53rd CIRP Conference on Manufacturing Systems

Distributed Incremental Class Learning for Live Image Recognition

Benjamin Maschlera\*, Simon Kamma, Michael Weyricha

aInstitut 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 method and is capable of learning new, formerly unknown classes as well as conducting its learning across multiple instances at multiple locations simultaneously without storing any images. In addition to tests on the ImageNet dataset, a prototype based upon a Raspberry Pi and a webcam is described and used for further evaluation. It can be shown that the proposed algorithm allows for the performant execution of image classification tasks while learning new classes at several sites at once, which enables its application to a variety of industry use cases.

© 2019 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)

Peer-review under responsibility of the scientific committee of the 53rd CIRP Conference on Manufacturing Systems

*Keywords:* Artificial Intelligence; Artificial Neural Networks; Deep Learning; Distributed Learning; Incremental Class Learning; Live Image Recognition; Transfer Learning

1. Introduction (Rest der Seite, Ms)

Here introduce the paper, and put a nome­nclature if necessary, in a box with the same font size as the rest of the paper. The paragraphs continue from here and are only separated by headings, subheadings, images and formulae. The section headings are arranged by numbers, bold and 10 pt. Here follows further instructions for authors.

1. Related Work (max. 1 Seite ggf. Etwas mehr, Ms / Ka)

Anlehnung an 1. Teil des Konzeptions-Dokuments: Recherche der unterschiedlichen Modul-Möglichkeiten (Ka)

Nothing I know (Ka)

Continual / Transfer / Federated Learning (Ms)

The Distributed Incremental Class Learning Algorithm bases on the Dual-Memory Method and consist of two modules. A slow-learning module (orientated on the Hippocampus) and a fast-learning module (orientated on the Neocortex). The slow-learning module, named Module A within this work, is a pre-trained Feature-Extraction Architecture. This module can be fix or further adjusted while lifetime with small update steps. In this work this module is fixed, so learning rate is 0. This module extracts features and forward them to the second module, Module B. Module B is an incremental Classificator, who can be trained “on-the-fly”. A graphical overview of the Algorithm is given in Figure 1.

Figure 1: Schematic Overview of the Methodology for Distributed Incremental Class Learning Algorithm

Feature Extraction is nowadays a task which can be effectively done by Deep Neural Networks (DNN). Depending on the application, different DNNs can be used for this module. On the one hand, the module should extract good and relevant features from the input data. But on the other hand, the DNNs should also run in real-time applications on (mobile) edge devices in this work, e.g. on a Raspberry Pi. Therefore, the memory consumption and computational complexity of the DNN is also a relevant criterion. Different DNN-Architectures are compared based on ImageNet in Table 1.

Table 1: Comparison of different DNN-Architectures

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DNN-Architecture | No. Parameters | Memory-Consumption Parameters | No. FLOPs | Top-1 Classification Accuracy | Top-5 Classification Accuracy |
| AlexNet | 60x106 | 240 MB | 0.7x109 | 36.7 % | 15.3 % |
| VGG-16 | 138x106 | 552 MB | 16x109 | 25.6 % | 8.1 % |
| VGG-19 | 144x106 | 576 MB | 20x109 | 25.5 % | 8.0 % |
| ResNet-50 | 25.6x106 | 102 MB | 4x109 | 20.7 % | 5.3 % |
| ResNet-101 | 44.5x106 | 178 MB | 8x109 | 19.9 % | 4.6 % |
| Inception-V3 | 24x106 | 96 MB | 4.8x109 | 21.6 % | 5.6 % |
| MobileNet-V2 | 3.5x106 | 14 MB | 0.3x109 | 28 % | 9 % |

Depending on the application and restrictions, a suitable DNN-Architecture can be selected. In this work, due to focus on memory-restricted Devices, MobileNet-V2 is chosen, because this network is designed for mobile applications with the focus on low memory consumption and operations per input data. Also, the possible classification is acceptable, and so the extracted features from the network should be good enough for classification within Module B. The last classification-specific layers of the DNN, typically the Fully-Connected Layers at the end, are removed and the features are forwarded to Module B.

