Methods

Benjamin Maschler*, Simon Kamm and Michael Weyrich

Deep industrial transfer learning at runtime for image recognition

Deep Industrial Transfer Learning für Bilderkennung zur Laufzeit

https://doi.org/10.1515/auto-2020-0119
Received July 15, 2020; accepted December 18, 2020

Abstract: The utilization of deep learning in the field of industrial automation is hindered by two factors: The amount and diversity of training data needed as well as the need to continuously retrain as the use case changes over time. Both problems can be addressed by industrial deep transfer learning allowing for the performant, continuous and potentially distributed training on small, dispersed datasets. As a specific example, a dual memory algorithm for computer vision problems is developed and evaluated. It shows the potential for state-of-the-art performance while being trained only on fractions of the complete ImageNet dataset at multiple locations at once.

Keywords: continual learning, deep learning, incremental class learning, live image recognition, transfer learning

Zusammenfassung: Die Nutzung von Deep Learning auf dem Gebiet der Industrie-Automatisierung wird durch zwei Faktoren behindert: Die benötigte Menge und Vielfalt von Trainingsdaten sowie die Notwendigkeit, fortlaufend nachzutrainieren, wenn sich der Anwendungsfall im Laufe der Zeit wandelt. Beide Probleme können durch Industrial Deep Transfer Learning gelöst werden: Es ermöglicht performantes, kontinuierliches und bedarfsabhängig verteiltes Lernen auf kleinen, verstreuten Datensätzen. Als konkretes Beispiel wird ein Dual-Memory-Algorithmus für Computer-Vision-Probleme entwickelt und evaluiert. Er zeigt das Potenzial für state-of-the-art Performanz, während er jeweils auf Ausschnitten des kompletten ImageNet-Datensatzes an verschiedenen Standorten gleichzeitig trainiert wird.

***Corresponding author: Benjamin Maschler,** Institute of Industrial Automation and Software Engineering, University of Stuttgart, Pfaffenwaldring 47, 70550 Stuttgart, Germany, e-mail: benjamin.maschler@ias.uni-stuttgart.de
Simon Kamm, Michael Weyrich, Institute of Industrial Automation and Software Engineering, University of Stuttgart, Pfaffenwaldring 47, 70550 Stuttgart, Germany, e-mails: simon.kamm@ias.uni-stuttgart.de, michael.weyrich@ias.uni-stuttgart.de

Schlagwörter: kontinuierliches Lernen, deep learning, inkrementelles Klassenlernen, Bilderkennung zur Laufzeit, Transferlernen

1 Introduction

The ongoing global trend towards connected industry, labeled e. g., Industry 4.0 or Industrial Internet of Things, has created a world of cyber-physical production systems providing a wealth of industrial data and interfaces [8, 19]. However, together with this data and accessibility came a substantial increase in the systems' complexity as well, requiring automatic solutions to problems such as control and monitoring tasks that still reflect each scenario's specific characteristics.

Machine learning, especially in the form of deep learning, promises such automatic, data-driven adaption to specific use case scenarios [30, 10]. However, although the number of publications utilizing such approaches in industrial applications is sharply on the rise [18], substantial practical problems hinder the practical use of these solutions:

- Deep learning requires training datasets diverse and large enough to include all relevant aspects of a given problem [4, 27]. In practice, acquiring such datasets poses a great challenge as data is oftentimes not shared due to technical, legal or securital concerns and cannot easily be collected by a single entity due to quantitative and qualitative requirements [14, 29, 26].
- Training deep learning algorithms is computationally intensive whereas using a trained algorithm does not require much computing power. However, most industrial use cases are somewhat dynamic, requiring the continuous gathering of data and, thus, retraining of the respective algorithm. In practice, this poses a challenge as an algorithm's user therefore needs to have the infrastructure not only to use, but to retrain the algorithm continuously [26, 31, 17].

Table 1: Notation.

Symbol	Definition	Symbol	Definition
D	Dataset	i	Instance counter
$P(\,)$	Distribution	n	Number of instances
X	Instance set	t	Number of tasks
$\mathcal{D}$	Domain	x	Feature vector
$\mathcal{T}$	Task	y	Label
$\mathcal{X}$	Feature space	α	Learning rate
$\mathcal{Y}$	Label space	ρ	Threshold value

These two problems could be solved by deep learning algorithms allowing cooperative, distributed and continuous training on small datasets facilitating a transfer of previously acquired knowledge. To this end, methods from the fields of transfer learning and continual learning can be utilized [17]. To examine their potential for industrial applications, in this paper an image recognition task, representing e. g., a vision-based quality control problem, is solved using a continual learning approach trained on small, distributed datasets.

In this article, concepts regarding the transfer of knowledge in industrial automation are introduced and a suitable approach selected in Section 2. The methodology used is then thoroughly described in Section 3 and evaluated experimentally in Section 4. Section 5 gives a short conclusion and presents an outlook on the next steps necessary to enhance the use of machine learning in industrial automation.

2 Related work

In this chapter, first, an overview of the transfer of knowledge in machine learning is given and from there the term 'deep industrial transfer learning' defined. Secondly, a general approach for a vision-based deep industrial transfer learning scenario is derived from literature.

For convenience, a list of symbols and their definitions as used throughout this paper is given in Tab. 1.

2.1 Deep industrial transfer learning

In humans and animals, 'natural' intelligence allows the transfer of knowledge from known problems and their solution towards unknown ones or to adapt previously learned lessons based upon new information – both while retaining older information or skills. This is commonly called '*continual learning*' [13, 21].

