© 2020 by the authors; licensee RonPub, Lübeck, Germany. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).

Research online Publishing www.ronpub.com

Open Access

Open Journal of Internet of Things (OJIOT)
Volume 6, Issue 1, 2020
http://www.ronpub.com/ojiot
ISSN 2364-7108

Towards Knowledge Infusion for Robust and Transferable Machine Learning in IoT

Jonathan Fürst, Mauricio Fadel Argerich, Bin Cheng, Ernö Kovacs

NEC Laboratories Europe GmbH, Kurfürsten-Anlage 36, Heidelberg, Germany,
{jonathan.fuerst, mauricio.fadel, bin.cheng, ernoe.kovacs}@neclab.eu

Abstract

Machine learning (ML) applications in Internet of Things (IoT) scenarios face the issue that supervision signals, such as labeled data, are scarce and expensive to obtain. For example, it often requires a human to manually label events in a data stream by observing the same events in the real world. In addition, the performance of trained models usually depends on a specific context: (1) location, (2) time and (3) data quality. This context is not static in reality, making it hard to achieve robust and transferable machine learning for IoT systems in practice. In this paper, we address these challenges with an envisioned method that we name Knowledge Infusion. First, we present two past case studies in which we combined external knowledge with traditional data-driven machine learning in IoT scenarios to ease the supervision effort: (1) a weak-supervision approach for the IoT domain to auto-generate labels based on external knowledge (e.g., domain knowledge) encoded in simple labeling functions. Our evaluation for transport mode classification achieves a micro-F1 score of 80.2%*, with only seven labeling functions, on par with a fully supervised model that relies on hand-labeled data. (2) We introduce guiding functions to Reinforcement Learning (RL) to guide the agents' decisions and experience. In initial experiments, our* guided reinforcement learning *achieves more than three times higher reward in the beginning of its training than an agent with no external knowledge. We use the lessons learned from these experiences to develop our vision of* knowledge infusion. *In knowledge infusion, we aim to automate the inclusion of knowledge from existing knowledge bases and domain experts to combine it with traditional data-driven machine learning techniques during setup/training phase, but also during the execution phase.*

Type of Paper and Keywords

Visionary paper: *Machine Learning, IoT, Robustness, Transfer Learning, Knowledge Infusion*

1 Introduction

The Internet of Things (IoT) is expanding rapidly in various sectors such as smart buildings, smart cities and smart transportation. At the same time, machine learning (ML) supported systems are increasingly used to provide meaningful insights by performing classification or prediction tasks on top of IoT data [32] and to efficiently orchestrate tasks between cloud and edge [11, 3]. Based on our practical experience, the IoT domain poses various challenges for generalizing traditional data driven ML approaches to provide robust and transferable results. This often has to do with changes in location, time and in data quality context as depicted in Figure 1:

1. **Location.** Different locations (e.g., separate sensor deployments in a single city or deployments in

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2020)* in conjunction with the VLDB 2020 conference in Tokyo, Japan. The proceedings of VLIoT@VLDB 2020 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

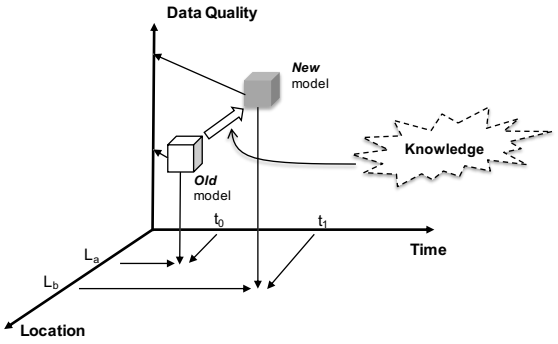


Figure 1: Context-dimensions of ML in IoT

multiple cities) behave differently. For example, the radio-frequency (RF) signal propagation depends heavily on the deployment location (e.g., through varying signal attenuation) [12].

2. **Time.** Deployment characteristics as well as data patterns change through time. I.e., a ML model that works well when training data is collected (e.g., in the lab), might degrade in performance throughout the lifetime of a deployment. Further, it is hard to quantify this degradation, because continuously obtaining ground truth data is expensive and deployment times comprise often several years.

3. **Data Quality.** In IoT deployments, data is frequently noisy, sparse and heavily imbalanced. E.g., an application might require the classification of relatively rare events such as road accidents or infrequent transport modes; a sensor network deployment might result in high variances in sampling frequency and missing data due to packet loss. As IoT devices have limited power and computation, collecting highly dense data is often not possible.

We believe that machine learning in the IoT domain needs to achieve robustness and transferability during setup/training as well as during execution phase in order to deal with such varying context.

