Journal of Intelligent Manufacturing (2022) 33:283–292
https://doi.org/10.1007/s10845-021-01793-0

Continual learning of neural networks for quality prediction in production using memory aware synapses and weight transfer

Hasan Tercan[1] · Philipp Deibert[1] · Tobias Meisen[1]

Received: 19 January 2021 / Accepted: 26 May 2021 / Published online: 5 June 2021

Abstract

Deep learning-based predictive quality enables manufacturing companies to make data-driven predictions of the quality of a produced product based on process data. A central challenge is that production processes are subject to continuous changes such as the manufacturing of new products, with the result that previously trained models may no longer perform well in the process. In this paper, we address this problem and propose a method for continual learning in such predictive quality scenarios. We therefore adapt and extend the memory-aware synapses approach to train an artificial neural network across different product variations. Our evaluation in a real-world regression problem in injection molding shows that the approach successfully prevents the neural network from forgetting of previous tasks and improves the training efficiency for new tasks. Moreover, by extending the approach with the transfer of network weights from similar previous tasks, we significantly improve its data efficiency and performance on sparse data. Our code is publicly available to reproduce our results and build upon them.

Keywords Continual learning · Deep learning · Artificial intelligence · Manufacturing · Predictive quality · Regression

Introduction

Predictive quality enables manufacturing companies to make data-driven in-process predictions of the quality of a produced product based on process data. The general approach to predictive quality involves three main steps: the collection and aggregation of process and quality data, the training of a predictive model, and the use of the model for real-time predictions as a basis for decisions on measures to be taken in the process. Machine learning and especially deep learning methods based on neural networks enable such predictions based on multi-modal process, sensor and machine data. In the current state of research, there are already many examples that successfully demonstrate the feasibility of deep learning based predictive quality in various manufacturing processes such as deep drawing, hydrocracking, lasermachining, or additive manufacturing[1] (Baumeister et al. 2018; Meyes et al. 2019; Yuan et al. 2020; Li et al. 2020; McDonnell et al. 2021; Hsu and Liu 2021).

The mentioned examples mainly focus on a particular learning problem, where the training of a neural network happens under the assumption that enough data is available for the respective problem. However, this assumption is often not met in production. In fact, a central challenge is that production processes are subject to continuous changes. For example, as soon as a new product is manufactured or a process is reparameterized, the process behavior changes and with it the relationships between process and quality data. Consequently, a lot of new process data would have to be collected each time to train another completely new model on it (Escobar et al. 2021). This strongly limits the sustainable use of deep learning in the production context, especially since the collection of representative process data is costly and time-consuming. Other common problems in the production domain are that, due to limited hardware capacities or corporate policies, long-term process data cannot be stored or accessed and model training must be carried out in a

✉ Hasan Tercan
tercan@uni-wuppertal.de

[1] Chair of Technologies and Management for Digital Transformation, University of Wuppertal, Rainer-Gruenter-Strasse 21, Wuppertal, Germany

[1] https://github.com/tmdt-buw/continual-learning-mas-cloning-injection-molding.

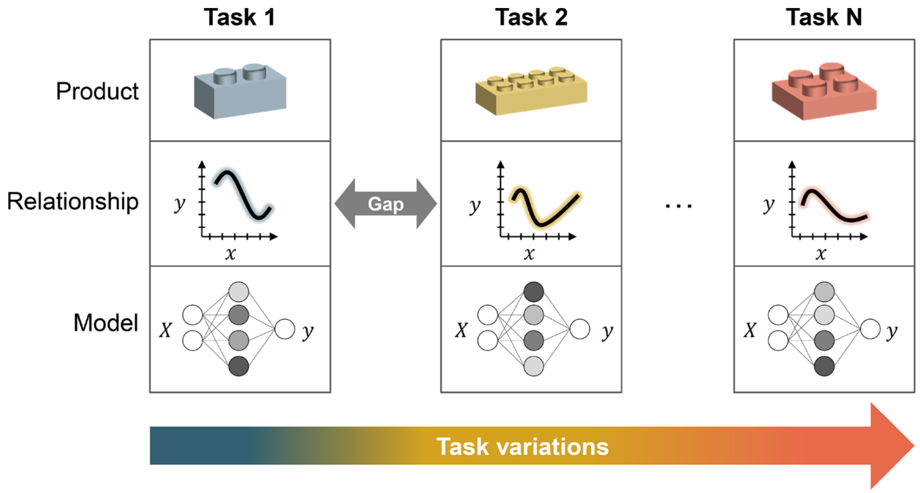

Fig. 1 Use of a neural network for regression to estimate the product quality y (here part deformation) on the basis of machine parameters X (e.g. pressure and time). With each plastic brick, the relationship between X and y changes. Note that each new product represents a new prediction task while the input and output always remain the same for every task

resource-friendly manner. It is therefore necessary to address this research gap and to find solutions for the efficient training of neural networks across production process variations with sparse data (Wang et al. 2018).