In 'artificial' deep learning, such transfer or adaption is not easily achieved, as new information simply tends to overwrite old information without gaining a considerable advantage compared to learning from scratch using only new information. This displacement process is called '*catastrophic forgetting*' [3].

The term '*continual learning*' describes approaches to overcome the problem of catastrophic forgetting in machine learning. More specifically, it describes a field focused on solving multi-task problems in a common feature space $\mathcal{X}_1 = \cdots = \mathcal{X}_t$ with differing distributions of input data $P(X_1) \neq \cdots \neq P(X_t)$ and known source and target labels. The goal is to learn those tasks sequentially without forgetting previously learned tasks, so that eventually all tasks can be solved by a single deep learning algorithm. Building upon [6, 20], three different problem cases can be differentiated:

- Incremental Task Learning (ITL) describes problems, in which a deep learning algorithm is sequentially trained on multiple observations $D_t = \{(x, y) \mid x_i \in \mathcal{X};\ y_i \in \mathcal{Y}_t;\ i = 1, \ldots, n_t\}$ corresponding to $t \in \mathbb{N}^+$ domains and tasks in order to be able to infer any $y_j \in \mathcal{Y}_t$ given $(x_j \in \mathcal{X}, t)$. This is commonly achieved using a multi-headed output layer of which only the head corresponding to task $\mathcal{T}_t$ is active.
- Incremental Domain Learning (IDL) describes problems, in which a deep learning algorithm is sequentially trained on multiple observations $D_t = \{(x, y) \mid x_i \in \mathcal{X};\ y_i \in \mathcal{Y}_t;\ i = 1, \ldots, n_t\}$ corresponding to $t \in \mathbb{N}^+$ domains and tasks in order to be able to infer any $y_j \in \mathcal{Y}$ given $x_j \in \mathcal{X}$.
- Incremental Class Learning (ICL) describes problems, in which a deep learning algorithm is sequentially trained on multiple observations $D_t = \{(x, y) \mid x_i \in \mathcal{X};\ y_i \in \mathcal{Y}_t;\ i = 1, \ldots, n_t\}$ corresponding to $t \in \mathbb{N}^+$ domains and tasks in order to be able to infer any $y_j \in \mathcal{Y}_t$ given $x_j \in \mathcal{X}$.

Fig. 1 illustrates these continual learning problem classes using the example of hand-written digit recognition: An algorithm is sequentially trained to label images of handwritten digits '1' and '2' (representing labels from the label space $\mathcal{Y}_1$) in task $\mathcal{T}_1$ and '3' and '4' (representing labels from the label space $\mathcal{Y}_2$) in task $\mathcal{T}_2$.

- ITL describes this algorithm's ability to correctly label new images when given information as to which label space $\mathcal{Y}_t$ shall be used – i. e., returning e. g., one of the two possible answers '1' and '2' for an input image associated with task $\mathcal{T}_1$.
- IDL, in contrast, describes the algorithm's ability to correctly label new images without specifying the label space $\mathcal{Y}_t$ used at all – i. e., returning one of the merely two possible answers '1 or 3' and '2 or 4'.

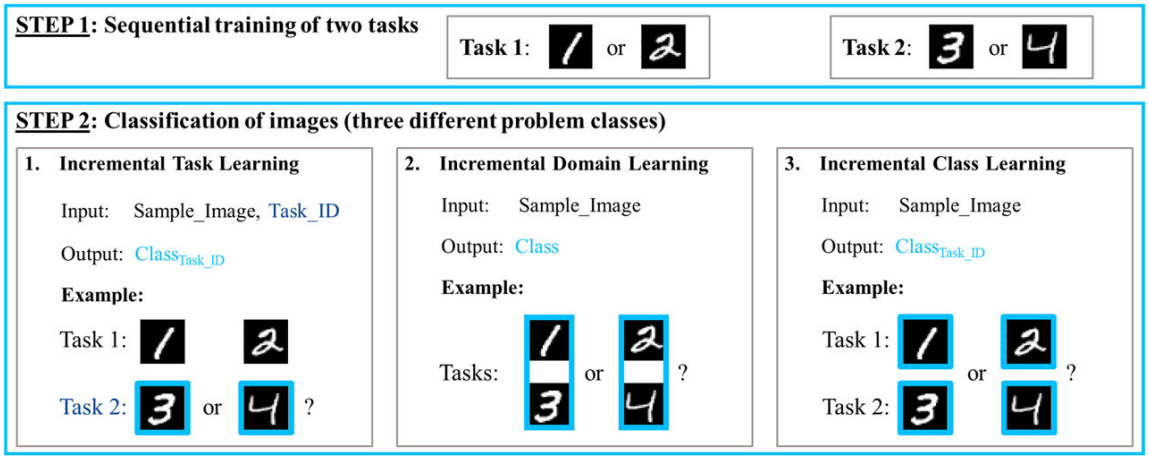

Figure 1: Continual learning problem classes using the example of hand-written digit recognition.

– ICL, finally, describes the algorithm's ability to correctly label new images using all labels from all label spaces $\mathcal{Y}_t$ – i. e., returning one of the four possible answers '1', '2', '3' and '4'.

According to [20], ICL is the most difficult, whereas ITL is the easiest of those continual learning problem classes. Numerous approaches to solve them have been published, of which [21] and [13] present recent surveys.

