Learning on the Job: Online Lifelong and Continual Learning

Bing Liu

Peking University and University of Illinois at Chicago
dcsliub@pku.edu.cn, liub@uic.edu

Abstract

One of the hallmarks of the human intelligence is the ability to learn continuously, accumulate the knowledge learned in the past and use the knowledge to help learn more and learn better. It is hard to imagine a truly intelligent system without this capability. This type of learning differs significantly than the classic machine learning (ML) paradigm of *isolated single-task learning*. Although there is already research on learning a sequence of tasks incrementally under the names of *lifelong learning* or *continual learning*, they still follow the traditional two-phase separate training and testing paradigm in learning each task. The tasks are also given by the user. This paper adds *on-the-job learning* to the mix to emphasize the need to learn during application (thus *online*) after the model has been deployed, which traditional ML cannot do. It aims to leverage the learned knowledge to discover new tasks, interact with humans and the environment, make inferences, and incrementally learn the new tasks on the fly during applications in a *self-supervised and interactive* manner. This is analogous to human on-the-job learning after formal training. We use chatbots and self-driving cars as examples to discuss the need, some initial work, and key challenges and opportunities in building this capability.

Introduction

The classic machine learning (ML) paradigm works by running an ML algorithm on a given training dataset to learn a model. The model is then deployed and used in an application. This paradigm has at least two major issues.

1. The learning process does not retain and use the previously learned knowledge. This paradigm is called ***isolated single-task learning*** (Chen and Liu, 2016). We humans never learn in isolation or from scratch. We learn continually, retain the learned knowledge, and use it to help future learning. Over time we become more and more knowledgeable and better and better at learning. Without accumulating the past knowledge and leveraging it in new task learning, an ML algorithm needs a huge amount of labeled data. We humans can learn very well with only a few examples. Labeling of data is often done manually, which is very time-consuming. As the world is too complex with too many tasks, which also change constantly, it is impossible to label a large amount of data constantly for every possible task.
2. There is no learning after the model is deployed in an application. Human learning is different as we continue to ***learn on the job*** after formal training. Studies have shown that about 70% of human knowledge is learned while working on a task or on the job. Only about 10% is learned through formal training and the rest 20% is learned through imitation of others. An AI system should also ***learn on the job*** during model applications.

Lifelong (or *continual*) *learning* (LL) attempted to imitate the human continuous learning process by learning a sequence of tasks incrementally, accumulate the learned knowledge, and adapt/use it to help future learning (Chen and Liu, 2016). However, the current LL methods still work in an *offline mode* and cannot do ***on-the-job learning*** after model deployment. Their tasks and training data are also provided by human users. This paper focuses on *on-the-job learning*, which may be regarded as *online LL*.

Definition: *On-the-job learning* studies (1) how to continuously discover new tasks by the agent itself, (2) gather training data also through the agent's own active effort, and (3) incrementally learn the new tasks during model application *without interrupting the application*.

Some works has been done on (1) and (3) under *open-world learning* and *continual learning*, respectively. However, little work has been done on (2). We propose ***interactive self-supervision*** **(ISS)** for labeled training data gathering via the agent's own inference, imitation, and natural interactions with humans and the environment. We should note that ISS is not a kind of unsupervised methods.

We use **chatbots** and **self-driving cars** as examples to discuss the necessity for on-the-job learning, some initial work, and major challenges. These systems all have to face the real world that is full of unknowns and they have to learn on the job (during actual conversation or driving) in order to function well because it is impossible to know what a person may say or what a car may see on the road in order to train the systems completely offline or beforehand.

It is hard to imagine a truly intelligent system without the *lifelong learning* and *on-the-job learning* capabilities.

Lifelong Learning and Continual Learning

This section introduces *lifelong learning* and *continual learning* (Chen and Liu, 2018). They basically mean the same thing, but the past research under the two names focuses on different aspects of the same problem.

Lifelong Learning (LL)

LL is defined in (Chen and Liu, 2016) as follows (based on (Thrun and Mitchell, 1995; Thrun, 1996; Silver, Yang, and Li, 2013; Ruvolo and Eaton, 2013; Chan and Liu, 2014):

Definition: At any time point, the learner has learned a sequence of N tasks, T_1, T_2, …, T_N. When faced with the (N+1)th task T_{N+1}, the learner can leverage the knowledge in the knowledge base (KB) to help learn T_{N+1}. KB maintains and accumulates the knowledge learned from the previous N tasks. After the completion of learning T_{N+1}, KB is updated with the knowledge gained from learning T_{N+1}.