1.1 Our Vision

In previous works, the transferability problem has been addressed often through transfer learning, which has the aim to transfer knowledge learned for one task to a related task. The robustness problem has been recently addressed in the context of safe learning systems (see Section 4 for a discussion of related work). In this paper, we envision a *Knowledge Infusion* method to address these issues: Instead of transferring trained knowledge to another task (as in transfer learning), Knowledge Infusion intends to transfer knowledge used for the training phase between tasks with little effort. We achieve this transferability through *adaptive weak and strong knowledge functions* that are interfacing with a knowledge graph and external knowledge sources (for example open knowledge bases such as OpenStreetMap [16] or Wikidata [36]) in order to adapt dynamically to a new context by querying these knowledge sources. Towards this goal, we propose two types of knowledge functions: weak knowledge functions, that are thought to be mostly true and strong knowledge functions that comprise axioms (canonical truths). We then combine these knowledge functions into a weak ensemble and a strong ensemble that together form a "white-box" knowledge model. Next, we use this knowledge model to train any traditional supervised machine learning model (e.g., using the labels generated with the knowledge model). As has been shown in previous works [25, 13], the machine learning model will usually perform better than the model/labels used for training it. The reason is that the machine learning model will learn to generalize the patterns found in the data while also partially removing noise in it.

In Knowledge Infusion, the "white-box" knowledge model is continuously updated (through new knowledge or added/updated knowledge functions) and then used to ensure robustness of our system during execution, while also identifying situations in which the context in the real world shifts away from the trained ML model and re-training is needed.

In this vision paper, we first present two promising previous experiences where external knowledge is exploited in the training process: (1) In supervised machine learning for a smart mobility use case by adapting the recently published weak-supervision framework Snorkel [25] to fit the IoT domain and (2) For reinforcement-learning, where we use external knowledge to guide agent exploration. These experiences and the lessons learned open up the way to our envisioned method for knowledge infusion (Section 3). We conclude our paper with a discussion of related work and an outlook on future work (Section 4 and Section 5).

2 Previous Experiences

In this section, we summarize two previous works in which we successfully used external knowledge in machine learning systems in the IoT domain: (1) In supervised machine learning for transport mode detection [13]; (2) for reinforcement-learning [4]. We take the lessons learned from these cases as a starting point to develop our vision of knowledge infusion.

2.1 Supervised Machine Learning: Transport Mode Detection

An important application of context-aware computing [1] is detecting transportation modes of people using mobile devices such as smartphones. Transport mode detection is a key enabler for physical activity monitoring as well as personal environmental impact monitoring [26, 21]. It is also critical to optimize urban multimodal human mobility [15] and to enable end-user applications, such as automated and individual CO2 emission tracking [30].

Transport mode detection requires a two step segmentation of sensor data to *trips* and *sections* (based on transport modes), and an accurate classification of these sections into modes (e.g., walking, bicycling, riding a bus). Previous works have proposed transport mode detection using smartphone sensors such as GPS [39], accelerometer [17], barometer [27], or combinations of these [26] and adding GIS data [33] to improve accuracy. Most of these works leverage supervised ML using labeled data points as training data. This labeled data is usually provided by users/participants of the studies.

2.1.1 Our Approach

We addressed the transport mode detection problem by adopting weak supervision techniques originally used in information extraction [25] to our problem. Figure 2 depicts our main building blocks.

First, sensor data (location and activity data) is sensed from user smartphones. In the *Segmentation Phase*, we then align time series from multiple sensors to the same sampling time and segment time series first into trips (by applying a dwell time heuristics) and section candidates (using a developed activity supported walk point segmentation algorithm). The outcome of this step is a set of section candidates, each candidate contains a time-series of location and activity data points.

Second, in the *Label & Training Phase*, we apply a set of labeling functions to these candidates. Each function encodes human heuristics and/or external knowledge (e.g., from OpenStreetMap) to "vote" on the transport class of a candidate section. We feed the resulting label matrix into Snorkel [25], which learns a generative model from all labeling functions and their votes on each candidate section, taking into account the underlying accuracy of each labeling function.

Finally, we use the generative model to label all candidate segments and train a discriminative ML model with the probabilistic labels. The ML model generalizes beyond the information encoded in the labeling functions. In the *Classification Phase*, this model is then used to classify incoming candidate section into transport modes. Based on this classification, we then re-segment by merging adjacent sections of the same mode and return the results (classified trips and sections) to users.

Table 1: F1 score of generated labels and end models against hand-labeled ground truth

Generative Model (generated Labels)	Weakly-supervised Random Forest	Fully-supervised Random Forest
74.1 %	80.2 %	81.0 %

2.1.2 Results

We validated our method against a dataset that we collect in the wild over a period of $4\,\mathrm{months}$, containing 300k datapoints from 8 end-users. We sense GPS location through iOS and Android Location API and accelerometer based activity data through the Activity API. Users have partially labeled data with a developed visual labeling tool. We use these data to evaluate our method, splitting our data in training (1/2) and test data (1/2). We classify four transport modes: walk, bike, car and train.