However, these continual learning problem classes only address a sub-space of the much greater problem space regarding the transfer of knowledge. More precisely, continual learning focusses on retaining old skills and thereby leads to an algorithm's generalization, focusing on similarities between different (sub-)problems, e. g., recognizing characteristics of the same problem described in different datasets. Contrastingly, '*transfer learning*' focusses just on the new skills [22] and thereby leads to an algorithm's differentiation, focusing on differences between different (sub-)problems, e. g., forming clusters of very similar problems described in different datasets. As useful such a distinction might be in classifying algorithms, it loses its usefulness when applied to practical problems: Industrial automation use cases as described in Section 1 usually require the generalization of information contained across potentially dispersed sub-datasets, e. g., at different locations, and across small changes over time as well as the differentiation between different variants of a problem. This double requirement leads the authors of this article to define the term '*industrial transfer learning*' as encompassing both, continual and transfer learning. More specifically, *deep industrial transfer learning* shall be examined as a method to mitigate the problems of conventional deep learning as described in Section 1 [17].

Luckily, both, continual learning and transfer learning, need to overcome the so-called 'stability-plasticity dilemma', which 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. This justifies the hypothesis, that at least some continual learning approaches should deliver good results when used for industrial transfer learning.

2.2 Dual memory method for image recognition

One such promising continual learning approach suitable for deep industrial transfer learning as described above is the dual-memory method [21, 9, 15]: A slow-learning module (inspired by a mammalian brain's neocortex section) and a fast-learning module (inspired by a mammalian brain's hippocampus section) combine their strengths to mitigate the stability-plasticity dilemma. The slow-learning module, hereon called module A, is used to extract general information from the input data and generalize it. In contrast, the fast-learning module, hereon called module B, is used to extract specific information of the input data and to save it as new memory.

This approach is not to be confused with the state-of-the-art process of using pre-trained convolutional neural networks to solve image recognition tasks. Instead of merely pre-training on a similar dataset, which would indeed solve the problem of training data availability, the proposed approach continuously and efficiently re-trains on every new instance provided and therefore solves the problem of continuously changing dynamic processes as well.

Table 2: Deep learning based image recognition algorithms from literature.

Feature-Extraction-Algorithm	No. of Para-meters	Memory Consumption	Top-1 Classification Accuracy	Top-5 Classification Accuracy
MobileNet-V2 [24]	3.5×10^6	14 MB	71.3 %	90.1 %
DenseNet121 [7]	8.1×10^6	33 MB	75.0 %	92.3 %
Xception [2]	22.9×10^6	88 MB	79.0 %	94.5 %
Inception-V3 [25]	23.9×10^6	92 MB	77.9 %	93.7 %
ResNet-50V2 [5]	25.6×10^6	98 MB	76.0 %	93.0 %
ResNet-152V2 [5]	60.4×10^6	232 MB	78.0 %	94.2 %

Today, feature extraction, i. e., the extraction of general information from input data, is usually carried out by deep neural networks (DNN). Different DNNs can be used for this purpose, but for image recognition tasks as examined here, convolutional neural networks (CNN) have proven most capable. Tab. 2 lists some of the most accurate CNNs, whose performance shall be compared in Section 2. Although the algorithms listed are full classification algorithms, only their feature extraction components will be used. Therefore, they are referred to as feature extraction algorithms.

While many CNN-based feature extractors can be used as module A, the requirements for module B are more demanding. In order to facilitate transfer learning for the use case described in Section 1, 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 raw data for training.

The first three criteria are met by Incremental Classifier and Representation Learning (iCaRL) [23] and FuzzyARTMAP [1, 12], while only FuzzyARTMAP fulfils all four. Therefore, FuzzyARTMAP was chosen as a foundation for module B. Its architecture allows it to solve the stability-plasticity-dilemma [1] 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 Section 2.2, we propose an architecture combining two different algorithms:

- Module A uses different, pre-trained image recognition CNNs from Tensorflow 2.0 with a fix learning rate $\alpha_A = 0$. However, for each one of those, the last fully connected layer is removed in order to just extract features, which are then relayed to module B.
- Module B is based on the FuzzyARTMAP algorithm enhanced by an updating mechanism allowing it to recognize completely new classes, too. Thus, module B serves as the aforementioned incremental, fast learning classifier.

The resulting classification process by this distributed incremental class learning algorithm (DICLA) is depicted in Fig. 2:

1. DICLA is presented with images $\underline{x}$. The pre-trained feature extraction algorithm f_A from module A reduces those to a feature vector $f_A(\underline{x}) = \underline{I}$.
2. This feature vector $\underline{I}$ is then relayed to module B where it is compared against a set of N stored feature vectors called representations $\underline{R}_n$ ($n \in [1, N]$) by computing the pair-wise cosine similarity $p(\underline{R}_n)$ as defined in equation (1).

$$p(\underline{R}_n) = \frac{\underline{I} * \underline{R}_n}{\|\underline{I}\|_2 \left\|\underline{R}_n\right\|_2}. \tag{1}$$

   These representations are mapped to the set of known classes C_m ($m \in [1, M]$): Each stored representation represents exactly one class while each class can be represented by one to many representations.
   Such stored representations do not violate the requirement not to store raw data for training, because those representations are to be considered as abstract representations of knowledge, i. e., learning results, rather than raw data.
3. The updater now evaluates the result of the comparison process by comparing it with a threshold value ρ. This threshold value can either be fix or be adaptive. An adaptive threshold $\rho_{adaptive}$ is proposed in equation (2). It shall ensure a "clear" winner by factorizing the mean of all similarity values with an additional

