Distributed Cooperative Deep Transfer Learning for Industrial Image Recognition

Benjamin Maschler[a*], Simon Kamm[a], Nasser Jazdi[a], Michael Weyrich[a]

[a]*Institut of Industrial Automation and Software Engineering, University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany*

* Corresponding author. Tel.: +49 711 685 67295; Fax: +49 711 685 67302. *E-mail address:* benjamin.maschler@ias.uni-stuttgart.de

Abstract

In this paper, a novel light-weight incremental class learning algorithm for live image recognition is presented. It features a dual memory architecture and is capable of learning formerly unknown classes as well as conducting its learning across multiple instances at multiple locations without storing any images. In addition to tests on the ImageNet dataset, a prototype based upon a Raspberry Pi and a webcam is used for further evaluation: The proposed algorithm successfully allows for the performant execution of image classification tasks while learning new classes at several sites simultaneously, thereby enabling its application to various industry use cases, e.g. predictive maintenance or self-optimization.

Keywords: Artificial Intelligence; Artificial Neural Networks; Continual Learning; Deep Learning; Distributed Learning; Dual Memory Method; Incremental Class Learning; Live Image Recognition; Transfer Learning

1. Introduction

The current trends in the manufacturing industries towards smaller, more customer-specific lots produced by interconnected, highly reconfigurable manufacturing systems just in time lead to a significant increase in these systems' complexity [1, 2, 3]. However, with a widespread availability of high-quality, high-resolution data [4] handling this complexity by artificial intelligence (AI) methods becomes feasible [3]. Yet, many of the approaches suggested in the last decade still require a great level of often manual adaption to the specific system's environment [5, 6, 7].

Deep learning based approaches offer a solution to this problem of automatic generalization [7, 8, 9], but require large amounts of training data [10, 11], which might not be available due to privacy or industrial espionage concerns, technical or legal reasons, and great computing power to conduct this training [3, 12]. These obstacles might be overcome by light-weight transfer learning approaches optimized for edge devices which enable the performant of deep learning algorithms on dispersed datasets during runtime without the so-called catastrophic forgetting [13, 14]. These approaches would then facilitate decentral AI solutions on the industrial shop floor which currently require machine learning on a central data lake, e.g. in predictive quality control [15], automatic control [5] or predictive maintenance [6]. However, as research in this field is still at the very beginning, the use-case considered here is of a more abstract, basic form, i.e. an image recognition task based on the ImageNet-dataset or a live camera-feed.

Objectives: In this paper, the challenge of cooperative distributed machine learning for industrial automation is derived (see *Sec. 2.1*) and literature regarding its solution in the field of image recognition surveyed (see *Sec. 2.2*). Thereon, a suitable approach based upon complementary learning systems (CLS) is selected and a specific implementation with a focus on low computing and storage requirements created (see *Sec. 3*). This implementation is then evaluated on a demanding standardized dataset and compared with published results (see *Sec. 4*). Finally, a prototype system is presented (see *Sec. 5*) before a conclusion with regards to the applicability of the approach in industrial use cases is drawn (see *Sec. 6*).

2. Related work

2.1 Transfer learning's challenges

'Natural' intelligence in humans and animals allows them to transfer knowledge from known problems and their solution towards unknown ones or to adapt previously learned lessons based upon new information. This is commonly called 'continual' learning [13].

In deep learning based AI, such transfer or adaption is not easily achieved, as new information tends to simply overwrite old information without gaining a considerable advantage compared to learning with new information from scratch. This displacement process is called '*catastrophic forgetting*' [16] and poses a significant obstacle for the construction of machine learning algorithms that are:

- *multi-purpose*, meaning the capability to successfully solve several different tasks learned sequentially,
- *multi-location*, meaning the capability to successfully train and infer on different locations parallel, or even just
- *highly adaptive*, meaning the capability to successfully adapt to changing inputs without major re-training,

Therefore, as in natural intelligence, the so-called '*stability-plasticity dilemma*' needs to be overcome. It refers to the opposing aims of having an algorithm stable enough to keep connections once learned and flexible enough to learn new connections once encountered. Accordingly, this form of machine learning is also termed 'continual' or sometimes 'transfer' learning.