We see that the goal of LL is to leverage the knowledge learned in the past to learn the new task T_{N+1} better.

Continual Learning (CL)

The term *continual learning* (CL) is more commonly used than LL in the deep learning community. The focus of CL is to solve *catastrophic forgetting* (CF) (Li and Hoiem, 2016; Seff et al., 2017; Shin et al., 2017; Kirkpatrick et al., 2017; Rebuffi et al., 2017; Lee et al., 2017; He and Jaeger, 2018; Yoon et al., 2018; Masse et al., 2018; Schwarz et al., 2018; Hu et al., 2019). CF means that when a neural network (NN) learns a sequence of tasks, the learning of each new task is likely to change the weights learned for previous tasks, which degrades the model accuracy for the previous tasks (McCloskey and Cohen, 1989). Human brains have the remarkable capability of learning a large number of tasks incrementally with little interference among them.

There are two main CL settings:

Class continual learning (CCL): In CCL, each task consists of one or more classes to be learned together but only one model is learned to classify all classes so far. In testing, a test instance from any class may be presented to the model for it to classify with no *task* information.

Task continual learning (TCL). In TCL, each task is a separate classification problem (e.g., one classifying different breeds of dogs and one classifying different types of birds). TCL builds a set of classification models (one per task) in *one* neural network. In testing, the system knows which task each test instance belongs to and uses only the model for the task to classify the test instance.

Deal with Forgetting and Improve Learning

Ideally, we would like LL or CL to achieve both objectives i.e., (1) learning the new task better (2) without forgetting the past models. Clearly, not all problems can achieve both. It is not obvious that different tasks or classes can help each other for CCL (except feature sharing). For TCL, if the tasks are entirely different, it is hard to improve the new task learning either. For example, one task is to classify whether one has a heart disease or not and another is to classify whether a loan application should be approved. Since the two tasks have little similarity, they have little knowledge sharing, and thus cannot help each other much. In these cases, CF is the only problem to solve.

However, for TCL, if the tasks are similar or share many aspects, then it is possible to achieve both objectives. Sentiment classification (SC) (Liu, 2012) is a good example because different tasks in SC have a lot of knowledge sharing, e.g., sentiment words/phrases (e.g., good, great, bad, and terrible) are similar across tasks. Each SC task is a separate classification problem that classifies whether a product review expresses a positive or negative sentiment. Lv et al. (2019) proposed a method, called SRK, which deals with CF and also improves the accuracy of the new task learning significantly. Additional experimental results show that the accuracy of the models of previous tasks also improves by 3% on average. But for some tasks, the results are also worse off significantly, which indicate some forgetting. It is interesting to deal with CF for these tasks.

On-the-Job Learning

Existing LL/CL techniques still mainly take the traditional batch or offline training and testing approach. They perform no learning after the learned model is deployed in its intended application, i.e., no learning while working on a task or on the job. As we discussed in the introduction section, we humans learn much of our knowledge on the job.

Note that there is a related ML paradigm called *online learning*, where the training examples arrive in a sequential order and when a new labeled example arrives, the existing model is quickly updated to produce the best model so far. On-the-job learning includes online learning but focuses on the more challenging task of continuous learning in the open environment where there are unseen objects or unexpected scenarios. Spotting and learning them *on the fly* is the core (including how to obtain class labels of the data).

Classic ML makes the *closed world assumption*.

Closed world assumption: The classes seen in testing or application must have been seen in training.

This means that in applications, the system cannot see anything new. This is not always true because in many applications, by nature the system will almost certainly encounter instances of unseen classes or unexpected situations that it has not been trained for. For example, during driving, a self-driving car is very likely to see new objects or unexpected scenarios (or corner cases). A chatbot will certainly encounter a user intent that it has never learned or an utterance that it cannot understand. In such cases, the system works in an ***open world*** environment in contrast to the ***closed world*** environment where the traditional ML operates. In the open world, the system has to learn on the job.