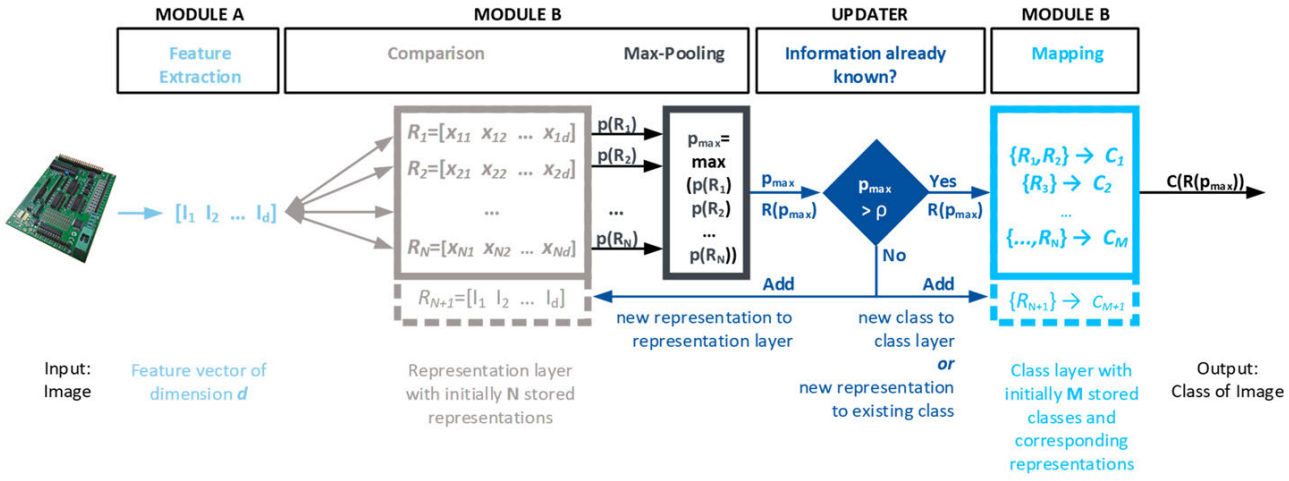

Figure 2: Structure of the Distributed Incremental Class Learning Algorithm (DICLA) based upon [16].

thresholding parameter s, which typically is slightly larger than 1, e. g., $s = 1.1$. With this parameter, the desired unambiguity of the winner representation is adjusted.

$$\rho_{adaptive} = s * \frac{1}{N} \sum_{i=1}^{N} p(\underline{R}_i). \tag{2}$$

a) If the similarity between the input feature vector and one of the representations passes the threshold value, then this input picture is classified by module B as belonging to the class associated with the representation the pictures' feature vector matched. When the algorithm is in training mode, the winning representation $\underline{R}_n$ is adapted ("trained") with the information obtained by the new input feature vector $\underline{I}$ and according to learning rate α_B as given in equation (3).

$$\underline{R}_{n\ adapted} = (1 - \alpha_B)\,\underline{R}_n + \alpha_B \underline{I}. \tag{3}$$

b) However, if the threshold value is not passed, then the input image is considered to manifest a new representation. This representation is then manually assigned to a class.
   i. If the representation belongs to a previously unknown class, a new class is created and the input feature vector used as a representation of this class.
   ii. If the representation adds new information to a previously known class, the input feature vector is used as a new representation of this already existing class.

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

The number of representations associated with a class can be reduced by consolidation, a form of flattening. Consolidation is not depicted in Fig. 2.

4 Experiments

In the following section, first, the influence of different feature extraction algorithms on DICLA's incremental learning performance shall be examined. The resulting, optimal version of DICLA is then compared to state-of-the-art incremental learning algorithms. Finally, this version of DICLA is evaluated regarding its distributing learning capabilities.

4.1 Incremental learning: comparison of different feature extractors on ImageNet-10

The following experiments compare the different feature extraction algorithms from Tab. 2 regarding their performance when used in module A. The experiments were executed on the ImageNet-10-dataset, a subset of the larger ImageNet-dataset. It can be characterized by a small number of different classes (10) and a large number of highly different natural images per class (>1250), challenging any image recognition algorithm to distinguish small as well as large differences.

Table 3: Comparison of the impact of the number of training images per class for every feature extraction algorithm from Tab. 2 regarding training time and final accuracy (double standard deviation indicated) averaged over five runs.