There exists a large variety of different approaches to continual learning, surveyed e.g. as shown in [13] or [14]. However, most of those still dwell in the area of basic research, many only allowing for one or two of the three dimensions of transfer mentioned above.

2.2 Dual-memory method for image recognition

A promising approach to mitigate the stability-plasticity dilemma while allowing for all three dimensions of transfer is the *dual-memory method* [13, 17, 18]. It consist of two separate AI modules. A slow-learning module (inspired by the neocortex in a mammalian brain) and a fast-learning module (inspired by the hippocampus in a mammalian brain). The slow-learning module (*module A*) is used to extract general information from the input data and generalise the information while training. The fast-learning module (*module B*) remembers specific information of the input data and saves new memories. Fig. 1 depicts a possible setup for image recognition.

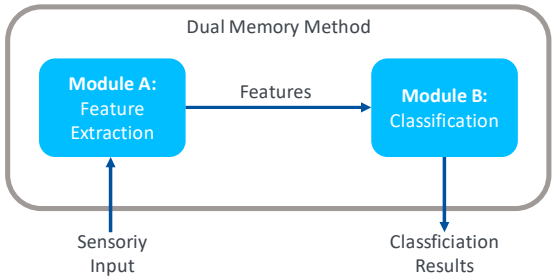

Figure 1: Dual Memory Method for Image Recognition

Extraction of general information from input data, i.e. feature extraction, is nowadays commonly carried out by deep neural networks (DNN). Depending on the application, different DNNs can be used for this purpose. Regarding the scenario presented in this paper, the following requirements need to be met: On the one hand, the algorithm for *module A* should extract good and relevant features from the input data. On the other hand, the DNNs should also run in real-time applications on (mobile) edge devices, e.g. on a Raspberry Pi. Therefore, the memory consumption and computational complexity of the DNN is a relevant criterion. Different feature extraction algorithms based upon a DNN architecture are compared based on their memory consumption, the number of Giga-FLOPs and their classification performance on the ImageNet-dataset in Table 1. Although those algorithms are usually full classification algorithms, only their feature extraction components are used further on. Therefore, they are referred to as 'feature extraction algorithms'.

Because of the focus on memory and computing power restricted devices, MobileNet-V2 is chosen for *module A*: It was designed with a focus on low memory consumption and operations per input data and still delivers acceptable classification – i.e. feature extraction – results [23].

However, while module A is just one of many possible DNN-based feature extractors, the requirements for *module B* are more demanding. In order to allow transfer learning as well as solve the use case as described above, an appropriate algorithm must

- be trainable on a data stream, where samples of the different classes appear randomly in time and order
- be able to classify already seen and trained classes at all times
- show a limited growth of computational complexity and memory consumption as the number of known classes grows and
- not need to store any training data.

While iCaRL [24] and FuzzyARTMAP [25, 26] fulfil the first three criteria, only the latter fulfils all four. Therefore,

Table 1: Comparison of feature extraction algorithms

DNN-Architecture	No. of Parameters x10^{-6}	Parameters' Memory-Consumption	No. of FLOPs x10^{-9}	Top-1 Classification Error	Top-5 Classification Error
AlexNet [19]	60	240 MB	0.7	36.7 %	15.3 %
VGG-16 [20]	138	552 MB	16	25.6 %	8.1 %
VGG-19 [20]	144	576 MB	20	25.5 %	8.0 %
ResNet-50 [21]	25.6	102 MB	4	20.7 %	5.3 %
ResNet-101 [22]	44.5	178 MB	8	19.9 %	4.6 %
Inception-V3 [22]	24	96 MB	4.8	21.6 %	5.6 %
MobileNet-V2 [23]	3.5	14 MB	0.3	28 %	9 %

FuzzyARTMAP was chosen as a foundation for *module B*. It's architecture allows it to solve the stability-plasticity-dilemma [25] by adding new knowledge without changing already trained information. The classification is then based on a comparison between the input data and the representations of known classes.

