of such models.

Finally, as for the execution time, the procedure triggered on each iteration for scenarios 1 and 2 took about 20 seconds and 6s in average, respectively, for both MNIST and FashionMNIST. Since the procedure is executed in the background, the difference is not very relevant.

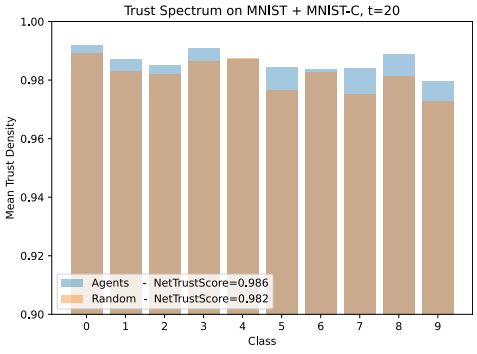

Fig. 8. Trust Spectrum on MNIST+MNIST-C dataset for 'Agents' and 'Random' scenarios for the latest model (timestep=20).

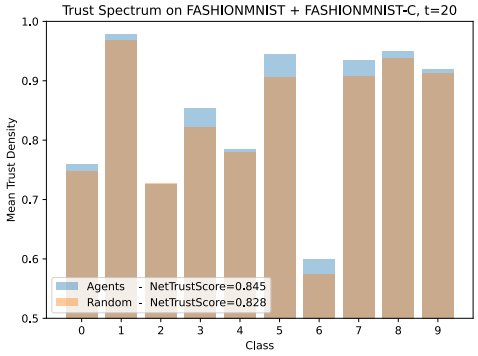

Fig. 9. Trust Spectrum on FashionMNIST+FashionMNIST-C dataset for 'Agents' and 'Random' scenarios for the latest model (timestep=20).

V. Conclusions and Future work

In this paper, we have discussed how to iteratively improve trust in a ML model. For this purpose, we proposed a multi-agent system that includes human and machine agents that interact with each other to detect anomalous instances through a symbolic rule based on the trust score of individual instances, augments them, and performs a retraining transferring the knowledge of the previous model. We evaluated this system using two well-known datasets in the field of computer vision, but our approach can be extrapolated to other datasets and ML fields since there is no modification in the model structure.

We showed that the great advantage of our agents is in their joint work on the model. The feedback that flows in each agent is the fundamental piece that improves both accuracy and trust. In contrast, the improvement in the random approach comes only from the labels that were added to the retraining.

Clearly, continuous intervention in ML models allows improving their behavior fast and efficiently. On the one hand, monitoring the trust of each instance during the inferences allows detecting any drift in the confidence of the predictions. On the other hand, well-defined rules contribute to guarantee the behavior of the model, in this case, by identifying the critical instances to be considered unsafe.

As future work, we will consider use cases beyond classification, such as regression and clustering. We also plan to create a way of learning agent behaviors. That is, start from a knowledge base in each agent and enable them to enhance their behavior through reinforcement learning to increase their reward using incremental improvements over time.

Acknowledgment

This work was supported by Lenovo as part of Lenovo-BSC 2020 collaboration agreement, by the Spanish Government under contracts PID2019-107255GB-C21 and PID2019-107255GB-C22, and by the Generalitat de Catalunya under contract 2017-SGR-1414 and under grant 2020 FI-B 00257.

References

[1] E. Beede, E. Baylor, F. Hersch, A. Iurchenko, L. Wilcox, P. Ruamviboonsuk, and L. M. Vardoulakis, "A Human-Centered Evaluation of a Deep Learning System Deployed in Clinics for the Detection of Diabetic Retinopathy," in *Proc. 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 2020, pp. 1–12.

[2] P. Branco, L. Torgo, and R. Ribeiro, "A Survey of Predictive Modelling under Imbalanced Distributions," 2015. [Online]. Available: https://arxiv.org/abs/1505.01658

[3] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the People: The Role of Humans in Interactive Machine Learning," *AI Magazine*, vol. 35, no. 4, pp. 105–120, Dec. 2014.