We believe that potential solutions lie in the research field of continual learning, a paradigm in deep learning that addresses the training of neural networks over multiple (similar) tasks. The common goal in continual learning is to keep the training effort low (i.e. reduced computational effort, increased memory efficiency) and to prevent the so-called catastrophic forgetting of the networks with each new task.

In this paper, we address this issue and demonstrate the successful application of continual learning for a real use case in injection molding, where we train a neural network for numerical prediction of product quality based on machine parameters. Tercan et al. (2018) demonstrated the feasibility of such predictions in previous work. However, the network is also continually confronted with new prediction tasks due to product changes. Figure 1 schematically illustrates this problem. Our goal is to investigate a learning process of a single neural network across theses prediction tasks. Since many production environments have the above mentioned constraints, our approach must meet the following criteria:

- *Learning without forgetting*: when training an already existing neural network for new tasks (i.e. products), its knowledge of the old tasks should not be forgotten and the model quality should not deteriorate.
- *No need to access data from previous tasks*: due to limited hardware capacities and restricted access to data, the training of new tasks should not involve process data from older tasks.
- *Strong performance on sparse data*: the network training for a new task is data efficient and should involve less process data than when learning a new neural network from scratch.

The two main contributions of this paper are:

1. We gain useful findings for the use of continual learning in a real-world predictive quality case in injection molding. We therefore perform extensive experiments with an existing continual learning method (i.e. memory-aware synapses) and evaluate its feasibility and benefit by comparing it with baseline methods.
2. We provide a valuable extension of the method by transferring (i.e. cloning) neural network weights from previous tasks. As a result, we achieve improved performance on sparse data and can also satisfy the criteria mentioned above.

The paper is organized as follows: Sect. 2 presents the state of the art of continual learning and discusses it with respect to the criteria posed. Section 3 describes our approach, which is based on the memory-aware synapses method for continually learning neural networks. In Sect. 4, the injection molding use case and the experimental setup for evaluation is provided. The evaluation results and discussions are provided in Sect. 5. Finally, Sect. 6 briefly summarizes the main issues of this paper and gives an outlook on the future research.

State of the art

Continual learning

One major assumption of machine learning learning is that the training data for a model is drawn from the same domain and shares the same characteristics (e.g. input features, distributions) as the test data. This ensures that models generalize well to new unseen data. However, in many real world scenarios this is not the case and the training data for a learning task becomes available only during a certain time.

In such cases, a new model would have to be trained from scratch on new data for each new task. Continual learning is a paradigm of machine learning that tackles this problem and deals with training machine learning models over time in such a way that they can both acquire knowledge for new tasks and retain knowledge from previously trained tasks (Parisi et al. 2019; Chen and Liu 2018). A related paradigm to continual learning is transfer learning, which has also been successfully investigated in industrial applications, such as by Zhao et al. (2020) or Zellinger et al. (2020). The main idea of transer learning is also to leverage the knowledge of a model pre-trained on a source task to a given new task (Pan and Yang 2009; Tan et al. 2018). However, when training neural networks with transfer learning, they may suffer from catastrophic forgetting. As soon as they are trained for sequentially occurring tasks, their performance for previously learned tasks drops due to changes of their parameters respectively network weights (Goodfellow et al. 2013). In contrast, continual learning methods address this issue and try to find a trade-off between the stability and plasticity of network parameters when training on new tasks. State-of-the-Art methods for continual learning can be roughly divided into three categories: memory-based rehearsal strategies, dynamic architectures, and regularization strategies (Parisi et al. 2019).

Rehearsal methods use a fixed sized memory to store data samples from previously trained tasks. Theses samples are then later revisited during the training of new tasks in order to mitigate catastrophic forgetting. For example, Rebuffi et al. (2017) keep an episodic memory with representative samples for each task. When training new tasks, they calculate an additional distillation loss to prevent the network's predictions for these samples from changing significantly. In contrast to that, Lopez-Paz and Ranzato (2017) use the memory to compute the network's gradients for previous tasks. They then formulate the learning of a new task as a dual optimization problem allowing the calculated gradients to minimize both the new loss and the previous losses. Shin et al. (2017) propose a pseudo-rehearsal approach that uses an autoencoder as a generative model to replay previous tasks and to generate new data for each of them when training a new task.

Approaches with dynamic architectures change the architecture of the network when training for new tasks. Often, they dynamically expand the capacity of a network in order to learn new patterns without conflicts. Parisi et al. (2017) use a growing when required (GWR) approach to train recurrent self-organizing neural networks that are hierarchically extended for new tasks. Rus et al. (2016) propose a progressively growing neural network, with the network being extended by an additional column when trained on a new task. By freezing the previous columns and using lateral connections to the new column, both catastrophic forgetting is prevented and previously learned knowledge is reused for the current task. Schwarz et al. (2018) consolidate previously learned knowledge in a base column (i.e. knowledge base) by means of knowledge distillation. For new tasks an additional active column is trained, which again is connected to the basis and can therefore leverage previous knowledge and at the same time acquire new knowledge.