Our implementation uses seven labeling functions that combine external knowledge with sensor data to vote on the transport classification of a section. For example, we implement functions that use the sensed speed together with human heuristics on common speeds for different modalities to vote on the transport mode (`LF_median_sensed_speed` and `LF_quantile_sensed_speed`). We also integrate OpenStreetMap (OSM) and use the provided annotations of public transport stops (`LF_osm`). We train ML models using the data labeled by the generative model, oversampling underrepresented classes with SMOTE [8].

We test several classifiers, from simple linear models to neural networks and observe the best performance with Random Forests (RF), achieving 80.2 % in F1 score.

We compare our results against a traditional supervised approach that uses the hand-labeled data of the training split results and results in an only marginally better F1 score of 81.0 %. Table 1 summarizes our overall results.

2.2 Reinforcement Learning: IoT-ML Pipeline Orchestration

Reinforcement Learning (RL) has proven its ability to deal with complex problems, achieving super-human performance in some cases [20, 35, 18]. However,

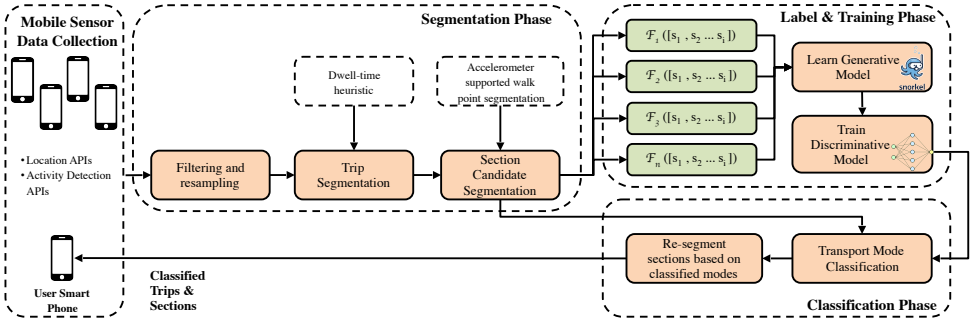


Figure 2: Transport mode detection steps

its applicability in real-world problems is still lagging for a number of reasons, among which we highlight: (1) learning on the real system from limited samples, (2) safety constraints that should never or at least rarely be violated. Several approaches have emerged to address these problems such as (1) learning from human demonstrations or historical data [7], (2) calculating uncertainty in decisions to manage safety [19] and combinations with supervised learning and human intervention [2].

Recently, researchers have applied RL to computer systems tasks, such as database management system configuration [28] or container orchestration for Big Data and IoT systems [3]. In this domain, the previously mentioned approaches are not applicable or inconvenient because (1) data from the system might not be available if the system is new, and even when data is available, the system deployment, load and parameters vary, making these data partially true at best; in addition, (2) the cost of incorrect actions might be high when dealing with systems in production environments, training a supervised model is not possible for the reasons mentioned before, and the reaction of a human operator is expensive and slow to accommodate the system performance in a timely manner.

2.2.1 Our Approach

To address these problems and make RL applicable to real-world problems, we have introduced Tutor4RL [4]. Figure 3 shows our overall design. We add a new component to the regular RL framework called *Tutor* to guide the agent's decisions when the agent has no or little experience. The tutor is able to guide the agent to take reasonable decisions because it can directly leverage a set of *knowledge functions* defined by domain experts. In this way, Tutor4RL improves greatly the performance of the agent in its early phase.

The tutor possesses external knowledge and interacts with the agent during training. The tutor takes as input the state of the environment, and outputs the action to take; in a similar way to the agent's policy. However, the tutor is implemented as programmable functions, in which external knowledge is used to decide the mapping between states and actions, e.g., for Atari Breakout, the tutor takes the frame from the video game as input, and outputs in what direction the bar should be moved. For every time step, the tutor interacts with the agent and gives advise to the agent for making better decisions, based on all provided knowledge functions.

In order to decide when to use the policy's decisions or the Tutor's guides, we define the parameter τ that assumes a value in the range $(0, 1)$, which is linearly decreased during training. τ is a parameter of our model and the best value to initialize it depends on the use case; thus its initial value is left to be decided during implementation.

The tutor's behavior provides a guide to the agent, but the agent can still improve upon it, reaching an even better performance than the one defined by experts. This is achieved thanks to two mechanisms that are already present in RL: (1) empirical evaluation of decisions and (2) ϵ-greedy exploration. In (1), if the action suggested by the Tutor is incorrect, i.e., it provides a negative reward, the agent will learn that this action is wrong and should therefore, not be used for the given state. In the case the reward is positive but could still be improved, the agent can still find an action which will return a higher reward thanks to maintaining an ϵ-greedy exploration (2). However, the Tutor knowledge is used heavily in the initial phase, more than exploration, so even when an incorrect random action is chosen, the Tutor knowledge can correct this action quickly in most cases. The initial value for ϵ can be chosen according to the use case and how high is the cost of an incorrect action.