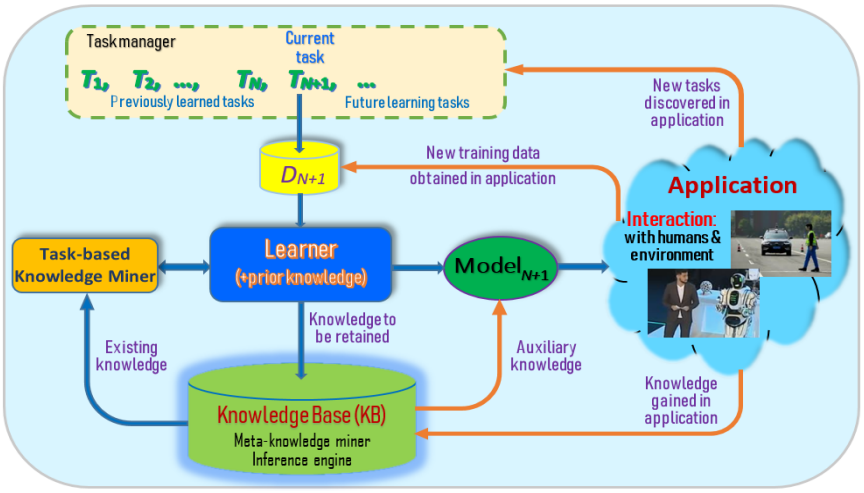

Figure 1: Architecture of lifelong learning incorporating on-the-job learning (best viewed in color).

Note: T_1 … T_N are the previously learned tasks, T_{N+1} is the current new task to be learned and D_{N+1} is its training data. The Learner learns by leveraging the relevant prior knowledge identifying by Task-based Knowledge Miner from the Knowledge Base (KB), which contains the retained knowledge in the past. The orange-colored lines indicate on-the-job learning.

Proposed New Lifelong Learning Architecture

As on-the-job learning is, by nature, a continuous learning process, we incorporate it into the lifelong learning definition and architecture in (Chen and Liu, 2016; 2018). The new architecture is given in Figure 1. The orange colored links reflect the on-the-job learning. We can see that during model application, the system can discover new tasks to be learned, new training data for learning, and new knowledge (auxiliary knowledge) to be added to the KB that may be leveraged in future learning or to improve the current model (Shu et al., 2017b). Note that dealing with CF is not reflected in the architecture as it stays in the algorithm of the Learner. Note also we do not present a new definition of lifelong learning as it is fairly easy to see from Figure 1.

Main Steps of On-the-Job Learning

We now discuss on-the-job learning in detail. We use supervised learning as an example to illustrate the essential steps. Later, we will discuss two use cases: learning during a dialogue or conversation and during driving of a self-driving car, where we will also see other forms of learning.

1. **Detect instances of unseen classes**: On-the-job learning starts with the system detecting instances of unseen (in training) classes, i.e., unexpected things, which form a new task to be learned incrementally. Formally, the problem is stated as follows (Chen and Liu, 2018):

   At any point in time, the learner has built a multi-class classifier F_N based on past N classes of data D_1, D_2, …, D_N with their corresponding class labels l_1, l_2, …, l_N. F_N can classify each test/application instance to either one of the N known classes or reject it and put it a set R.

   Several researchers have studied this problem, e.g., Scheirer et al. (2013), Bendale and Boult, 2015; Fei and Liu (2016), De Rosa et al (2016), and Xu et al. (2019).

2. **Identify unseen classes and gather training data**: Once a set of instances R are found to be from some unseen classes, the system identifies the hidden (unseen) classes C and gather training data in order to incrementally learn them. For example, a robot is built for greeting hotel guests. At any time, it has learned to recognize all existing guests. When it sees an existing guest, it greets and chats with him/her like a friend. When a new guest arrives, it should detect that it has never seen him/her before. It can say hello, asks for his/her name, take some pictures (training data), and then learn to recognize the guest. Next time when it sees the person, it can address the person by his/her name and chat like an old friend. This sounds easy! However, in general, this step is challenging as it is an unsupervised learning problem. The system does not know the number of classes in C or which instance belongs to which class. The next subsection is dedicated to this problem.
3. **Incrementally learn the unseen classes**: Assume that there are k new/unseen classes in C that have enough training data. The learner incrementally learns the k classes based on the gathered training data. That is, the existing model F_N is updated to produce a new model F_{N+k}. This is a *class continual learning* (CCL) problem.