Regularization strategies reduce catastrophic forgetting of neural networks by restricting network parameter updates while training on new tasks. In elastic weight consolidation (EWC) by Kirkpatrick et al. (2017), this is realized by penalizing changes of parameters that are important for previous tasks. The parameter importance is estimated via probability densities using the fisher information matrix. Because of that, EWC is suited for continual classification tasks. In contrast to that, the memory-aware synapses approach proposed by Aljundi et al. (2018, 2019) represents the importance of network parameters by the sensitivity of the network output to changes of the parameters. By incorporating changes of important parameters into the loss function, mainly those networks weights are adapted to a new task that are not yet important. A number of other regularization strategies exist that constrain the learning of a network in a different way. Zhizhong and Hoiem (2018) use a distillation loss to prevent the actual outputs (i.e. predictions) of the updated network from deviating too much from those of its older version. Pomponi et al. (2020) applies regularization to penalize significant changes of a network's embeddings (i.e. activations) for previously learned tasks.

The mentioned methods show promising results and most of them could also be applied for predictive quality scenarios with varying process conditions. However, only few of the them fulfill the three requirements defined in Sect. 1. Although rehearsal methods can yield the best results (Hsu et al. 2018), they require access to old data and a sufficiently large memory for training. Dynamic architectures manage learning without forgetting but are mostly inefficient and do not scale well with the number of tasks. Among the established regularization methods, the memory-aware synapses approach by Aljundi et al. (2018) is most suitable for our use case because this method can be used for regression (unlike EWC by Kirkpatrick et al. (2017)) and it does not require access to old data (like by Lopez-Paz and Ranzato (2017)). The importance of the network weights for a task is computed only once and withheld for future training. Our proposed approach in this paper is therefore mostly based on memory-aware synapses (see detailed description in Sect. 3).

Continual learning in industrial applications

Methods of continual learning are not only researched in the core research areas of machine learning such as computer vision and natural language processing, but are also evaluated in applied research areas. In the field of medicine, for example, methods like elastic weight consolidation are used for semantic segmentation (Baweja et al. 2018; Karin van Garderen et al. 2019) and classification of medical images (Matthias Lenga et al. 2020) or for pattern recognition in physiological time series data (Kiyasseh et al. 2020). In the field of engineering, research effort is put into continual learning strategies for robotic systems based on supervised learning or deep reinforcement learning scenarios (Lesort et al. 2020; Wong 2016). The goals are mainly to enable intelligent and autonomous agents to perform lifelong learning of new tasks and to overcome the challenge of needing large data sets from real robotic controls by means of incremental learning strategies. Examples are found by Dehghan et al. (2019) and Ayub and Wagner (2020), where both work deal with visual object detection systems for robotic tasks that allow incrementally adding new objects.

With regard to manufacturing and production, few works exist yet on the use of continual learning methods. Tercan et al. (2019) continually train neural networks for predictive quality tasks, where the learning approach consists of a finetuning on the new task and a subsequent retraining on data of the old tasks, yielding better learning rates than training models from scratch. Maschler et al. (2020) use EWC for fault prediction of turbofan engines based on LSTM networks. Their results show that though EWC successfully prevents forgetting in similar tasks, its performance deteriorates as soon as the tasks are very different from each other. Tian et al. (2021) propose an online learning method for Long Short Term Memory (LSTM) networks in vibration signal prediction.

Approach

Memory-aware synapses (MAS) by Aljundi et al. (2018) is a regularization-based continual learning approach for training a neural network across a sequence of consecutive tasks T_n. Given a neural network with parameters θ_i (i.e. network weights), changes of these parameters are penalized proportionally to their importance Ω_i w.r.t. to previously learned tasks. The importance values are derived from the respective output layer's sensitivity to the parameter changes. Considering a model trained on a task T_A with training data examples $x_1, \ldots, x_N$ and learned parameter values $\theta_{A,i}$, the importance of each parameter for T_A is then estimated using the gradients w.r.t. the squared l_2-norm of the output layer:

$$\Omega_{A,i} = \frac{1}{N} \sum_{k=1}^{N} \left| \frac{\partial l_2^2(O_A(x_k), 0)}{\partial \theta_{A,i}} \right|$$

where O_A is the network's output for the data examples and 0 is zero vector of same size as $O_A(x_k)$. In order to learn another task T_B, we create a separate output head for T_B and use an additional loss term during the training:

$$L(x) = L_B(x) + \lambda \sum_i \Omega_{A,i} \cdot (\theta_i - \theta_{A,i})^2$$