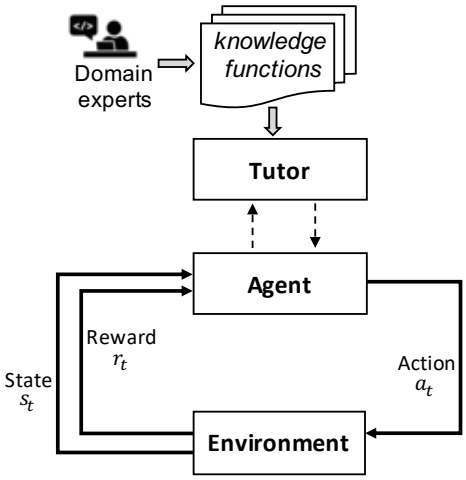


Figure 3: Overall working of Tutor4RL

2.2.2 Results

In order to leverage existing methods and libraries, we chose a well-known task in RL to evaluate our approach: Atari games. We implement Tutor4RL for a DQN [20] agent, one of the most popular and successful approaches to solve this kind of tasks in literature. We use the DQN implementation provided in the library Keras-RL [24] with Tensorflow and apply it to the game Breakout, using OpenAI Gym [6]. To evaluate the performance of our approach, we compare it to a regular DQN agent and use the same set of parameters for both agents; DQN with Tutor4RL and plain DQN.

Figure 4 depicts the mean reward per episode of both agents during training. Until step $500,000$, the plain DQN Agent shows a predictable low reward (< 15 points), while the DQN Agent with Tutor4RL—thanks to the use of its tutor's knowledge—manages to achieve a mean reward between 15 and 35 points, almost double the maximum of the plain DQN Agent. From step $500,000$ we see the plain DQN agent improves thanks to acquiring more experience, until finally in step 1.1M the plain DQN agent achieves similar results to the tutored one. From there on we see a similar reward for both agents, with the tutored agent achieving a slightly higher mean reward.

To evaluate the performance of the learned policy, we test both agents after 1.75M steps with $\epsilon = 0.05$ and $\tau = 0$, so no tutor knowledge is used. We see that the plain DQN agent achieves an average reward of 40.75 points while the agent trained with Tutor4RL achieves a reward of 43.

2.3 Lessons Learned

Both, our experience with weak-supervision for a typical classification problem in the Internet of Things domain, as well as applying external knowledge in form of guiding functions to the reinforcement learning process have shown promising results. For transport mode detection, we were able to encode external knowledge into 7 simple functions, potentially reducing the requirement for huge amounts of hand-labeled data in supervised learning scenarios. We also bootstrapped the performance of an inexperienced agent in reinforcement learning cases through external knowledge. What is currently missing to tackle the challenges of dealing with different context in IoT are three aspects:

1. Self-adaptive functions. Functions need to become adaptive based on the current context. E.g., a function for transport mode detection needs to adapt to different locations and their context (e.g., different speed limits) automatically.

2. Notion of weak and strong functions. We experienced that often humans can easily define some conditions which they know to be true. We need to have a notion to transfer this strong knowledge into the training and execution phase of machine learning.

3. Common method for both training and execution phase. We need to be able to introduce knowledge into both the training/setup phase of machine learning, but also continuously during execution in order to ensure robustness throughout changing contexts.

Based on these lessons learned, we present an initial idea for our knowledge infusion method that improves on these aspects in the next section.

3 KNOWLEDGE INFUSION

Knowledge Infusion aims to dynamically infuse knowledge, i.e., logic based on human reasoning and internal and external knowledge bases (e.g., stored in a knowledge graph) into teacher-based machine learning methods (i.e., supervised learning and reinforcement learning) to improve overall robustness, transferability and accuracy. The infusion of weak and strong knowledge has two benefits: (1) it reduces the effort needed to train or startup a ML model/agent and (2) the knowledge model can be executed side-by-side with the ML model/agent to correct wrong outputs, thereby improving robustness, and enabling the calculation of an uncertainty value that gives an indication of when

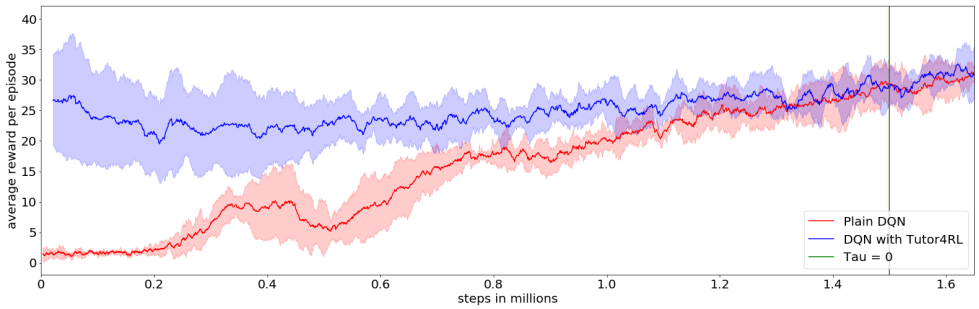


Figure 4: Average mean reward per episode achieved by plain DQN agent and DQN agent with Tutor4RL during training. Data was averaged over 4 tests for each agent and with a rolling mean of 20 episodes, bands show 0.90 confidence interval.

reality and the ML model have shifted too much apart so that performance would suffer.