Module B is an incremental Classificator, which should generally meet the following requirements:

* Trainable on a Data Stream, where samples of the different classes appear randomly in time and order
* All the time there must be a working Classificator for the already seen and trained Classes
* Computational Complexity and Memory Consumption should be limited or grow slowly with respect to the number of known classes

These are general requirements for an incremental Classificator. Additionally, the Algorithm within this work should not save any training data. Based on these Requirements a FuzzyARTMAP-Network was chosen. A big advantage is the property of the FuzzyARTMAP to solve the plasticity-stability dilemma due to his architecture. The FuzzyARTMAP-Network compare input data to known representations of categories (details in the following section). If a fix or adaptive threshold (called Vigilance Parameter) is not reached, the so called Nothing I Know-Mechanism is active.

Nothing I Know detects an unknown class with the help of an adaptive threshold. It is calculated based on the matching-values in the network and the number of category nodes. With a factor, the “clearness” of the winning node is adjusted. The final threshold is given by the following formula:

Depending on the application, the threshold could also be a fixed value for.

1. Methodology (max. 1 Seite eher weniger, Ka)

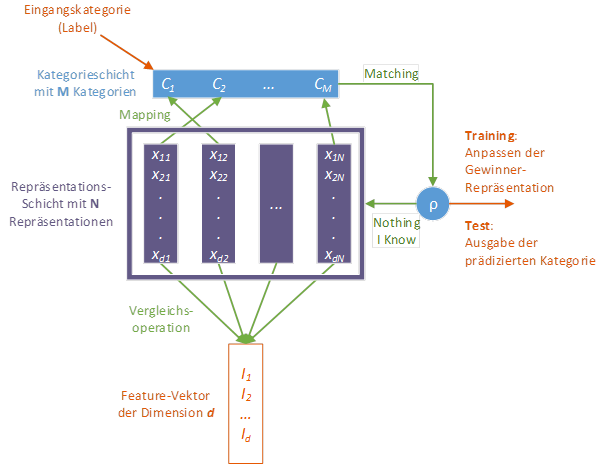
In this work, MobileNet-V2 is used as Feature Extraction module. It is designed for mobile application with low memory consumption and operations per input image. The available and pre-trained MobileNet-V2 Architecture within Tensorflow 2.0 is used. Pre-Training took place on ImageNet. The last fully connected layers of MobileNet-V2 are removed and the features will be the input to Module B.

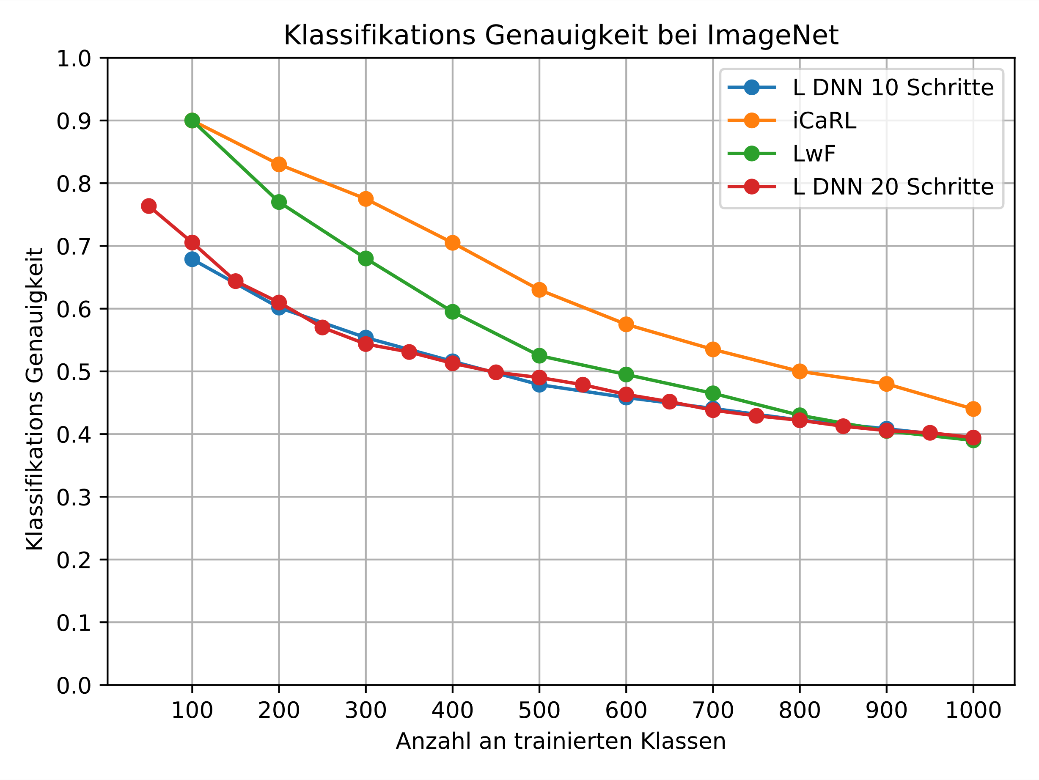
Figure 2: Structure of the FuzzyARTMAP-Network