   After learning in step 3, the system goes to step 1 and F_{N+k} becomes F_N. The process continues.

If learning and prediction is not the end task, the system will use the newly learned knowledge to help perform the end task, e.g., to generate a response to the user in a dialogue, or to generate a control action to control the car.

Steps 1 and 3 have been studied in *open-world learning* (Shu et al., 2017a, 2018; Chen and Liu, 2018) and *continual learning* (Chen and Liu, 2018; Parisi et al., 2018) respectively. Step 2 needs significant research, which we discuss next and propose a new method to do it.

Interactive Self-Supervision

Step 2 is a key challenge for on-the-job learning, i.e., how to find the hidden classes and obtain labeled training data. This must be done through actions initiated by the system itself without interrupting the application, as an intelligent agent should not rely solely on the user-provided training data to learn passively offline forever. It must learn actively on its own based on its prior knowledge by observing and interacting with the environment and humans to obtain explicit or implicit feedback to serve as supervision.

We propose an *interactive self-supervision* (ISS) framework for obtaining supervisory information via interactions with humans and the environment. It has three main forms:

- ***Trial-and-error*:** The system performs actions in the environment and observes the effects to gather training data. It is assumed the system knows right and wrong.
- ***Asking the user***: The system interacts with human users by *asking them questions* (see the two use cases below). Their answers can serve as the supervisory information.
- ***Reasoning and imitation***: The system uses its past knowledge and context to infer class labels through reasoning and imitation. For instance, in the guest greeting robot example, the robot should know that the pictures (training data) taken are all from that guest (class label). We will see how imitating others can help identify labels in the self-driving example later. Zero-shot learning (Palatucci et al. 2009) is also applicable here.

Note that asking human users is justified because it is evident that a large amount of our own knowledge is not learned by ourselves through our own observations or experiences but passed to us through teaching or other means. For example, most of us did not personally group vehicles into categories (or clusters) and called them buses, cars and trucks. We were told of these classes and their names or class labels by others. In a similar way, an intelligent agent should interact with and learn from humans who already possess a lot of knowledge. However, the interactions must be done in a natural and seamless way with little burden on the humans, unlike large-scale manual data labeling.

On-the-Job Learning during Conversations

Chatbots are now used in many applications, but they still have major weaknesses. One is that they cannot learn new knowledge during conversations, i.e., their knowledge is fixed beforehand. This is different from human conversations. We learn a great deal in our conversations. We either learn directly from the utterances of others, or by asking others questions. In this way, our knowledge grows over time and we become better and better at conversing.

There are many opportunities to learn new knowledge during a conversation. Here are a few examples.

Extract knowledge from user utterances. For example, when a user says "*I had a cheeseburger at McDonald's*," the chatbot can extract this piece of knowledge and save it in its knowledge base if it is unknown. When someone else asks "*Do you know whether McDonald's sell cheeseburger*?" The chatbot can easily answer *yes*.

Ask when it does not understand something: Mazumder et al. (2019a) proposed a method to do this in the context of building *natural language interfaces* (NLI). One of the key issues is how to understand paraphrased natural language (NL) commands from users in order to map a user command to a system's API call. Clearly, it is hard to exhaust all possible sentences that a user may utter for saying the same thing. The system must learn new paraphrased commands on the job when the user is using the NLI. This paper proposed a method to learn them when it has difficult to understand a user command via an interactive dialogue with the user. In this way, the system becomes more powerful. When the same or a similar NL command is given by this or another user, the system will have no problem to understand it.

Ask and infer: New knowledge may be inferred from the knowledge embedded in user utterances and the existing knowledge in the system's knowledge base.

For example, when someone asks us a question that we are unable to answer, we try to retrieve some relevant knowledge in our memory and reason over it before responding. We may also ask for some related information from the person whom we are conversing with. The acquired knowledge in the answers is then used to help the current reasoning. Mazumder et al. (2018; 2019b) proposed two methods to imitate this process when the user asks a *yes-or-no* question or a *WH-question* that the chatbot is unable answer. The system first formulates some questions to ask the user, whose answers are called *supporting facts*. Based on the supporting facts and the knowledge already in the chatbot's knowledge base, the system tries to infer the answer. Both the user-provided supporting facts and the chatbot inferred knowledge can be added to the chatbot's knowledge base to make the chatbot more knowledgeable and better able to answer user questions in the future.