		No. of Training Images per Class								
		1	10	50	100	250	500	750	1000	1250
Mobile-Net-V2	Avg. Training Time [s]	12.9	13.8	23.4	34.3	74.4	144.7	239.4	331.8	439.7
	Avg. Final Accuracy [%]	47.8±3.8	67.6±1.3	75.4±2.8	75.4±3.9	75.7±2.5	74.5±2.4	76.3±2.1	75.9±2.1	75.4±2.7
DenseNet121	Avg. Training Time [s]	28.5	30.8	51.8	79.2	168.9	334.5	514.2	729.8	1017.9
	Avg. Final Accuracy [%]	37.1±5.2	56.5±3.5	69.4±2	69.7±2.9	71.2±2.2	73.2±3.4	72±3.7	72.3±1.1	72.9±2.1
Xception	Avg. Training Time [s]	14.7	18.5	40.7	66.7	164.1	393.1	700.1	1051.7	1647.6
	Avg. Final Accuracy [%]	34.9±5.9	55.2±3	63.6±1.9	64±1.8	63.6±0.8	67±2.8	65.8±0.8	65.7±1.9	66.3±3.1
Inception-V3	Avg. Training Time [s]	26.0	30.1	60.8	96.8	223.7	493.4	826.6	1344.4	2119.8
	Avg. Final Accuracy [%]	23.8±0.8	39.6±3.1	49.5±2.1	52.5±3.1	55.3±1.7	56.8±2.4	57±3.5	56.7±2.6	58±2.4
ResNet-50V2	Avg. Training Time [s]	17.7	22.0	47.4	80.3	197.1	442.9	779.7	1172.2	1679.5
	Avg. Final Accuracy [%]	34.9±5.3	54.9±3.5	62.4±1.4	65.7±1.8	65.2±2.3	66.7±2.9	65.4±3.8	67±5.8	66.3±2.2
ResNet-152V2	Avg. Training Time [s]	43.4	49.4	98.0	155.9	360.7	778.4	1343.4	2130.8	2888.5
	Avg. Final Accuracy [%]	34.9±4.2	54.2±3.1	61.7±2	65.1±3.1	68.1±1	67.3±2.4	69.6±2.6	66.9±1	67.6±2.4

The training of DICLA was executed with one epoch per incremental step, so that every training image is seen just once. As described in Section 3, DICLA contains different hyperparameters, which were optimized based on a grid search. Because the feature extractor, module A, was fixed for these experiments ($\alpha_A = 0$), solely hyperparameters of module B, threshold value ρ and the learning rate α_B, were optimized. Both parameters can range from 0 to 1 and were investigated in this range with a step width of 0.1, resulting in 121 combinations for the grid search. To ensure meaningful results, we performed cross validation. We split the training data (<1250 images) into a training set of 20 and a validation set of 100 randomly drawn images per execution. This was repeated five times per possible combination, resulting in a total of 606 tests. Based on this hyperparameter optimization, the following parameter were chosen for conducting the final results:

- Training images per class: 10 to 1250 – randomly drawn
- Test images per class: 50 (all available images)
- Threshold value $\rho = 0.5$ (fix threshold)
- Learning rate of module B $\alpha_B = 0.2$

Experiments were carried out on an Intel i5-6500 CPU at 3.2 GHz, a NVIDIA Geforce 1050Ti 3 GB GPU and 8 GB memory using CuDNN-supported Tensorflow 2.0.

Tab. 3 lists the average training time, the average final accuracy and the average final accuracy's double standard deviation for every feature extraction algorithm from Tab. 2 on different numbers of randomly drawn training images ranging from 1 to 1250. Averages and deviations are calculated based upon five runs. It can be seen that with an increase in the number of training images per class, the final recognition accuracy rises at the cost of a higher training time. However, while the calculative load increase for higher numbers of training images per class rises sharply, the accuracy gain decreases rapidly. While this is true for every feature extraction algorithm, the extend differs, e. g., MobileNet-V2 already comes close to its peak accuracy using only 50 training images and taking 23.4 s whereas Inception-V3 needs at least 500 images and 493.4 s to do the same. Furthermore, the algorithms greatly vary in their peak accuracy and the training time needed, ranging from 58 % (Inception-V3) to 76.3 % (MobileNet-V2) and from 439.7 s (MobileNet-V2) to 2888.5 s (ResNet152V2).

To further compare the different algorithms' accuracies against each other, Fig. 3 shows their performance on 10, 100 and 1000 training images per class and 1, 2, 5 and 10 incremental learning steps, each averaged over five runs for each of the six feature extraction algorithms listed in Tab. 2 respectively used for module A.

The diagrams on the left side depict the accuracy after each group's training in a scenario where the ten classes are split into ten groups of one class each and learned sequentially using 10 (top), 100 (middle) or 1000 (bottom) training images per class. While the small number of training samples in the top diagram leads to several algorithms not starting out with 100 % accuracy after just one trained class, this gets better with an increase in the number of training images. Inception-V3 consistently performs much weaker than the other feature extraction algorithms, which perform similarly well except for MobileNet-V2 and to a lesser extend DenseNet121 reaching better accuracies.