3. Methodology

As outlined in Sec. 2.1, we propose an architecture combining two different algorithms:

Module A uses the pre-trained MobileNet-V2 algorithm from Tensorflow 2.0 with a fix learning rate $\alpha_A = 0$. However, the last fully connected layers of MobileNet-V2 are removed in order to just extract features, which are then relayed to module B.

Module B is based on the FuzzyARTMAP algorithm, which is complemented by an updater that allows it to recognize completely new classes, too. Thereby, module B serves as an incremental, fast learning classifier.

The resulting classification process by this *distributed incremental class learning algorithm* (DICLA) based upon [27] is depicted in Fig. 2:

1. DICLA accepts input in form of images. The pre-trained feature extraction algorithm from *module A* reduces the image to a feature vector.
2. This feature vector is then relayed to *module B* where it is compared against a set of stored feature vectors called representations, which are mapped to the set of known classes: Each stored representation represents exactly one class while each class can be represented by one to many representations.
3. The *updater* now evaluates the result of the comparison process:
   A. If the similarity between the input feature vector and one of the representations passes a threshold value ρ then the input picture is classified by *module B* as belonging to the class associated with that representation.
   B. However, if this threshold is not passed, then the input image is considered to belong either to a class previously unknown or to add new information to a previously known class.
      i. In the former case, a new class is created and the input feature vector used as a representation of this class.
      ii. In the latter case, the input feature vector is used as a new representation of an already existing class.

   Thereby, both cases lead to an expansion of the algorithm's knowledge base during runtime. In both cases, *module B* outputs the class associated with that representation.

The number of representations associated with a class can be reduced by consolidating those into a single one. This is done by a separate process not depicted in Fig. 2.

4. Experiments

4.1 Experimental setup

The following experiments were executed on the ImageNet-dataset [28]. It features RGB-images of 224x224 pixels belonging to 1,000 different classes.

Due to time restrictions, for some experiments a subset of this dataset, the ImageNet-10 dataset was used. It features only 10 randomly drawn classes from the complete ImageNet-dataset. Additionally, its RGB-images consist of only 96x96 pixels. To allow for a comparison of different tests, the same 10 classes with the class indices of 145, 153, 289, 404, 405, 510, 805, 817, 867 and 950 were used throughout our experiments.

The training of the DICLA was executed with one epoch per incremental step, so every training image is seen just once. Based on a hyperparameter optimization, the following parameter were chosen:

- Training images per class: 100 (ImageNet-10)/ 10 (ImageNet) – randomly drawn
- Test images: 50 (all available images)
- $\rho = 0.5$ (fix threshold)