3.1 Application Areas

Some important application areas that can benefit from Knowledge Infusion in the IoT domain are:

- *Smart city analytics and control (actuation).* Cities are increasingly becoming smart, connecting heterogeneous sensing and actuating infrastructure through smart city platforms. The key promise of such platforms is to improve certain aspects of a city (e.g., traffic, air quality, life quality, waste reduction, etc.) in an informed manner, based on the processing of sensor data and the machine learning processing pipelines for this data. Knowledge infusion helps to: (1) build these pipelines efficiently; (2) enable a transfer of pipelines between cities and automatically adapt them based on the context provided through the queried knowledge (for example adapt models based on knowledge of speed limits and public transport stops); (3) achieve robust ML systems, which outputs can be directly applied through actuations in the city (e.g., to traffic lights, smart parking or to guide public transport planning).

- *Smart retail decisions.* The retail sector is increasingly using machine learning to optimize their stock to fit the expected customers' purchases. Knowledge infusion enables shops to develop a set of strong knowledge functions to ensure base availability of certain products or the availability of some seasonal/holiday specific products (e.g., seasonal sweets for Christmas holidays). Such predictions can be hard to learn for a ML model, but further seasons/holidays are often distinct in different countries (for example, orthodox Christmas is later than roman-catholic Christmas). In such scenario, knowledge functions can automatically adapt themselves by querying a knowledge base (knowledge graph) that contains the particular holiday dates (for example Wikidata, or an internal knowledge base).

- *Energy management for buildings.* Buildings are one of the prime consumers of energy as humans in many countries spend most of their time inside them. Intelligently managing the energy consumption in buildings can save large amounts of energy especially for heating, ventilation and cooling (HVAC) [10]. The most advanced energy management systems for buildings have applied machine learning (for example they have been controlling HVAC based on an occupancy prediction) [14]. However, commercial buildings have strong safety regulations that specify certain conditions that always need to be met (e.g., a minimum flow of ventilation). With our method, these regulations can be encoded in strong knowledge functions. They can be updated dynamically to regulations in different states or countries through the knowledge base. Our combination of knowledge model (the combined ensemble of weak and strong knowledge functions) and ML model can safely be applied to actuate the energy management system of a building, ensuring robustness in relation to the current regulations.

In the following, we describe our envisioned method through two steps: (1) Setup Phase and (2) Execution Phase.

3.2 Setup Phase

We infuse knowledge through adaptive weak and strong knowledge functions. Knowledge functions output a