where L_B is the regular loss of choice for the task and θ_i the current parameter values during training. The hyperparameter λ is a positive real number representing the weighting of the additional Ω-loss term. The term $\theta_i - \theta_{A,i}$ describes the change in the network parameter value from its original value after learning T_A. This approach allows to leverage the network's already acquired knowledge using shared parameters while minimizing changes of important network parameters. After training on task T_B, we then estimate the importance values $\Omega_{i,B}$ w.r.t. T_B and accumulate them with the previously computed values:

$$\Omega_{AB,i} = \Omega_{B,i} + \gamma\ \Omega_{A,i}$$

Note that this procedure can easily be extended to arbitrary numbers of tasks. Figure 2 illustrates the approach for three tasks. In the equation above, γ represents another positive real number hyperparameter adjusting the impact of previous tasks. In our injection molding use case, each task is equally important in terms of forgetting, which is why we set $\gamma = 1$. In general, the optimal selection of the hyperparameters λ and γ depends on the learning problem, the tasks and the training data and must be evaluated individually for each use case. In Sect. 5.1 we show the results of our hyperparameter search.

The described MAS method creates randomly initialized output heads of the network for each new task. However, we want to take advantage of previously learned task when training on new tasks and propose an extension of the method based on the idea of transfer learning: instead of randomly initializing an output head before training a new task, we transfer (i.e. clone) the weights of a already trained head from a similar previous task and subsequently finetune it on the data for the new task. More precisely, we extend the MAS method by the following steps before training the model on a new task:

1. Iterate through all pre-trained output heads of the model and compute the loss of the network with each head on the new task data.
2. Identify the head with the lowest loss.
3. Create a new output head for the new task by cloning the parameter values of the identified output head.
4. Start training the model on the new task data according to MAS.

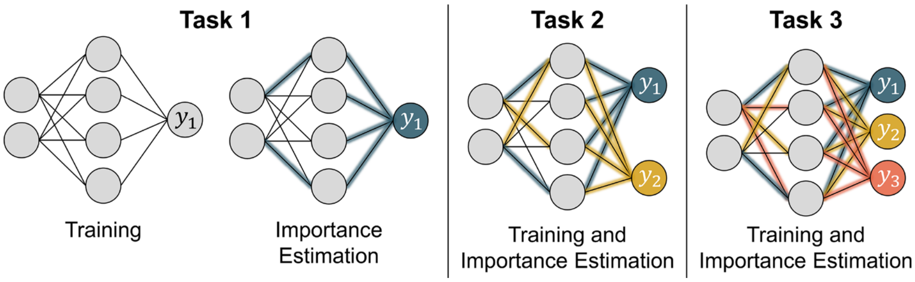
Task 1
Task 2
Task 3
y_1
y_2
y_3
Training
Importance Estimation
Training and Importance Estimation
Training and Importance Estimation

Fig. 2 MAS method for training the network across three consecutive tasks. Highlighted network edges represent weights that are important for the respective task output

In the following, we refer to this approach as *MAS-Cloning*. Our assumption behind this approach is that similar prediction tasks in the given injection molding use case also lead to similar parameter values for their respective output heads. Our experimental results in Sect. 5 show that this assumption is verified.

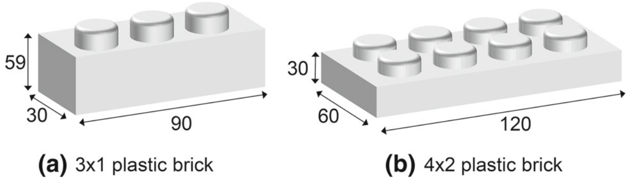

Fig. 3 Used plastic brick parts with different numbers of studs on the top of the bricks and part heights: **a** 3 studs in a single row, **b** 4 studs in two rows with 50% height

Use case and experimental setup

Data basis

Injection molding is a manufacturing process for producing plastics in a single production steps (Kashyap and Datta 2015). During the process, plastics material is plasticized and subsequently injected into the cavity of the mold, where it is formed and cooled down to the final shape of the product. Injection molding involves complex behavior of interdependent process variables, with the mold design and the settings of the machine parameters in particular having a major impact on the quality of the product produced. The principle goal in this use case is to train a neural network for predicting the part quality, namely the maximum deformation under load, on the basis of six machine setting parameters (holding pressure level, holding pressure time, mold temperature, cooling time, melt temperature, and volume flow). Since the process behavior changes when producing a different product, we apply our proposed continual learning method to improve the performance of the model when applied to the data of a new part.