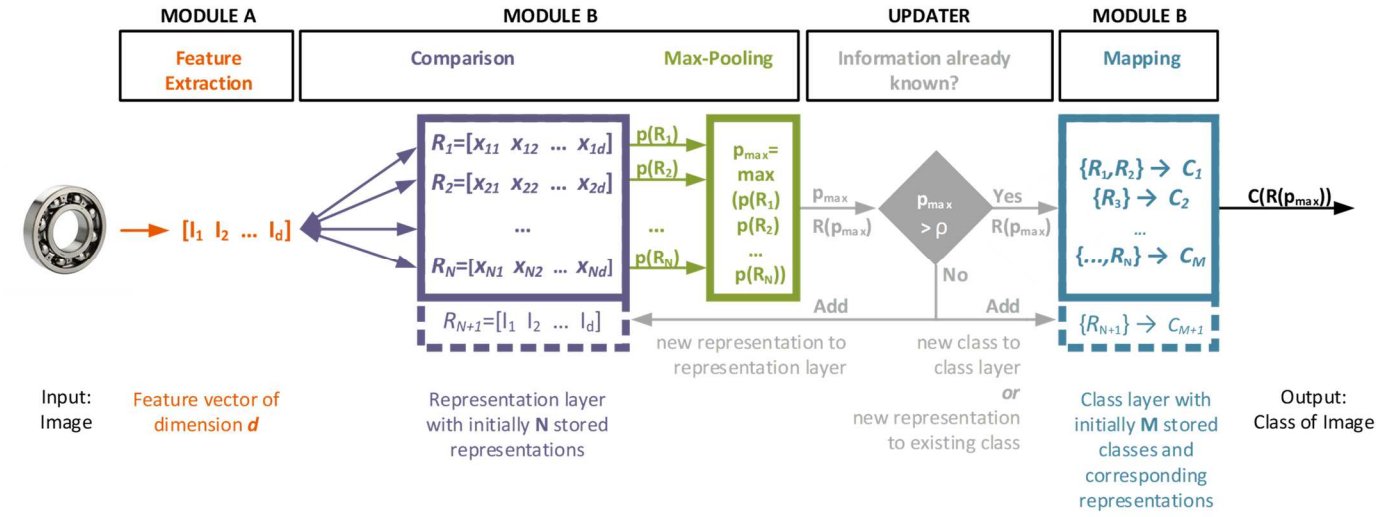

Figure 2: Structure of the Distributed Incremental Class Learning Algorithm (DICLA) according to [27]

- $\alpha_B = 0.2$ (learning rate of module B)

For an evaluation of the proposed algorithm different tests on ImageNet were performed. As a reference, iCaRL [24] and Learning without Forgetting (LwF) [29] are used. iCaRL and LwF use a ResNet-Architecture for feature extraction, which achieves a better classification accuracy than MobileNet-V2 (see Table 1). They perform 100 epochs per incremental step and use all training images (about 1,300 per class) of which iCaRL saves about 20,000. Results are obtained from [30].

4.2 Results: Incremental learning performance

The results for the three algorithms on the complete ImageNet dataset with 10 incremental training steps of 100 classes each are shown in Fig. 3:

In the beginning, there is a big performance gap between the better performing LwF and iCaRL on the one hand and DICLA on the other hand. However, while the number of classes increases this difference decreases, leading to DICLA finally performing about as good as LwF and only slightly worse than iCaRL. However, one has to keep in mind the considerable differences in computational complexity and storage required between iCaRL and DICLA. Furthermore, DICLA's accuracy is much more stable, decreasing by about 29 points as compared to 46 points (iCaRL) and 51 points (LwF).

As the other two algorithms are sensitive to the number of incremental steps [30], DICLA was tested with different numbers of incremental steps as well: In addition to the experiment described above, experiments with 1 and 20 steps were carried out. The results for 20 incremental steps are shown in Fig. 4. The curve is very similar to the curve with 10 incremental steps. Furthermore, the accuracy achieved with just 1 incremental step is about the same, too. Therefore, the algorithm does not seem to be sensitive to the number of incremental steps. The final classification results of LwF, iCaRL and the different set-ups of the DICLA are given in Table 2.

With the help of consolidation, the memory consumption and computational complexity of the DICLA can be further reduced. Two consolidation strategies and the impact of those were evaluated regarding classification accuracy and memory consumption. For these investigations, ImageNet-10 was used. One consolidation strategy consolidates all representations after every incremental step. The other strategy consolidates the representations only once after all incremental training steps have been carried out and just before calculating the final test accuracy. The mean accuracies with standard deviations over 10 runs can be seen Table 3 while Fig. 5 shows just the mean accuracies.