On-the-Job Learning during Self-Driving

In its simplest form, self-driving involves *perception*, *risk assessment*, and *response generation*. Perception identifies what are on the road and in the surrounding area. Risk assessment predicts the danger level of each object based on its location and behavior, and where the danger may be. Response generation uses the prediction results to generate actions to control the car. On-the-job learning is *necessary* because it is very hard to exhaust all objects and all corner cases in offline training using manually labeled data.

We first discuss on-the-job learning for perception and risk assessment and then give a personal experience and a case for learning preferences. It is extremely hard to train a perception system purely based on offline manually labeled data because the real-world environment is highly

complex and unexpected situations occur all the time. Thus the system should learn during driving (on the job).

Identify unseen or unfamiliar objects (step 1). For example, when the car sees a black patch on the road that it has never seen before, it must first recognize this is an unseen/unexpected object.

Gather training data by taking pictures (step 2) of it for learning. Additionally, in order to predict whether it presents a danger for the car, the car needs to learn its behavior. But during driving, there is not enough time or data to learn the object behavior. The car should slow down and observe and possibly ask the human passenger for assistance. However, during driving, in many cases, one can imitate the behavior of the car before it, which indirectly provides some supervisory information. For example, if the other cars before it have driven through the patch with no issue, it can infer that the patch does not present a hazard (supervision). The car can just drive through it as well. In fact, the car can learn a great deal from the cars before it via imitation.

Note that in step 2, we also need to group unknown objects into classes. This can be difficult during driving as it may need to ask the human passenger. This is not an issue if the car is parked. It can ask for the names (class labels) of the objects by shown him/her the video. Note also even if the car is not driving, we still consider this on-the-job learning because the car has been sold and is out of the factory, and it is learning from its passengers, not through offline training initiated by its engineers.

Learn to recognize the unseen objects (step 3) incrementally based on the collected data (pictures). Learn to predict the risk of the objects using the data with the supervisory information gained via imitation above. Then, in the future, even when there is no other car in front, the car can still drive through without hesitation.

Based on the risk assessment, generate appropriate control actions, which is the end task. Following the example above, the car can just drive through normally.

A real experience: I worked on self-driving cars before. Once we were testing a car on the road. It stopped suddenly right in the middle of road and refused to move, but the road was wide open. We could not see anything that could be hazardous. After back to the lab, we watched the video, debugged the system, and found there was a small pebble on the road, which human eyes would not have noticed, but was caught by a sensor. If the system had a dialogue system and the on-the-job learning capability, we could have told the car to go ahead, and our instruction will serve as a form of supervision for the car to learn (on the job) so that it will not have problem next time in a similar situation.

Learning user preferences: A self-driving car manufacture cannot produce a car that suits everyone's preferences. For example, a safety conscious user may like the car to drive slowly. But we cannot ask the user to drive the car for a while to gather his/her driving habits and preferences and then learn from the data because the user may not know how to drive. Thus, the system has to learn from the feedback of the user in natural language. For example, when the car is driving very fast on a particular road, the user says "slow down." Based on such feedback, the car should learn to drive to best suit the user's preferences.

Challenges and Opportunities

In this section, we highlight some major challenges, which also present potential research opportunities. Their solutions can have fundamental impact on on-the-job learning and LL/CL in particular, and on ML and AI in general.