[4] J. A. Fails and D. R. Olsen, "Interactive Machine Learning," in *Proc. 8th Intl. Conference on Intelligent User Interfaces, IUI)*, 2003, pp. 39–45.

[5] D. Kahneman, *Thinking, fast & slow*. Farrar, Straus & Giroux, 2011.

[6] G. Booch, F. Fabiano, L. Horesh, K. Kate, J. Lenchner, N. Linck, A. Loreggia, K. Murugesan, N. Mattei, F. Rossi, and B. Srivastava, "Thinking Fast and Slow in AI," 2020. [Online]. Available: https://arxiv.org/abs/2010.06002

[7] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision," in *Proc. 7th International Conference on Learning Representations, ICLR*, 2019.

[8] M. M. Veloso and W. T. B. Uther, "The CMTrio-98 Sony-Legged Robot Team," in *RoboCup-98: Robot Soccer World Cup II*, ser. Lecture Notes in Computer Science, vol. 1604. Springer, 1998, pp. 491–497.

[9] K. Pietroszek, "Providing Language Instructor with Artificial Intelligence Assistant," *iJET*, vol. 2, no. 4, 2007. [Online]. Available: https://www.online-journals.org/index.php/i-jet/article/view/142

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[11] K. M. Khalil, M. Abdel-Aziz, T. T. Nazmy, and A.-B. M. Salem, "MLIMAS: A Framework for Machine Learning in Interactive Multi-agent Systems," *Procedia Computer Science*, vol. 65, pp. 827–835, 2015.

[12] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE - a FIPA-compliant agent framework," in *Proc. International Conference on Practical Applications of Intelligent Agents, PAAM*, 1999, pp. 97–108.

[13] R. W. Collier, E. O'Neill, D. Lillis, and G. O'Hare, "MAMS: Multi-Agent MicroServices," in *Proc. 2019 World Wide Web Conference, WWW*. ACM, 2019, pp. 655–662.

[14] A. Wong, X. Y. Wang, and A. Hryniowski, "How Much Can We Really Trust You? Towards Simple, Interpretable Trust Quantification Metrics for Deep Neural Networks," 2021. [Online]. Available: https://arxiv.org/abs/2009.05835

[15] B. Zadrozny and C. Elkan, "Transforming Classifier Scores into Accurate Multiclass Probability Estimates," in *Proc. 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2002, pp. 694–699.

[16] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," 2017. [Online]. Available: https://arxiv.org/abs/1612.01474

[17] Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," *IEEE Transactions on Computers*, vol. C-29, no. 12, pp. 1104–1113, 1980.

[18] G. Bravo-Rocca, P. Liu, J. Guitart, A. Dholakia, D. Ellison, J. Falkanger, and M. Hodak, "Scanflow: A multi-graph framework for machine learning workflow management, supervision, and debugging," *Expert Systems with Applications*, 2022.

[19] H. Jiang, B. Kim, M. Y. Guan, and M. Gupta, "To Trust or Not To Trust a Classifier," 2018. [Online]. Available: https://arxiv.org/abs/1805.11783

[20] B.-B. Gao, C. Xing, C.-W. Xie, J. Wu, and X. Geng, "Deep Label Distribution Learning With Label Ambiguity," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2825–2838, Jun 2017.

[21] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning," 2017. [Online]. Available: https://arxiv.org/abs/1712.04621

[22] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A Comprehensive Survey on Transfer Learning," 2020. [Online]. Available: https://arxiv.org/abs/1911.02685

[23] [dataset] Feiyang Chen, N. Chen, H. Mao, and H. Hu, "Assessing four Neural Networks on Handwritten Digit Recognition Dataset (MNIST)," 2018. [Online]. Available: https://arxiv.org/abs/1811.08278

[24] [dataset] Dan Hendrycks and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," 2019. [Online]. Available: https://arxiv.org/abs/1903.12261

[25] [dataset] Han Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1708.07747