By consolidation, the memory consumption can be reduced drastically by about 90%. With the consolidation after every training step, the final classification accuracy (74.62%) is close

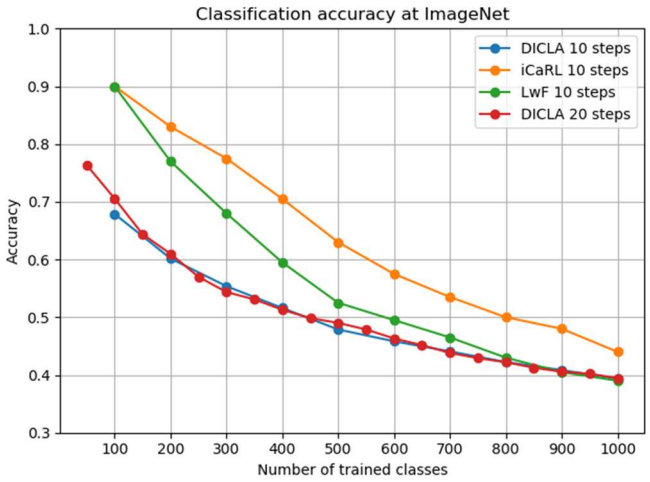

Figure 4: Results for ImageNet, 10 and 20 incremental steps

Table 2: Final Classification Accuracy on ImageNet

Algorithm	Final Classification Accuracy ImageNet	Final Memory Consumption ImageNet
DICLA (1 incremental step)	39.1 %	527.2 MB
DICLA (10 incremental steps)	39.3 %	530.5 MB
DICLA (20 incremental steps)	39.4 %	528.8 MB
iCaRL	44 %	2123 MB
LwF	39 %	-

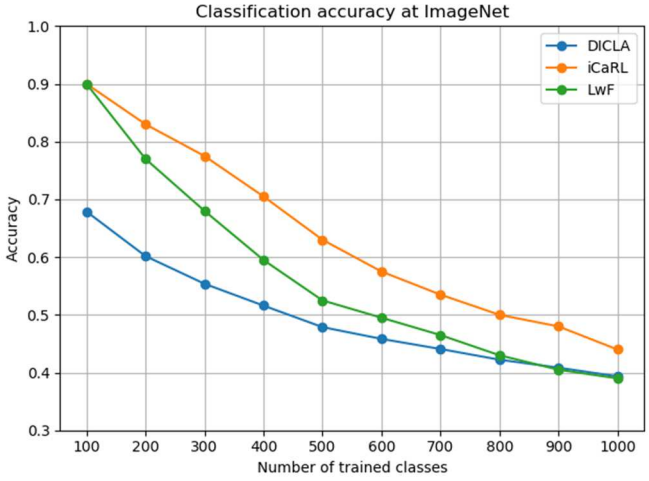

Figure 3: Results for ImageNet, 10 incremental steps

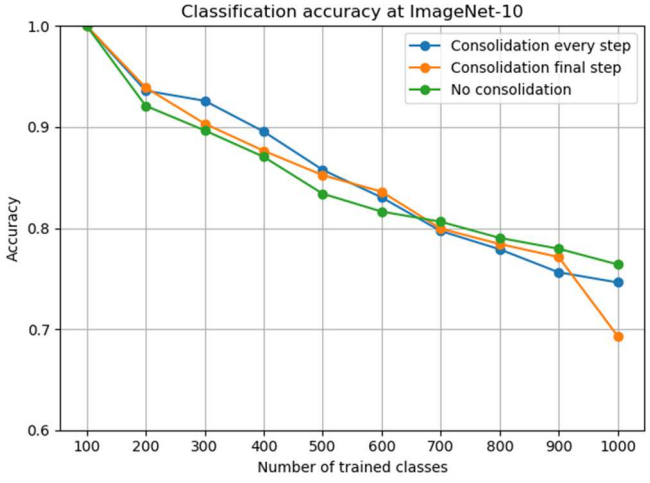

Figure 5: Impact of Consolidation on ImageNet-10, 10 incremental steps

Table 3: Final Classification Accuracy on ImageNet-10 by Consolidation Method

Consolidation Method	Final Classification Accuracy ImageNet-10	Final Memory Consumption ImageNet-10
No consolidation	76.40 (± 1.2) %	0.96 MB
Consolidation after every step	74.62 (± 2.19) %	0.10 MB
Consolidation after final step only	69.32 (± 4.81) %	0.10 MB