The chosen FuzzyARTMAP-Network was adjusted slightly, and the principal structure can be seen in Figure 2. The input data , here the extracted features from Module A, is a *d*-dimensional vector. In the so-called representation layer are *N* representations , which are each realized with a *d*-dimensional vector, too. Every representation performs a comparison operation. In this work this operation is realized with the cosine similarity. After this step, the Mapping from the representations to the corresponding classes take place in the Category Layer. A Class can have more than one representation in this network. After this mapping the Categories have so called Matching-Values, which can be compared to the (adaptive) threshold . If the “winner”-representation has a matching value above the threshold, the winning class will be the prediction of the network while inference. In Training the winning representation will further be adjusted and generalized with the input data and learning rate with the following formula:

If the threshold is not reached, the network declares the input as Nothing I Know. In training phase, a new representation is created with the input vector and the training label is used to perform a correct mapping to the Category Layer. While Inference, the network forces the user to give a label for the input data, or it can just randomly give the input data a class label, e.g. 8 if 7 classes are known already.

Additionally, the FuzzyARTMAP has the possibility to execute a consolidation step. There all representations of one class are summarized to reduce the memory consumption of the representation layer. In this work, this consolidation is done by a simple mean-value calculation of the available representations, but also a more complex logic could be implemented.

1. Experiments (1 Seite eher ein bisschen mehr, Ka / Ms)

**Experimental Setup (Ka)**

The following experiments are executed on the ImageNet-dataset. Due to time restrictions for some experiments an ImageNet-10 dataset was used, where 10 random classes are drawn from the ImageNet-Dataset. For ImageNet 224x224x3 images are used, while for ImageNet-10 96x96x3 images are used. ImageNet-10 consists in this work out of the following classes (class indices are given): 145, 153, 289, 404, 405, 510 ,805, 817, 867, 950. Training of the DICL-Algorithm was executed with one epoch per incremental step, so every training image is seen just once. Based on a Hyperparameter-Optimization, the following parameter are chosen and used in the following:

* Train images per class: 100 (ImageNet-10)/ 10 (ImageNet) – randomly drawn
* Test images: 50 (all available images)

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

* (fix threshold for this datasets)
* (learning rate)

For these evaluations, a fix threshold was used, so the Nothing I Know-Concept was not used here, since a lot of user feedback would be necessary on these big datasets.