1. *Learning based on a few examples*. Using interactive self-supervision, the system is unlikely to collect a large number of training examples. Then few-shot learning (Lake et al. 2011) that can leverage previously learned knowledge will be critical.
2. *Interacting with the environment*: To do so effectively, the system has to make decisions about what is right and what is wrong through reasoning based on its existing knowledge. This is a challenging task.
3. *Imitation*: In the self-driving car example, we saw a case that the car imitates the cars in front it. However, it is challenging to decide what to and what not to imitate and how to imitate. Prior knowledge is again important.
4. *Disaster proof*: In order to learn on the job, the initial deployed model must have the ability to avoid disasters. For example, a self-driving car cannot afford to do any trial-and-error during driving if there is no guarantee that an action will not cause a catastrophe.
5. *Natural language dialogue*: To interact with humans, a good natural language interface is needed, which is hard to build. Mazumder et al. (2019b) proposed a new approach, which includes on-the job learning itself.
6. *Knowledge representation and reasoning*: All the above activities and LL need to leverage the knowledge learned in the past. Knowledge representation and reasoning are critical. So far, little research has been done about them (especially reasoning) in the LL context.
7. *Correctness of knowledge*: In order to use a piece of prior knowledge, the system must ensure the knowledge is correct. Incorrect knowledge is harmful. Some initial but limited work has been done in (Chen and Liu, 2014; Mitchell et al., 2015; Bou Ammar et al, 2015).
8. *Applicability of knowledge*: A piece of knowledge may be correct and useful in some previous domains, but not applicable or harmful to the new domain. There is still no general method for dealing with this issue. Some heuristics have been tried (Chen, Ma and Liu, 2015).

This list of challenges is by no means exhaustive. The current techniques are still primitive. Much research is needed to make breakthroughs. Since learning on the job needs rich prior knowledge, the system should also learn from other sources offline, e.g., Web text (Mitchell et al., 2015).

Acknowledgement: I think my former and current students: Jiahua Chen, Zhiyuan Chen, Sepideh Esmaeilpour, Geli Fei, Wenpeng Hu, Zixuan Ke, Gyuhak Kim, Huayi Li. Guangyi Lv, Nianzhu Ma, Sahisnu Mazumder, Arjun Mukherjee, Lei Shu, Hao Wang, Shuai Wang, and Hu Xu, for contributing many ideas about lifelong learning.

References

Bendale, A. and Boult, T. 2015. Towards Open World Recognition. In *Proceedings of CVPR*-2015.

Bou Ammar, H.; Eaton, E.; Luna, Jose Marcio, and Ruvolo, Paul. 2015. Autonomous Cross-domain Knowledge Transfer in Lifelong Policy Gradient Reinforcement Learning. In *Proceedings of AAAI-2015*.

Chen, Z. and Liu, B. 2014. Topic Modeling using Topics from Many Domains, Lifelong Learning and Big Data. In *Proceedings of ICML-2014*.

Chen, Z.; Ma, N. and Liu, B. 2015. Lifelong Learning for Sentiment Classification. In *Proceedings of ACL-2015*.

Chen, Z and Liu, B. 2018. *Lifelong Machine Learning*. Morgan & Claypool Publishers.

De Rosa, R.; Mensink, T. and Caputo, B. 2016. Online Open World Recognition. ArXiv:1604.02275.

Fei, G. and Liu, B. 2016. Breaking the Closed World Assumption in Text Classification. In *Proceedings of NAACL-HLT-2016*.

He, X. and Jaeger, H. 2018. Overcoming Catastrophic Interference Using Conceptor-aided Backpropagation. In *Proceedings of ICLR-2018*.

Hu, W.; Liu, Z.; Liu, B.; Tao, C.; Tao, Z.; Ma, J.; Zhao, D. and Yan, R.. 2019. Overcoming Catastrophic Forgetting for Continual Learning via Model Adaptation. In *Proceedings of ICLR-2019*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.; Milan, Kl; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming Catastrophic Forgetting in Neural Networks. In *Proceedings of the National Academy of Sciences*.

Lake, B.; Salakhutdinov, R.; Gross, J. and Tenenbaum J. 2011. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*. 33(33).

Lee, S.; Kim, J.; Jun, J.; Ha, J. and Zhang, B. 2017. Overcoming Catastrophic Forgetting by Incremental Moment Matching. In *Proceedings of NIPS-2017*.

Li, Z. and Hoiem, D. 2016. Learning without Forgetting. In *Proceedings of ECCV-2016*, 9908(1):614–629.

Liu, B. 2012. *Sentiment Analysis and Opinion Mining*. Morgan Claypool Publishers.

Lv. G.; Wang, S.; Liu, B.; Chen, E. and Zhang K. 2019. Sentiment Classification by Leveraging the Shared Knowledge from a Sequence of Domains. In *Proceedings of the 24th International Conference on Database Systems for Advanced Application*s (*DASFAA-2019*).