For evaluation purpose, we conducted molding simulations of different plastic brick specimens that provide the data basis. We designed 16 plastic bricks of different sizes by varying the number of studs on top of the brick (3, 4, or 6 studs per row, 1 or 2 rows) and the height of the bricks. Figure 3 illustrates two exemplary bricks with 3 × 1 studs and 4 × 1 studs. The simulations were performed with the software Cadmould 3D-F. For each part respectively prediction task, we varied the parameters in a central composite experimental design with 77 examples.

Experiments and pre-tests

In our experiments, with the exception of those in Sect. 5.1, we train the neural network incrementally over sequences with 16 consecutive tasks (one base task and 15 following incremental tasks). Thereby, all increments are conducted in a sequential way. The learning of the first (base) task involves an untrained network and is regarded as the zeroth increment. In each increment, we train the model on training data of a new task and compute its loss a on separate test set. Thereby we use the last trained model for the new increment.

To obtain reliable results, we run the experiments on ten different task sequences, each with a randomly selected task order, and each sequence with five different shuffles for training and holdout testing data set (thus 60 experiments in total). In addition, we also run the experiments on a reduced data set (45% of training data) to investigate how well the approaches perform on sparse data.

For the evaluation of the methods, we are mainly interested in both their ability to learn new tasks as well as the capability to retain already learned tasks. For the former, we compute the mean squared error (MSE) on the test data of a new task. For the latter, we compute the backward loss for an increment, which defines the (positive or negative) effect of the increment on the model's performance on all previously trained tasks:

$$\text{Backward\_Loss}_t := \frac{1}{t}\sum_{i=0}^{t-1}(L_{i,t} - L_{i,t-1})$$

where $t > 0$ is the number of the increment of interest and $L_{i,j}$ refers to the test loss (MSE) of the ith task after the

Table 1 Parameters and values varied for the initial hyperparameter grid search for the neural network

Hyperparameter	Values
Number of hidden layers	[1,2,3]
Number of neurons per layer	[10,20,50,100]
Activation functions	$[ReLU, Tanh]$
Learning rate	[0.01, 0.001, 0.0001, 0.00001]

jth increment. A positive value—i.e. an increase in average loss—corresponds to a negative effect on previous tasks. Conversely, a negative value signifies that the tasks actually benefited from learning the new task. For a sequence of t increments, we can further compute the average backward loss over all its increments.

We compare both approaches, the regular MAS method and MAS-Cloning, with two baseline approaches:

- *From scratch*: for each task a new neural network is trained from scratch, thus leading to 16 different networks by the end of the sequence.
- *Finetuning*: a neural network is further trained on new tasks via finetuning and without any regularization, thus leading to a single network with a single output head by the end of the sequence. Because the training in each increment uses an already pre-trained network without any restriction (regularization), we assume that the finetuning approach provides the best possible forward transfer and thus the lower limit for the average new task loss.

We implement our methods using PyTorch, an open source library for deep learning (Pytorch 2020). Prior to the main evaluation, we conducted initial grid search based tests on the data to identify the best performing topologies and hyperparameters for the neural network (see Table 1 for parameter ranges), resulting in a two-layer multi-layer perceptron (MLP) with 20 neurons per layer and the Rectified Linear Unit (ReLU) activation functions. Each training is performed using early stopping with a patience of 50 epochs and a validation set size of 20% of the available training set. The training is performed using the Adam optimizer (Kingma and Ba 2017) with a batch size of 16 and a learning rate of 0.001 throughout.

The result of the grid search, which was performed on the data sets using fivefold cross-validation, provided a MSE of 0.037 for the best performing neural network. In order to better assess this performance and to know how good the network is, we performed the same experiments with a polynomial regression of degrees two and three for comparison. The MSEs of these two methods are 0.046 and 0.044, respectively, thus significantly higher than the error of the neural network.

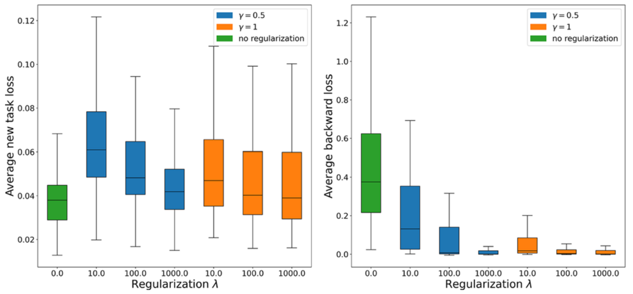

Fig. 4 Performance comparison of different values for the hyperparameters λ and γ regarding the average loss after training new tasks (left) and the average backward loss for all previous tasks (right). Note that the regular MAS method was used here without cloning of the output heads

Results

Hyperparameters for MAS

MAS has basically two hyperparameters, namely λ and γ, which affect the abilities to learn new tasks and to avoid catastrophic forgetting. In order to find the best possible selection, we conduct an initial hyperparameter search by performing reduced experiments with 10 random sequences of seven tasks (one base task and six incremental tasks). Each experiment is further performed with five different data set shuffles. The box plots in Fig. 4 show the results for selected parameter combinations. Note that MAS with $\lambda = 0$ (green bars) does not use any regularization for the network training w.r.t. the Ω-loss.