**Incremental Learning Performance**

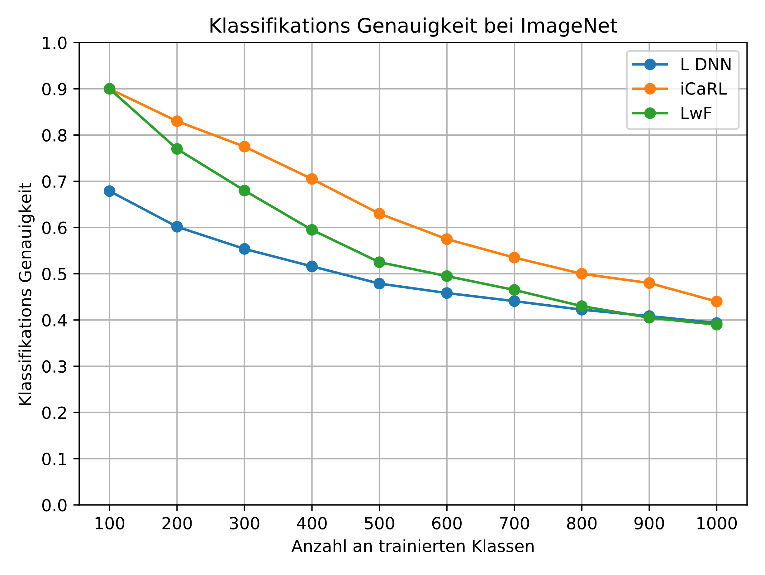
For an evaluation of the proposed algorithm different tests on ImageNet are performed. As a reference, iCaRL and Learning without Forgetting (LwF) are used. iCaRL and LwF use a ResNet-Architecture as Feature Extraction Module, which achieves better classification accuracy than MobileNet-V2 (see Table 1). They perform 100 epochs per incremental step and use all training images (around 1300 per class). iCaRL saves 20.000 training images to achieve these results. Results are obtained from [x].

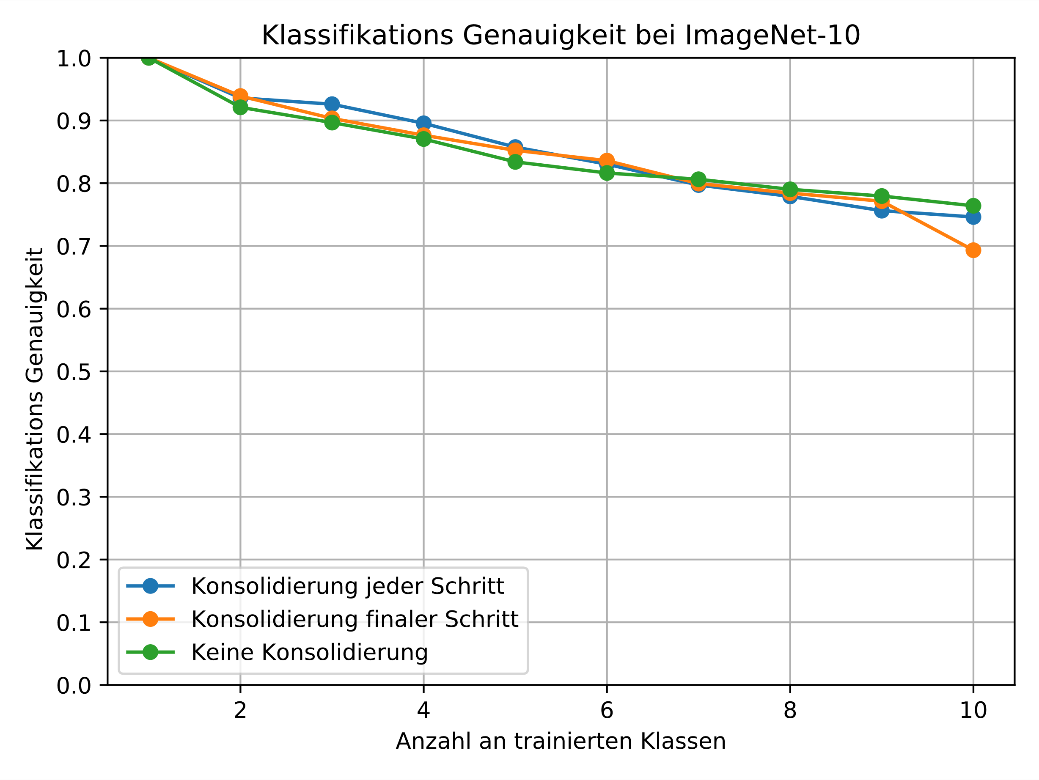
Figure 3: Results for ImageNet, 10 incremental steps

The results for the three Algorithms with 10 incremental steps are shown in Figure 3. In the beginning there is a big performance gap between LwF and iCaRL on the one hand, and DICL on the other hand. But after performing all incremental steps, DICL achieves better results than LwF and has similar results as iCaRL, although no training data is stored, and a worse feature extraction module is used.