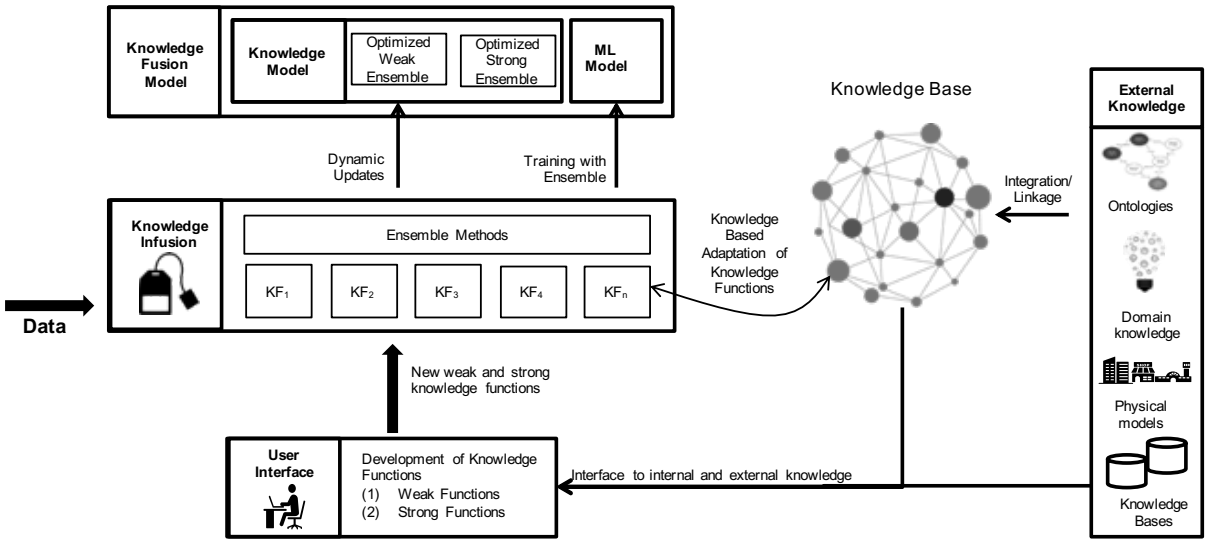


Figure 5: Knowledge infusion overview and setup

single value, which can either be a label (for supervised machine learning) or an action (for reinforcement learning). Functions consist of a reasoning part, defined by human provided logic (e.g., conditional operators) and input knowledge that is derived/queried from internal and external knowledge bases through well-defined interfaces (e.g., through SPARQL, other graph query languages such as GraphQL or IoT specific standards such as NGSI-LD [9]). Through such well-defined interface, knowledge functions are practically able to adapt their behavior automatically, based on changes in the input derived from the knowledge bases. For example, a knowledge function that outputs if there is a fire based on values from a temperature sensor can interface with a knowledge base that contains current weather data, including the current temperature. The reasoning logic inside the knowledge function might then use the weather temperature and the value from the temperature sensor to decide if there is a fire (binary classification, fire: True/False). It becomes clear that such function would adapt to different weather conditions automatically.

Weak knowledge functions are thought to be mostly true, while strong knowledge functions comprise axioms (canonical truths). Strong functions can be used as a fail-safe mechanism for the ML model, correcting outputs that experts know as a fact, cannot be true. In the previous example, we know that if temperature from the temperature sensor is within a margin of error from the weather temperature, then there is no fire. Even though the ML model will work correctly almost always in this case, it could happen that a very high, unusual weather temperature is reached in a hot wave and the model incorrectly indicates there is a fire. In this case, our strong function will correct the output.

In knowledge infusion, we create an ensemble of these knowledge functions into a knowledge model and use it to label data that we later use to train a supervised ML model with it. In RL, this step is not needed but instead, the knowledge model will guide the exploration of the agent while training its policy. During execution, both are then used jointly, i.e., both are processing every new input. See Figure 5 for an overview of these setup steps. As depicted in the figure, external knowledge might either be directly accessed by the knowledge functions (e.g., a weather API) or integrated/linked into a common knowledge base (e.g., a knowledge graph that integrated various aspects of a smart city).

3.3 Execution Phase

During the execution phase, new data is processed by both the ML model and the knowledge model. As described previously, the knowledge model allows us now to correct some obviously wrong outputs of the ML model (improving overall robustness) and to calculate an uncertainty value that enables knowledge infusion systems to understand if the reality has drifted away from the ML model. Such uncertainty value can be calculated based on the probability values obtained from the ML model as well as the windowed disagreement between the knowledge and the ML models. We then use a moving average of this uncertainty value to decide if we should add the output to the knowledge base when the output is certain, or if we need to re-train the ML model. As the re-training will occur with an updated knowledge model, the resulting ML model will then better fit to reality, resulting in improved performance.

4 Related Work

Related work to knowledge infusion can be found in the machine learning sub domains of transfer learning, weakly-supervised learning and safe/robust or trustworthy learning.

Transfer learning, aims to transfer knowledge learned for one task to a related task [23]. Transferring such knowledge has the benefit of reducing the training effort, both from computing time and from data collection and labeling time (state of the art models in computer vision or NLP have been trained with millions of datapoints and the training compute time can costs millions of dollars for large models [31]). Transfer learning can, for example, be achieved by replacing the final layer of a large deep neural network (e.g., a model trained with millions of images for object detection) to fit the model to the new task (e.g., a more specific object recognition for a single domain, such as smart appliances in a building). Further, existing approaches are fine-tuning larger models to fit to a new task, by re-training only few layers, while leaving the more task-unspecific layers of the neural network fixed [29, 22]. In addition, hierarchical approaches in RL aim to break down the task in sub-tasks and introduce the use of sub-policies to solve them. Once these sub-policies are trained, they can be re-used in similar tasks to the original one [35].

Weakly-supervised learning deals with the lack of labeled training data in machine learning systems through the use of noisy and higher-level supervision signals [40]. With Snorkel, the authors recently proposed a framework for weak supervision in which users, such as domain experts, write labeling functions that are then combined in a generative process [25]. Snorkel has been successfully applied and adapted to tasks such as knowledge base construction from transistor data sheets [38], product classification [5] up to our own recent application to transport mode detection [13].

Safe learning systems, such as safe reinforcement learning ensure that the current system state is bound to a terminal set of states, that is known to be safe [37]. Uncertainty estimates are also used to limit the risks assumed by RL agents, regulating the exploration of actions in early learning phases as well as in new situations in which the agent has not experience and is uncertain about how to act [19]. Another related emerging field is trustworthy machine learning, which among others, tries to improve adversarial robustness to protect against adversaries which can fool traditional machine learning systems.