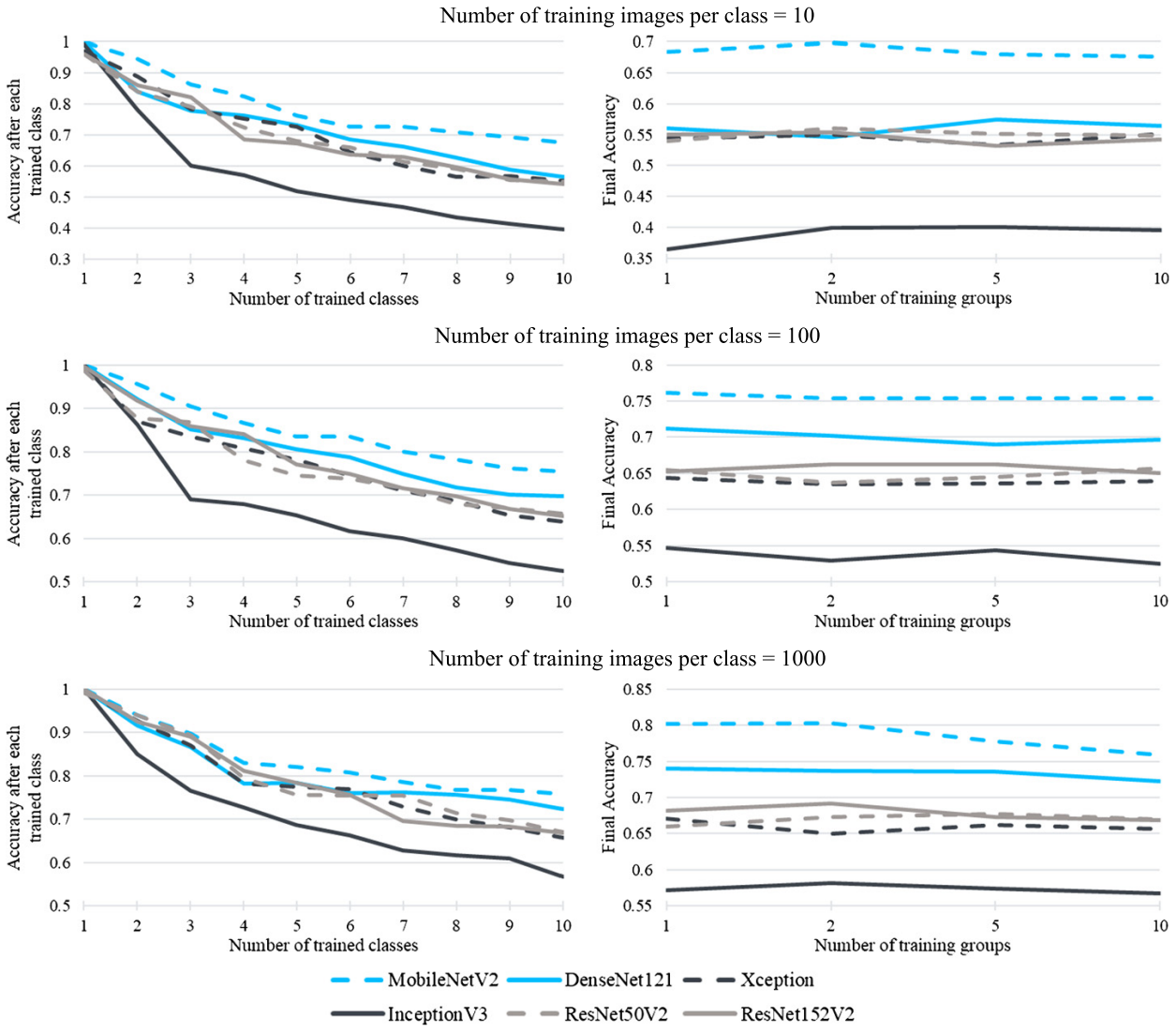

Figure 3: Results for DICLA using different feature extraction algorithms and different parameter sets on the ImageNet-10 dataset.

The right diagrams depict the final accuracy after the complete, sequential training of a scenario where the ten classes are split into 1, 2, 5 or 10 training groups using 10 (top), 100 (middle) or 1000 (bottom) training images per class. Altogether, the number of training groups does not appear to have a large influence on the algorithms' performance. This supports the claim that DICLA is capable of performant incremental learning, thereby mitigating the plasticity-stability-dilemma so that old skills are retained while new ones are learned.

Concludingly, the experiments on ImageNet-10 clearly point towards the use of MobileNet-V2 in module A, combining the highest final accuracies with the lowest training time needed among all feature extractors compared. Furthermore, MobileNet-V2 is the feature extractor with the smallest memory footprint (see Tab. 2).

4.2 Incremental learning: evaluation of DICLA using MobileNet-V2 on ImageNet

Based upon the findings in the previous section, a version of DICLA using MobileNet-V2 in module A is further evaluated. To this extend, experiments are executed on the full ImageNet-dataset, which can be characterized by a large number of different classes (1000) and a large number of highly different natural images per class (>1250), challenging any image recognition algorithm to distinguish small as well as large differences. The well-published algorithms iCaRL and Learning without Forgetting (LwF) [11] are used for comparison (accuracies obtained from [28]).

Training of DICLA is executed using the same parameters as in the previous section and 10 and 100 training images per class.

Table 4: Comparison of the accuracy after each trained group of classes of DICLA using 10 training images per class, DICLA using 100 training images per class, iCaRL and LwF on the full ImageNet dataset (best value indicated in bold print).

Number of Trained Classes	100	200	300	400	500	600	700	800	900	1000
DICLA10 [%]	67.9	60.2	55.4	51.6	47.9	45.8	44.1	42.2	40.8	39.3
DICLA100 [%]	75.1	68.5	64.1	62.5	59.4	57.3	**55.1**	**53.3**	**51.5**	**49.6**
iCaRL [%]	**90**	**83**	**77.5**	**70.5**	**63**	**57.5**	53.5	50	48	44
LwF [%]	**90**	77	68	59.5	52.5	49.5	46.5	43	40.5	39

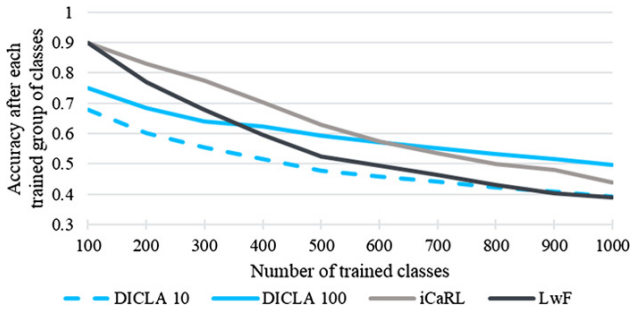

Figure 4: Results of DICLA using 10 training images per class, DICLA using 100 training images per class, iCaRL and LwF on the full ImageNet dataset.