to the accuracy without consolidation (76.40%). With the consolidation after the final step only, the accuracy is significantly lower (69.32%) without saving any more memory space. Therefore, it can be said that with a suitable consolidation strategy, the memory consumption can be reduced significantly by only reducing the performance (here: classification accuracy) slightly. This makes this method highly useful for applications with little available memory.

4.3 Results: Distributed learning performance

To evaluate the performance of the algorithm for distributed scenarios, the ImageNet-10 dataset is used. The number of distributed edge devices is set to 2, 5 and 10, the integer factors of the number of classes. The 10 classes are uniformly and randomly distributed to the edge device, so that every device has the same number of classes to learn. After learning the local independent classes, the knowledge in every edge devices' module B is exchanged and the final accuracy on all classes is tested. For reference, the incremental class learning on ImageNet-10 with a single edge device (see Table 3) is used. The mean accuracies with standard deviations over 10 runs are given in Table 4.

Table 4: Distributed Learning Performance on ImageNet-10

Number of Edge Devices	Final Classification Accuracy ImageNet-10
1	76.40 (± 1.2) %
2	76.26 (± 1.5) %
5	73.36 (± 2.5) %
10	74.86 (± 2.1) %

It can be seen in the table that the performance is almost stable. The standard deviation of the final accuracy increases with a higher number of devices, but the final mean accuracy is not decreasing significantly when the training is distributed to more (5 or 10) devices and the knowledge combined only afterwards. Thus, the DICLA is well capable of training on local data and exchange the knowledge with other devices without sharing the training data.

The authors therefore believe that DICLA can be successfully applied to industrial use cases such as predictive quality control, self-optimization or predictive maintenance, allowing for a cooperation in learning without sharing one's input data.

5. Prototype

In order to demonstrate DICLA's features in a less abstract use case and on an actual edge device, a prototype setup was created (see Fig. 6):

A Raspberry Pi Camera Module v2 periodically captures images of a white placement area and the objects thereon. The camera is connected to a standard-issue, non-overclock Raspberry Pi 3 Model B+, on which an instance of DICLA is analyzing the images captured. The Raspberry Pi is further connected to a screen, on which the current image, the class of this image according to DICLA as well as DICLA's certainty regarding this assessment is displayed. A user can correct DICLA's classification via keyboard and mouse by entering a different already existing or new class name, causing the algorithm to create new representations or classes respectively. Furthermore, the user can trigger a consolidation.

The prototype setup proved to be capable of handling an image stream of more than one 96x96 image per second while correctly learning and recognizing new classes of every day office and lab equipment like pencils, screwdrivers or pens without overheating or lagging behind. A distribution of the learning task was possible without a deterioration of performance as could be expected according to Section 4.3.

6. Conclusion and transfer

In this paper, an algorithm for image recognition based on the dual-memory-method and capable of transfer learning, i.e. cooperatively and performant learning on distributed, evolving datasets possibly describing different tasks without forgetting previously acquired knowledge, is presented.