The results clearly illustrate the trade-off between avoiding forgetting and learning of new tasks. On average, the model without regularization performs best on new tasks (mean loss of 0.04), but suffers greatly from forgetting. In contrast, MAS effectively prohibits forgetting for large λ values while only resulting in minor performance losses for new tasks. The best performing values are $\lambda = 1000$ and $\gamma = 1$ with a mean task loss of 0.05 and a mean backward loss of 0.04. We use these values in all subsequent experiments. It can also be seen that the loss values in the experiments vary significantly. We assume that this is mainly due to the fact that the plastic bricks and prediction tasks sometimes differ greatly from each other.

Avoiding catastrophic forgetting

Table 2 (left columns) provides the computed average task losses and backward losses across all experiments when using 100% of the available training data. The task loss

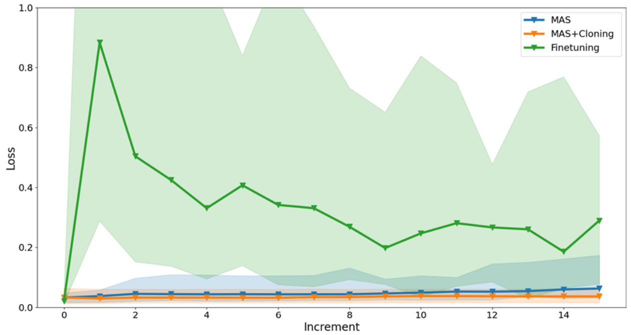

Fig. 5 Comparison of three approaches regarding catastrophic forgetting. Lines represent the evolution of the test loss for the base task after all increments (mean curves and the interquartile ranges across all experiments)

results show that though the MAS methods are constrained by regularization, they can learn new tasks similarly well as unconstrained models. In particular, MAS cloning performs best on new tasks—even slightly better than the finetuning approach. This also applies to the backward loss. While, as expected, simple finetuning causes the neural network to underperform on old tasks, both MAS approaches strongly prevent the network from forgetting when training on new tasks. Cloning has also a positive effect on forgetting. The reason might be that using pre-trained network weights leads to faster convergence of training and thus to fewer changes of the parameters. As a consequence, cloning leads to an average backward loss of almost 0.

The difference between regularization (MAS) and finetuning on forgetting becomes clear when looking at the network performances across the whole task sequences. Figure 5 shows how the task loss for the base task (increment 0) changes over all following increments. While the loss remains almost constant with MAS-Cloning and slightly increases for the regular MAS, finetuning significantly deteriorates the performance. Interestingly, its loss slowly decreases again with each task—a phenomenon we have observed often, but without the model ever reaching the original loss value again.

Training with sparse data

In order to evaluate the performances on sparse data, we conduct the same experiments with a reduced training data size of 45%. Table 2 (right columns) provides their results. On the one hand, they show that still both MAS approaches prevent the network from forgetting, though the backward loss values slightly increase. On the other hand, finetuning and MAS-Cloning perform best for new tasks. In particular, cloning the output head significantly improves the data efficiency of our method. While the regular MAS method suffers from the data reduction, cloning ensures that even with sparse data very good performances are achieved, which are almost as good as a model trained from scratch on 100% of the training data. The differences between the methods are also illustrated in Fig. 6a, which presents the loss values for each increment. It can be seen that finetuning and MAS cloning perform better in each increment than regular MAS, which sometimes performs very poorly, and training a new network from scratch.

The finetuning approach performs almost as well with only 45% of the training data as with 100% of the data. This raises the question of how much data all other methods need for training a new task to achieve similarly good performances. For this purpose, we conduct further experiments by training the networks on different proportions of target training data. On this basis, we identify the proportion that is required to achieve the same test loss (with 10% margin) as with training on 100% of the data within the respective increment. Figure 6b depicts the proportions for all evaluated approaches. It shows that both finetuning and MAS-Cloning have a higher data efficiency than for example the regular MAS method. Starting with the first increment, MAS-Cloning requires only about 50% of the training data to achieve a similarly good performance as with 100% of the data. However, it can also be seen that its data efficiency does not improve over the course of the increments, so that a certain amount of training data is always necessary.

Cloning and task similarity

The results of our experiments show that cloning the output head of a similar previous task has a significant positive impact on the forward transfer of the neural network. To show that cloning must happen from a similar task, we compare our approach with a minor modification: instead of selecting the output head with the lowest loss on the new data for the initialization, we use the one with the highest error (called negative cloning). Figure 7a shows its test results, where the experiments are conducted the same as before. Compared to MAS-Cloning, negative cloning not only provides worse performance, it actually results in a larger loss than MAS with randomly initialized weights. This indicates a knowledge transfer that has a negative impact on new tasks, a problem that is called negative transfer in deep learning research.