Fig. 4 and Tab. 4 show the resulting accuracies: iCaRL and LwF start out at 90 % accuracy, DICLA using 100 training images per class (DICLA100) at 75.1 % and using 10 training images per class (DICLA10) at 67.9 %. However, over the course of the incremental learning process, this gap between DICLA and the other algorithms narrows considerably. In the end, DICLA10 is even slightly more accurate than LwF (39.3 % to 39 %) and DICLA100 considerably more accurate than iCaRL (49.6 % to 44 %). This is despite the fact, that iCaRL as well as LwF utilize a ResNet-Architecture for feature extraction, which generally achieves a better classification accuracy than MobileNet-V2 (see Tab. 2) and both perform 100 epochs per incremental step and use all about 1,300 training images per class of which iCaRL saves about 20,000.

4.3 Distributed learning: evaluation of DICLA using MobileNet-V2 on ImageNet-10

Finally, DICLA using MobileNet-V2 in module A is experimentally evaluated regarding its distributed learning capabilities. To this end, the ten classes of ImageNet-10 are evenly distributed to 1, 2, 5 and 10 edge devices conducting the training, so that every edge device has the same number of classes to learn. Training is executed using the same parameters as in the previous sections and 100 training images per class. After the training, the knowledge acquired is then shared between the edge devices and the algorithm's performance is tested on the evaluation dataset consisting of the test images.

Tab. 5 shows the results of this experiment: It can be seen that the average final accuracy is almost fully independent from the number of edge devices. Because of the design of DICLA, this could be expected, because representations and class assignments can easily and losslessly be exchanged between different edge devices. Small differences could be explained by the still rather large spread of the results and a slight advantage for more centralized learning, i. e., on a smaller number of edge devices, due to cross-effects.

5 Conclusion and outlook

In order to allow the utilization of the benefits of deep learning without having to aggregate large amounts of diverse data on a single resource or train the algorithms from scratch every time new data is collected, methods allowing a transfer of knowledge between different locations and stages of an algorithm's training should be used.

In combining the differentiation capabilities of transfer learning with the generalization capabilities of continual learning, so-called deep industrial transfer learning is well suited for the challenges imposed by real-life learning scenarios, e. g., in industrial automation. Still, some specific continual learning methods should be applicable to those problems. Therefore, the dual-memory method is used for the complex image recognition task examined in this article.

The algorithm presented is capable of sequential or incremental learning by transferring knowledge from previous trainings to the current one without storing any raw data. Its performance is better than state-of-the-art machine learning methods while using less computational resources and less time. This allows for inline utilization, e. g., in live production lines.

Table 5: Comparison of the impact of the number of edge devices carrying out the training distributedly regarding the final accuracy (single standard deviation indicated) averaged over ten runs.

No. of Edge Devices	1	2	5	10
Avg. Final Accuracy [%]	76.4 (±1.2)	76.3 (±1.5)	73.4 (±2.5)	74.9 (±2.1)

Furthermore, the algorithm is capable of distributed learning by transferring knowledge among different training devices without exchanging any raw data. It could be shown, that such a distribution of training does not significantly deteriorate the learning performance. This allows for cooperative learning approaches across different sites even though the data used might be highly confidential.

Future research shall focus on data types apart from images, more specifically on developing a version adapted to time series datasets as used in e. g., predictive maintenance scenarios. This shall further highlight the potentials of the algorithm presented for industrial automation applications.