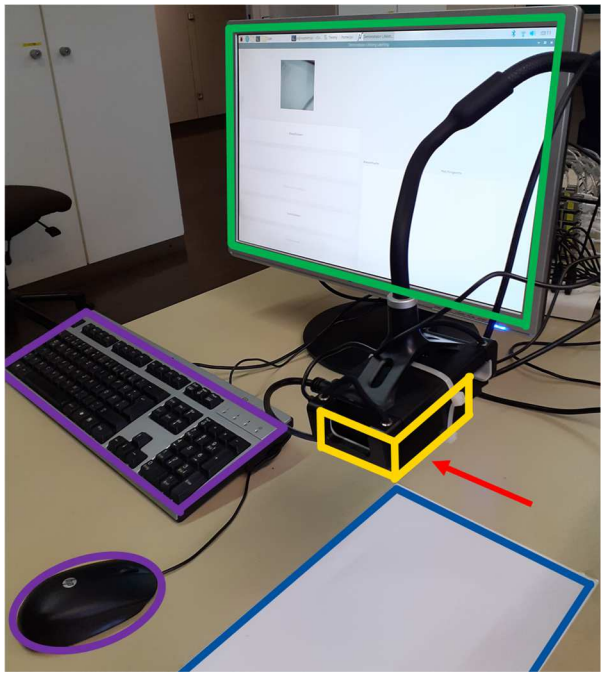

Figure 6: Prototype Setup with Placement Area (blue), Raspberry Pi (yellow), Camera (location indicated by red arrow), User Input Devices (violet), User Output Device (green)

A special focus is laid on this algorithm's industrial applicability, calling for low requirements regarding computational power and storage volume in order to allow its use in edge devices on the industrial shop floor.

The proposed distributed incremental class learning algorithm (DICLA) was implemented and compared on the ImageNet benchmark dataset with two state-of-the-art transfer learning algorithms, i.e. LwF and iCaRL. It could be demonstrated, that despite its slightly lower accuracy compared to iCaRL, it is more performant taking the considerably lower hardware consumption into account. Furthermore, it is – other than LwF and iCaRL – not sensitive to the number of training steps and does not suffer losses from distributing the learning to different devices and combining the acquired knowledge only in the end.

To underline these findings, a prototype setup based upon a standard Raspberry Pi 3 and a camera was created. The recognition of everyday objects could be learned from an image stream online and without deterioration even if different devices were learning different objects separately.

The capabilities demonstrated thereby support the assumption that DICLA is a good foundation for solving industrial use cases: By expanding their respective deep learning algorithms to incorporate the presented approach, they, too, could learn on light-weight edge devices. Naturally, this is easier for image recognition based use cases, but the authors are currently working on applying their approach to time series data as well. Thereby, a broad range of use cases from predictive quality control over self-optimization to predictive maintenance could be realized on distributed edge devices without sharing the input data. The authors happily invite other researchers to examine this potential on a broader scale and in a real-life scenario.

References

[1] Efthymiou K, Pagoropoulos A, Papakostas N, Mourtzis D, Chryssolouris G. Manufacturing Systems Complexity Review: Challenges and Outlook. Procedia CIRP 2012;3:644–9.

[2] Bruzzone AAG, D'Addona DM. New Perspectives in Manufacturing: An Assessment for an Advanced Reconfigurable Machining System. Procedia CIRP 2018;67:552–7.

[3] Meister M, Beßle J, Cviko A, Böing T, Metternich J. Manufacturing Analytics for problem-solving processes in production. Procedia CIRP 2019;81:1–6.

[4] Kagermann H. Change Through Digitization—Value Creation in the Age of Industry 4.0. In: Albach H, Meffert H, Pinkwart A, Reichwald R, editors. Management of Permanent Change. Wiesbaden: Springer Fachmedien Wiesbaden; 2015, p. 23–45.

[5] Lindemann B, Karadogan C, Jazdi N, Liewald M, Weyrich M. Cloud-based Control Approach in Discrete Manufacturing Using a Self-Learning Architecture. IFAC-PapersOnLine 2018;51(10):163–8.

[6] Lughofer E, Sayed-Mouchaweh M. Prologue: Predictive Maintenance in Dynamic Systems. In: Lughofer E, Sayed-Mouchaweh M, editors. Predictive Maintenance in Dynamic Systems. Cham: Springer International Publishing; 2019, p. 1–23.