Additionally, the DICL Algorithm was tested with different number of the incremental steps. The other two algorithms, iCaRL and LwF are sensitive to the choice of the number of incremental steps. Therefore, the ImageNet-Dataset was tested with 20 incremental steps and with one step, where all classes are learned in one training step. The results for 20 incremental steps are shown in Figure 4. The curve is like the curve with 10 incremental steps, so 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 DICL algorithm are given in Table 2.

Table 2: Final Classification Accuracy ImageNet

|  |  |  |
| --- | --- | --- |
| Algorithmus | Finale Klassifikations-Genauigkeit ImageNet in % | Finaler Speicherbedarf ImageNet-10 in MB |
| L DNN  (one incremental step) | 39,1 | 527,2 |
| L DNN  (10 incremental steps) | 39,3 | 530,5 |
| L DNN  (50 incremental steps) | 39,4 | 528,8 |
| iCaRL | 44 | 2123 |
| LwF | 39 | - |

With the help of consolidation, the memory consumption and computational complexity of the DICL-Algorithm 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 after all incremental steps before calculating the test accuracy. The results can be seen in Figure 5.

The training on this dataset without any consolidation is used as reference (green curve). The blue curve shows the results for consolidation after every incremental step, the orange curve shows the results for consolidation after the final step. The final classification results and the memory consumption for the three strategies is given in Table 3.

Table 3: Final results for the different consolidation methods

|  |  |  |
| --- | --- | --- |
| Consolidation Method | Final Classification Accuracy ImageNet-10 in % | Final Memory Consumption ImageNet-10 in MB |
| **No Consolidation** | 76,40 +/- 1,2 | 0,96 |
| **Consolidation every step** | 74,62 +/- 2,19 | 0,1 |
| **Consolidation after final step** | 69,32 +/- 4,81 | 0,1 |

It can be seen, that the memory consumption can be reduced drastically to around 1/9 of the value. With the consolidation at every step the final classification accuracy (74,62%) is close the accuracy without consolidation (76,40%). With the consolidation after the final step the accuracy is lower (69,32%). With the suitable consolidation strategy, the memory consumption can be reduced significantly by only reducing the performance (here classification accuracy) slightly. So, this method is useful for applications with few available memory.

**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 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 of Module B is exchanged and the final accuracy on all classes is tested. As reference the incremental class learning on ImageNet-10 with one edge device is used. The mean accuracy with standard deviation of ten runs are given in Table 4.

Table 4: Distributed Learning Performance on ImageNet-10

|  |  |
| --- | --- |
| No Edge Devices | Final Classification Accuracy ImageNet-10 in % |
| 1 | 76,40 +/- 1,2 |
| 2 | 76,26 +/- 1,5 |
| 5 | 73,36 +/- 2,5 |
| 10 | 74,86 +/- 2,1 |

As it can be seen, the performance is almost stable. The deviation of the final accuracy increases by a higher number of devices, but the final accuracy is not decreasing significantly by distributing the training on more (5 or 10) devices and meld the knowledge afterwards. The DICL-Algorithm is thus capable of training on local data and exchange the knowledge with other devices without sharing the training data.

Figure 5: Impact of Consolidation

**Bessere Performance mit ResNet (Ms)**

Anlehnung an Evaluationsdokumente

Weitere Experimente im Verlauf des Herbstes (Ms)

1. Prototype (max. 0,5 Seiten, Ms)

Beschreibung des Raspberry-Pi-basierten Prototyps

Bild des Prototypen

Ggf. Abbildung der GUI aus Ausarbeitung

1. Conclusion and Transfer (max. 0,5 Seiten, Ms)

Modularität: Einfache Anpassbarkeit auf Anforderungen (bessere Hardware erlaubt bessere Ergebnisse)

Transfer im Sinne eines Ausblicks auf potentielle Anwendungsfelder?

References

[1] Van der Geer J, Hanraads JAJ, Lupton RA. The art of writing a scientific article. J Sci Commun 2000;163:51-9.

[2] Strunk Jr W, White EB. The elements of style. 3rd ed. New York: Macmillan; 1979.

[3] Mettam GR, Adams LB. How to prepare an electronic version of your article. In: Jones BS, Smith RZ, editors. Introduction to the electronic age. New York: E-Publishing Inc; 1999. p. 281-304.