References

1. Carpenter, G. A., Grossberg, S.: Adaptive Resonance Theory. Springer, New York, 2010.
2. Chollet, F.: Xception: Deep Learning With Depthwise Separable Convolutions. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, pp. 1251–1258, 2017.
3. French, R.: Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences 4/3, pp. 128–135, 1999.
4. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (USA), 2016.
5. 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). Las Vegas, pp. 770–778, 2016.
6. Hsu, Y.-C., Liu, Y.-C., Ramasamy, A., Kira, Z.: Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines. In NeurIPS Continual Learning Workshop 2018. arXiv:1810.12488.
7. Huang,G., Liu, Z, van der Maaten, L., Weinberger, K. Q.: Densely Connected Convolutional Networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, pp. 4700–4708, 2017.
8. Kagermann, H.: Change Through Digitization—Value Creation in the Age of Industry 4.0. In Albach, H., Meffert, H. et al., (Eds.): Management of Permanent Change. Springer Fachmedien. Wiesbaden, pp. 23–45, 2015.
9. Kemker, R., McClure, M., Abitino, A., Hayes, T., Kanan, C.: Measuring Catastrophic Forgetting in Neural Networks. In 2018 AAAI Conference on Artificial Intelligence. New Orleans, pp. 3390–3398, 2018.
10. Lindemann, B., Fesenmayr, F., Jazdi, N., Weyrich, M.: Anomaly Detection in Discrete Manufacturing Using Self-Learning Approaches. In 2018 CIRP Conference on Intelligent Computation in Manufacturing Engineering. Naples, pp. 313–318, 2018.
11. Li, H., Hoiem, D.: Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence 40, pp. 2935–2947, 2018.
12. Merten, A. M.: Adaptive Resonance Theory [ART] – Ein neuer Ansatz lernender Computer. Lecture Notes, University of Ulm, 2003. http://www.informatik.uni-ulm.de/ni/Lehre/WS03/ProSemNN/ART.pdf, Accessed on: 24.03.2020.
13. Maltoni, D., Lomonaco V.: Continuous learning in single-incremental-task scenarios. Neural Networks 116, pp. 56–73, 2019.
14. Maschler, B., Jazdi, N., Weyrich, M.: Maschinelles Lernen für intelligente Automatisierungssysteme mit dezentraler Datenhaltung am Anwendungsfall Predictive Maintenance. In 20. Leitkonferenz der Mess- und Automatisierungstechnik Automation. Baden-Baden, pp. 739–751, 2019.
15. Maschler, B., Kamm, S., Jazdi, N., Weyrich, M.: Distributed Cooperative Deep Transfer Learning for Industrial Image Recognition. In 2020 CIRP Conference on Manufacturing Systems (CMS). Chicago, pp. 437–442, 2020.
16. Maschler, B., Weyrich, M.: Deep Transfer Learning at Runtime for Image Recognition in Industrial Automation Systems. In 2020 Technical Conference EKA – Design of Complex Automation Systems. Magdeburg, 2020.
17. Maschler, B., Weyrich, M.: Deep transfer learning for industrial automation: a review and discussion of new techniques for data-driven machine learning. Industrial Electronics Magazine (accepted), 2021.
18. Maschler, B., White, D., Weyrich, M.: Anwendungsfälle und Methoden der künstlichen Intelligenz in der anwendungsorientierten Forschung im Kontext von Industrie 4.0. In ten Hompel, M., Vogel-Heuser, B., et al. (Eds.): Handbuch Industrie 4.0. Springer Reference Technik. Springer Vieweg, Berlin, Heidelberg, pp. 1–15, 2015.
19. Müller, T., Jazdi, N., Schmidt, J.-P., Weyrich, M.: Cyber-Physical Production Systems: Enhancement with a Self-organized Reconfiguration Management. In 2020 CIRP Conference on Intelligent Computation in Manufacturing Engineering (ICME). Naples, 2020.
20. van de Ven, G. M., Tolias, A. S.: Three Scenarios for Continual Learning. In NeurIPS Continual Learning Workshop, 2018. arXiv:1904.07734.
21. Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks 113, pp. 54–71, 2019.
22. Pan, S. J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22, pp. 1345–1359, 2010.
23. Rebuffi, S.-A., Kolesnikov, A., Sperl, G., Lampert, C. H.: iCaRL:

Incremental Classifier and Representation Learning. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, pp. 5533–5542, 2017.

24. 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). Salt Lake City, pp. 4510–4520, 2018.
25. 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). Las Vegas, pp. 2818–2826, 2016.
26. Tercan, H, Guajardo, A., Meisen, T.: Industrial Transfer Learning: Boosting Machine Learning in Production. In 2019 IEEE Conference on Industrial Informatics (INDIN). Helsinki, pp. 274–279, 2019.
27. Wang, J., Ma, Y., Zhang, L., Gao, R. X., Wu, D.: Deep learning for smart manufacturing. Journal of Manufacturing Systems 48, pp. 144–156, 2018.
28. 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). Long Beach, pp. 374–382, 2019.
29. Xu, G., Liu, M., Wang, J., Ma, Y., Wang, J., Li, F., Shen, W.: Data-Driven Fault Diagnostics and Prognostics for Predictive Maintenance: A Brief Overview. In 2019 IEEE International Conference on Automation Science and Engineering (CASE). Vancouver, pp. 103–108, 2019.
30. Yao, X., Zhou, J., Zhang, J, Boer, C. R.: From Intelligent Manufacturing to Smart Manufacturing for Industry 4.0. Driven by Next Generation AI and Further On. In 2018 IEEE International Conference on Enterprise Systems. Beijing, pp. 311-318, 2017.
31. Zellinger, W., Grubinger, T., Zwick, M., Lughofer, E., Schöner, H., Natschläger, T., Saminger-Platz, S.: Multi-source transfer learning of time series in cyclical manufacturing. Journal of Intelligent Manufacturing 31, pp. 777–787, 2020.

Bionotes

Benjamin Maschler
Institute of Industrial Automation and Software Engineering, University of Stuttgart, Pfaffenwaldring 47, 70550 Stuttgart, Germany
benjamin.maschler@ias.uni-stuttgart.de

Benjamin Maschler, M. Sc., studied Renewable Energies and Sustainable Electrical Energy Supply at the Universities of Stuttgart and Cape Town. Since 2017, he has been a research assistant at the Institute of Industrial Automation and Software Engineering at the University of Stuttgart. His research focusses on solving practical decentralized deep learning problems without an exchange of datasets, so-called deep industrial transfer learning, using methods from transfer and continual learning.

Simon Kamm
Institute of Industrial Automation and Software Engineering, University of Stuttgart, Pfaffenwaldring 47, 70550 Stuttgart, Germany
simon.kamm@ias.uni-stuttgart.de

Simon Kamm, M. Sc., studied Electrical Engineering and Information Technology at the University of Stuttgart. Since 2020, he has been a research assistant at the Institute of Industrial Automation and Software Engineering at the University of Stuttgart. His research focusses on machine learning based on heterogeneous data sources and types.

Michael Weyrich
Institute of Industrial Automation and Software Engineering, University of Stuttgart, Pfaffenwaldring 47, 70550 Stuttgart, Germany
michael.weyrich@ias.uni-stuttgart.de

Prof. Dr.-Ing. Michael Weyrich teaches at the University of Stuttgart and is head of the Institute of Industrial Automation and Software Engineering. His research focusses on intelligent automation systems, complexity control of cyber-physical systems and validation and verification of automation systems.