A further closer look at the cloning approach reveals that the source task of which the output head is cloned for initialization is often similar to the new target task (similarity in terms of plastic bricks with similar dimensions). This leads us to the assumption that similar tasks also form similar network weights during training. To test this, we extract the trained weights (including the bias weight) of all output heads after a fully trained sequence of 16 tasks and visualize them in a two-dimensional space using the dimensionality reduction

Table 2 Comparison of the approaches trained on 60 sequences with 57 training examples (100%, left)) and 25 training examples (45%, right) per task

	100% Training data		45% Training data	
Approach	Avg. new task loss	Avg. backward loss	Avg. new task loss	Avg. backward loss
From Scratch	0.036	–	0.069	–
Finetuning	0.031	0.196	0.033	0.35
MAS	0.044	0.054	0.093	0.099
MAS+Cloning	0.028	0.00002	0.039	0.0009

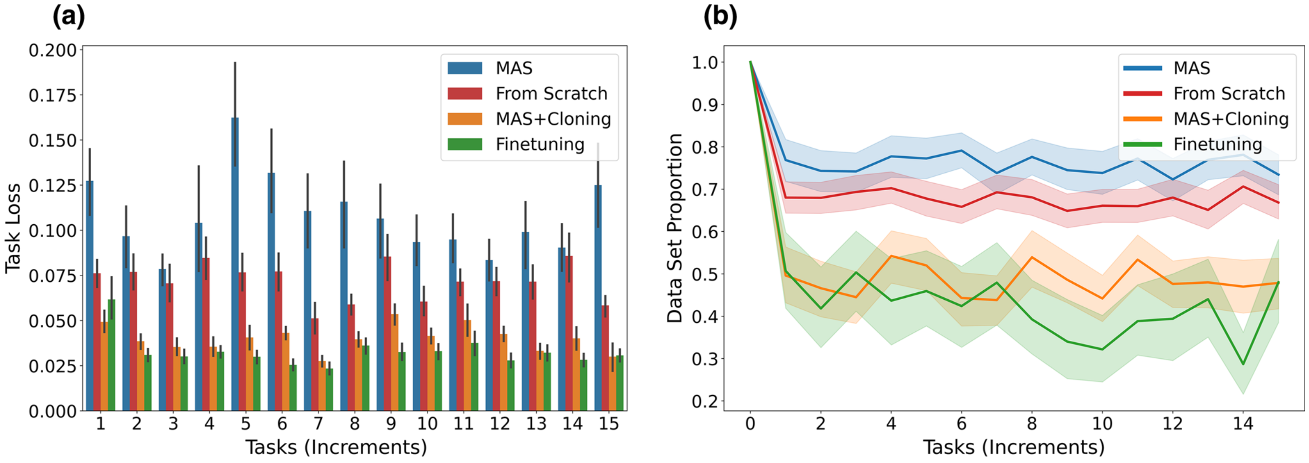

Fig. 6 **a** Average new task losses across all increments while training on only a subset of training data (45%). **b** Average proportion of training data required to reach at least 90% of the accuracy as with training on 100% training data (mean values with a 95% confidence interval across all experiments