Knowledge infusion differs from these works. Instead of transferring trained knowledge to another task (as in transfer learning), Knowledge Infusion intends to transfer knowledge used for the training phase between tasks with little effort. Compared to existing weak supervision approaches, we employ our knowledge model not only during the training phase, but use it during execution. By introducing the concept of adaptive weak and strong knowledge functions, we are able to provide robustness during execution, while triggering a re-training of the ML model in case of a context shift. With our approach we are partially inspired by Leslie Valiant's early work on knowledge infusion [34]. In the future, we intend to evaluate to which extend his ideas of combining reasoning techniques with learned knowledge can be introduced to our method.

5 Conclusion and Future Work

We presented some initial experiences in which external knowledge has been introduced to a data-driven machine learning process. Our experiences for supervised machine learning and reinforcement learning and the lessons learned with them open up our vision of Knowledge Infusion. Knowledge Infusion aims to improve the robustness, transferability and accuracy of machine learning in IoT systems. Towards that goal, we proposed two types of knowledge functions: (a) weak knowledge functions, that are thought to be mostly true and (b) strong knowledge functions that comprise axioms (canonical truths). The purpose of these functions is two-fold: (1) They enable the infusion of knowledge into the training phase of supervised machine learning methods through ensemble methods; (2) They act as a white box counter-part to the black-box trained model/agent of a supervised or reinforcement learning system during the execution phase that (a) can identify and correct (infrequent) wrong outputs of the ML model/agent (by using the set of strong knowledge functions) and (b) can identify through the calculation of uncertainty values when a model has drifted away from the input data characteristics and must be re-trained with new data to re-gain adequate performance.

We have already started to build such Knowledge Infusion system and intend to evaluate it on various use cases in the IoT domain. Knowledge Infusion opens up many interesting avenues for future research on how and to which extend to combine data-driven machine learning techniques with knowledge and reasoning.

Acknowledgements

The research leading to these results has received funding from the European Community's Horizon 2020 research and innovation programme under grant agreement n° 779747.

REFERENCES

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *International symposium on handheld and ubiquitous computing*. Springer, 1999, pp. 304–307.

[2] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[3] M. F. Argerich, B. Cheng, and J. Fürst, "Reinforcement learning based orchestration for elastic services," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, 2019, pp. 352–357.

[4] M. F. Argerich, J. Fürst, and B. Cheng, "Tutor4rl: Guiding reinforcement learning with external knowledge." in *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering (1)*, 2020.

[5] S. H. Bach, D. Rodriguez, Y. Liu, C. Luo, H. Shao, C. Xia, S. Sen, A. Ratner, B. Hancock, H. Alborzi *et al.*, "Snorkel drybell: A case study in deploying weak supervision at industrial scale," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 362–375.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[7] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[9] ETSI GS CIM 009, "Context Information Management (CIM); NGSI-LD API," https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.02.01_60/gs_CIM009v010201p.pdf, 2019.

[10] J. Fürst, *IoT Based Human-building Interaction*. IT-Universitetet i København, 2017.

[11] J. Fürst, M. F. Argerich, K. Chen, and E. Kovacs, "Towards adaptive actors for scalable iot applications at the edge," *Open Journal of Internet Of Things (OJIOT)*, vol. 4, no. 1, pp. 70–86, 2018, special Issue: Proceedings of the International Workshop on Very Large Internet of Things (VLIoT 2018) in conjunction with the VLDB 2018 Conference in Rio de Janeiro, Brazil. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:101:1-201808051903887853107

[12] J. Fürst, K. Chen, H.-S. Kim, and P. Bonnet, "Evaluating bluetooth low energy for iot," in *2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*. IEEE, 2018, pp. 1–6.

[13] J. Fürst, M. Fadel Argerich, K. Shankari, G. Solmaz, and B. Cheng, "Applying Weak Supervision to Mobile Sensor Data: Experiences with TransportMode Detection," in *AAAI-20 Workshop on Artificial Intelligence of Things*, New York, New York, USA, feb 2020.

[14] J. Fürst, G. Fierro, P. Bonnet, and D. E. Culler, "Busico 3d: building simulation and control in unity 3d," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, 2014, pp. 326–327.

[15] R. Gallotti and M. Barthelemy, "Anatomy and efficiency of urban multimodal mobility," *Scientific reports*, vol. 4, p. 6911, 2014.

[16] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[17] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proceedings of the 11th ACM conference on embedded networked sensor systems*. ACM, 2013, p. 13.

[18] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman *et al.*, "Human-level performance in 3d multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, 2019.

[19] B. Lötjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8662–8668.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, 2015.

[21] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda, "Peir, the personal environmental impact report, as a platform for participatory sensing systems research," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 55–68.

[22] A. Ng, "Nuts and bolts of building ai applications using deep learning," *NIPS Keynote Talk*, 2016.

[23] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[24] M. Plappert, "keras-rl," https://github.com/keras-rl/keras-rl, 2016.

[25] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *The VLDB Journal*, vol. 29, no. 2, pp. 709–730, 2020.

[26] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2, p. 13, 2010.

[27] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh, "Using mobile phone barometer for low-power transportation context detection," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 2014, pp. 191–205.