Masse, N. Y.; Grant, G. D. and Freedman, D. J. 2018. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. arXiv:1802.01569.

Mazumder, S.; Liu, B.; Wang, S. and Ma, N. 2019a. Lifelong and Interactive Learning of Factual Knowledge in Dialogues. In *Proceedings of SIGDIAL-2019*.

Mazumder, S.; Liu, B.; Wang, S. and Esmaeilpour, S. 2019b. Building an Application Independent Natural Language Interface. arXiv:1910.14084, 2019.

McCloskey, M. and Cohen, N. J. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation*, vol. 24, 109–165.

Mitchell, T.; Cohen, W.; Hruschka, E.; Talukdar, P.; Betteridge, J.; Carlson, A.; Dalvi, B.; Gardner, M.; Kisiel, B.; Krishnamurthy, J.; Lao, N.; Mazaitis, K.; Mohamed, T.; Nakashole, N.; Platanios, E.; Ritter, A.; Samadi, M.; Settles, B.; Wang, R.; Wijaya, D.; Gupta, A.; Chen, X.; Saparov, A.; Greaves, M.; and Welling, J.; 2015. Never-ending learning. In *Proceedings of AAAI-05*.

Palatucci, M.; Pomerleau, D.; Hinton, G. and Mitchell, T. 2009. Zero-shot learning with semantic output codes. In *Proceedings of 23rd Annual Conference on Neural Information Processing Systems* (*NIPS-2009*). 1410–1418.

Parisi, G. I; Kemker, R.; Part, J. L; Kanan, C. and Wermter, S. 2018. Continual Lifelong Learning with Neural Networks: A review. arXiv:1802.07569.

Rebuffi, S.; Kolesnikov, A.; Sperl, G. and Lampert, C. H. 2017. ICARL: Incremental Classifier and Representation Learning. In *Proceedings of CVPR-2017*.

Ruvolo, P. and Eaton, E. 2013. ELLA: An Efficient Lifelong Learning Algorithm. In *Proceedings of ICML-2013*, 507–515.

Scheirer, W. J.; de Rezende Rocha, A.; Sapkota, A. and Boult, T. E. 2013. Towards open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7), 1757–1772.

Schwarz, J.; Luketina, J.; Czarnecki, W. M; Grabska-Barwinska, A.; Teh, Y. W.; Pascanu, R. and Hadsell, R. 2018. Progress & Compress: A Scalable Framework for Continual Learning. In *Proceedings of ICML-2018*.

Seff, A.; Beatson, A.; Suo, D. and Liu, H. 2017. Continual Learning in Generative Adversarial Nets. arXiv:1705.08395.

Shin, H. Lee, J.; Kim, J. and Kim J. 2017 Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems*, 2990–2999.

Shu, L.; Xu, H. and Liu, B. 2017a. DOC: Deep Open Classification of Text Documents. In *Proceedings of EMNLP-2017*.

Shu, L.; Xu, H. and Liu, B. 2017b. Lifelong Learning CRF for Supervised Aspect Extraction. In *Proceedings of ACL-2017*.

Shu, L.; Xu, H. and Liu, B. 2018. Unseen Class Discovery in Open-world Classification. arXiv:1801.05609.

Silver, D. L.; Yang, Q. and Li, L. 2013. Lifelong Machine Learning Systems: Beyond Learning Algorithms. In *Proceedings of AAAI Spring Symposium on Lifelong Machine Learning*.

Thrun, S. 1996. Is learning the n-th thing any easier than learning the first? In *Proceedings of NIPS-1996*.

Thrun, S. and Mitchell, T. 1995. Lifelong Robot Learning. *In* L. Steels (ed.), *The Biology and Technology of Intelligent Autonomous Agents*, Springer-Verlag.

Xu, H.; Liu, B., Lei S. and Yu, P. 2019. Open-world Learning and Application to Product Classification. In *Proceedings of WWW-2019*.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; Hovy, E. 2016. Hierarchical Attention Networks for Document Classification. In *Proceedings of NAACL-HLT-2016*.

Yoon, J.; Yang, E.; Lee, J. and Hwang, S. 2018. Lifelong Learning with Dynamically Expandable Networks. In *Proceedings of ICLR-2018*.