[7] Yao X, Zhou J, Zhang J, Boer CR. From Intelligent Manufacturing to Smart Manufacturing for Industry 4.0 Driven by Next Generation Artificial Intelligence and Further On. In: 2017 5th International Conference on Enterprise Systems (ES). IEEE; 2017, p. 311–318.

[8] Kozjek D, Vrabič R, Kralj D, Butala P. A Data-Driven Holistic Approach to Fault Prognostics in a Cyclic Manufacturing Process. Procedia CIRP 2017;63:664–9.

[9] Yang R, Huang M, Lu Q, Zhong M. Rotating Machinery Fault Diagnosis Using Long-short-term Memory Recurrent Neural Network. IFAC-PapersOnLine 2018;51(24):228–32.

[10] Wang J, Ma Y, Zhang L, Gao RX, Wu D. Deep learning for smart manufacturing: Methods and applications. J Manu Sys 2018;48:144–56.

[11] Jordan MI, Mitchell TM. Machine learning: Trends, perspectives, and prospects. Science 2015;349(6245):255–60.

[12] Maschler B, Jazdi N, Weyrich M. Maschinelles Lernen für intelligente Automatisierungssysteme mit dezentraler Datenhaltung am Anwendungsfall Predictive Maintenance. In: VDI-Kongress Automation 2019, Baden-Baden, Germany. VDI 2019;2351:739–751.

[13] Parisi GI, Kemker R, Part JL, Kanan C, Wermter S. Continual lifelong learning with neural networks: A review. Neural Netw 2019;113:54–71.

[14] Maltoni D, Lomonaco V. Continuous learning in single-incremental-task scenarios. Neural Netw 2019;116:56–73.

[15] Lindemann B, Fesenmayr F, Jazdi N, Weyrich M. Anomaly detection in discrete manufacturing using self-learning approaches. Procedia CIRP 2019;79:313–8.

[16] French R. Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences 1999;3(4):128–35.

[17] Hinton GE, Plaut DC. Using fast weights to deblur old memories. In: 1987 CSS 9th Annual Conference of the Cognitive Science Society (CogSci). CSS; 1987, p. 177-186.

[18] Kemker R, McClure M, Abitino A, Hayes T, Kanan C. Measuring Catastrophic Forgetting in Neural Networks. In: 2018 AAAI 32nd Conference on Artificial Intelligence. AAAI; 2018, p. 3390-3398.

[19] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun. ACM 2017;60(6):84–90.

[20] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition; 2014.

[21] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2016, p. 770–778.

[22] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2016, p. 2818–2826.

[23] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2018, p. 4510–4520.

[24] Rebuffi S-A, Kolesnikov A, Sperl G, Lampert CH. iCaRL: Incremental Classifier and Representation Learning. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2017, p. 5533–5542.

[25] Carpenter GA, Grossberg S. Adaptive Resonance Theory. Springer US; 2010, p-22-35.

[26] Merten AM. Adaptive Resonance Theory [ART] – Ein neuer Ansatz lernender Computer. University of Ulm, 2003 (Online: http://www.informatik.uni-ulm.de/ni/Lehre/WS03/ProSemNN/ART.pdf, last accessed on 2020-01-09).

[27] Maschler B, Weyrich M. Deep Transfer Learning at Runtime for Image Recognition in Industrial Automation Systems. 16th Technical Conference EKA – Design of Complex Automation Systems, Magdeburg, 2020.

[28] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S et al. ImageNet Large Scale Visual Recognition Challenge. Int J Comput Vis 2015;115(3):211–52.

[29] Li Z, Hoiem D. Learning without Forgetting. IEEE Trans Pattern Anal Mach Intell 2018;40(12):2935–47.

[30] Wu Y, Chen Y, Wang L, Ye Y, Liu Z, Guo Y et al. Large Scale Incremental Learning. In: 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2019.