[28] M. Schaarschmidt, A. Kuhnle, B. Ellis, K. Fricke, F. Gessert, and E. Yoneki, "Lift: Reinforcement learning in computer systems by learning from demonstrations," *arXiv preprint arXiv:1808.07903*, 2018.

[29] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[30] K. Shankari, J. Fuerst, M. Fadel Argerich, E. Avramidis, and J. Zhang, "Mobilitynet: Towards a public dataset for multimodal mobility research," in *Climate Change AI 2020 (Spotlight Talk)*. ICLR, 2020.

[31] O. Sharir, B. Peleg, and Y. Shoham, "The cost of training nlp models: A concise overview," *arXiv preprint arXiv:2004.08900*, 2020.

[32] G. Solmaz, E. L. Berz, M. F. Dolatabadi, S. Aytaç, J. Fürst, B. Cheng, and J. d. Ouden, "Learn from iot: pedestrian detection and intention prediction for autonomous driving," in *Proceedings of the 1st ACM Workshop on Emerging Smart Technologies and Infrastructures for Smart Mobility and Sustainability*, 2019, pp. 27–32.

[33] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu, "Transportation mode detection using mobile phones and gis information," in *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2011, pp. 54–63.

[34] L. G. Valiant, "Knowledge infusion," in *AAAI*, vol. 6, 2006, pp. 1546–1551.

[35] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3540–3549.

[36] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[37] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *arXiv preprint arXiv:1906.10417*, 2019.

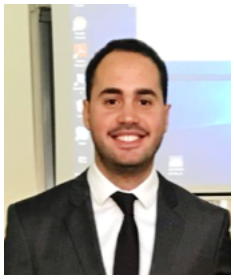
[38] S. Wu, L. Hsiao, X. Cheng, B. Hancock, T. Rekatsinas, P. Levis, and C. Ré, "Fonduer: Knowledge base construction from richly formatted data," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 1301–1316.

[39] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pp. 312–321.

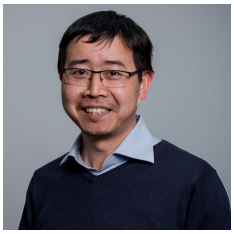
[40] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 2018.

AUTHOR BIOGRAPHIES

Jonathan Fürst is a Research Scientist in the IoT Platform group at NEC Laboratories Europe. Previously he was a PostDoc at IT University of Copenhagen (ITU) in Denmark. He holds a Ph.D. and a M.Sc. from ITU advised by Prof. Philippe Bonnet and a B.Eng. in Industrial Engineering at University of Aalen (HTW Aalen). During his Ph.D., he spent six months as a visiting researcher at UC Berkeley in AMPLab (Software Defined Buildings group) advised by Prof. Randy Katz and Prof. David Culler. His interests are in the area of IoT systems and practical applications such as smart buildings & cities, indoor localization and augmented reality. His current research focuses on developing better abstractions to program and run IoT applications easily and efficiently from edge to cloud and automated data integration and knowledge extraction using machine learning.



Mauricio Fadel Argerich is a Research Associate at NEC Laboratories Europe. He has received his Master's degree in Data Science from University of Rome and his Engineering degree in Information Systems from the National Technological University (UTN), Argentina. He has worked as a Software Engineer, being involved in the development of multiple web and mobile systems. His research focuses on methods and applications of Machine Learning for IoT analytics and context-aware services, with particular interest in Reinforcement Learning.



Dr. Bin Cheng is a senior researcher in the group of IoT Platform at NEC Laboratories Europe. He has been leading the design and implementation of an open source edge computing framework called FogFlow, which is a unique generic enabler to support serverless fog computing in the FIWARE ecosystem for smart cities and smart industry. He also led the design and implementation of a big data and analytics platform for the Santander city, which has been already in production use. His recent research interests include Edge AI, serverless fog computing, and knowledge extraction for IoT platforms. He had publications at several top-tier journal and conferences such as IEEE IoT Journal, EuroSys, NSDI, and IMC.

Ernö Kovacs received his Ph.D from the University of Stuttgart in 1996. In 1997, he joined Sony's R&D lab in Stuttgart and was responsible for research projects in mobile multimedia, multi-access service portals (integrating mobile and broadband access), context-aware services, adaptation technologies for services and content, Bluetooth PAN standardization, and middleware for UWB-based ad-hoc networks. In 2005, he joined NEC's "Networking Laboratories" as Senior Manager for "Cloud Services and Smart Things". His group works on Cloud Computing, IoT analytics, self-organisation and context-aware services. He was a leading architect in the SPICE and in the MAGNET Beyond project. He is currently contributing to the FIWARE, FIESTA and Mobinet projects. He was an advisor to the Singapore Smart Nation program in the Functional Specification round table and was leading NECs engagement in the Safe City Singapore test bed. He has publications in various journals (including "IEEE Communication Magazine" and "IEEE Personal Communications"), conferences and books. He holds 27 granted patents.