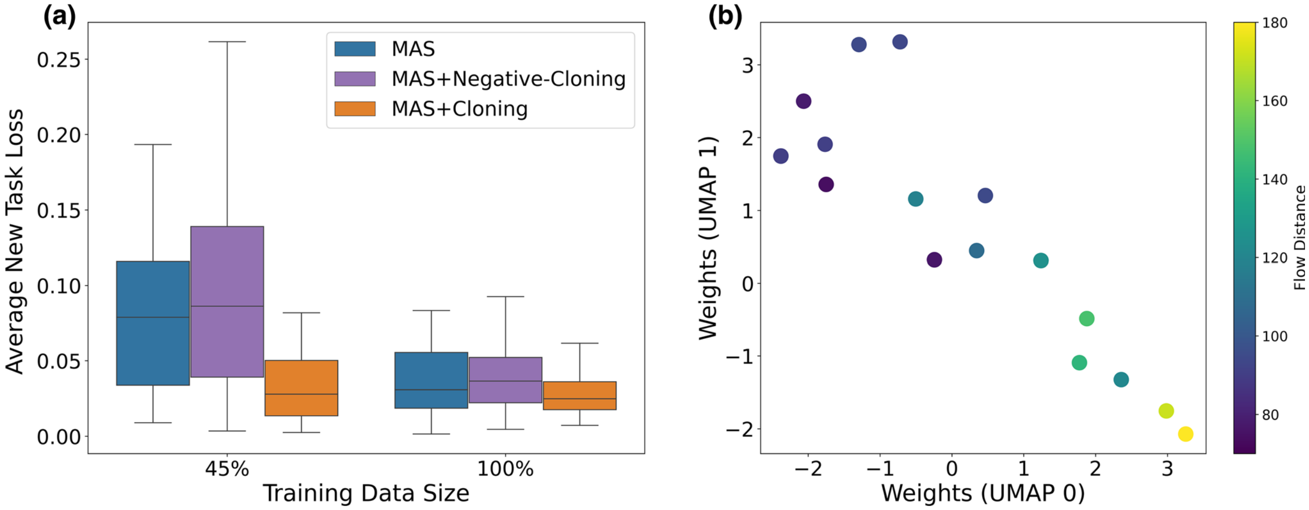

Fig. 7 **a** Comparison of the MAS, MAS-Cloning and MAS with negative cloning regarding the average new task losses across all increments, training on 45% (left) and on 100% training data. **b** MAS-Cloning: weights and bias of each task output head of the neural network, transformed into a two-dimensional space using UMAP and the cosine similarity function. The output heads are coloured according to the maximum flow distance of the corresponding task (i.e. plastic brick)

method UMAP (McInnes et al. 2018), see Fig. 7b. Each point in the plot accordingly represents one output head. For the representation of task similarity, we use the maximum flow distance of the plastic bricks. This parameter defines the distance the molten plastic must travel during molding and can be computed from the simulation for each brick. The figure shows that output heads for bricks that are more similar to each other are also closer to each other in the space and share similar network weights accordingly.

Conclusion and outlook

In this paper, we investigated a deep learning-based continual learning method for quality prediction across several different product variations in an injection molding use case. Our proposed approach extends the existing memory-aware synapses method by transferring pre-trained network weights. Thus, when learning a new product, the neural network is able to use network structures and weights already trained for previous products as well as train so far unused weights. Our extensive experiments showed that our approach can learn over multiple product variations without forgetting and does not require any data from previous tasks. In addition, it also performs better on sparse data than a trained model from scratch.

We also detected that the training of the network weights is related to the product characteristics. In future work, we will therefore investigate approaches to incorporate task-related information (such as product geometries) into the learning process. The goal here will be to develop a learning system that adapts its weights for new task variations based on their characteristics, thus increasing the systems generalizability and data efficiency. Furthermore, we will investigate the applicability of the proposed continual learning approach in other production use cases beyond injection molding. We expect that the findings and methods obtained are generalizable to other application fields.

Acknowledgements Funding was provided by Bergische Universität Wuppertal (Grant No. 11504).

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., & Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In: *Computer vision—ECCV 2018* (pp. 144–161). Cham: Springer. https://doi.org/10.1007/978-3-030-01219-9_9.

Aljundi, R., Kelchtermans, K., & Tuytelaars, T. (2019). Task-free continual learning. In: *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 11246–11255). https://doi.org/10.1109/CVPR.2019.01151.

Ayub, A., & Wagner, A. R. (2020). Tell me what this is: Few-shot incremental object learning by a robot. arXiv preprint arXiv:2008.00819.

Baumeister, T., Brunton, S. L., & Nathan Kutz, J. (2018). Deep learning and model predictive control for self-tuning mode-locked lasers. *Journal of the Optical Society of America B*, *35*(3), 617. https://doi.org/10.1364/JOSAB.35.000617.

Baweja, C., Glocker, B., & Kamnitsas, K. (2018). Towards continual learning in medical imaging. In *Medical imaging meets NIPS workshop, 32nd conference on neural information processing systems (NIPS)*.

Chen, Z., & Liu, B. (2018). *Lifelong machine learning, synthesis lectures on artificial intelligence and machine learning,* (2nd ed., Vol. 38). San Rafael: Morgan & Claypool Publishers.

Dehghan, M., Zhang, Z., Siam, M., Jin, J., Petrich, L., & Jagersand, M. (2019). Online object and task learning via human robot interaction. In *2019 international conference on robotics and automation (ICRA)* (pp. 2132–2138). IEEE. https://doi.org/10.1109/ICRA.2019.8794036.

Escobar, C. A., McGovern, M. E., & Morales-Menendez, R. (2021). Quality 4.0: a review of big data challenges in manufacturing. *Journal of Intelligent Manufacturing*, *2*(2), 15. https://doi.org/10.1007/s10845-021-01765-4.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville. A., & Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks.

Hsu, C. Y., & Liu, W. C. (2021). Multiple time-series convolutional neural network for fault detection and diagnosis and empirical study in semiconductor manufacturing. *Journal of Intelligent Manufacturing*, *32*(3), 823–836. https://doi.org/10.1007/s10845-020-01591-0.

Hsu, Y. C., Liu, Y. C., Ramasamy, A., & Kira, Z. (2018). Re-evaluating continual learning scenarios: A categorization and case for strong baselines. arXiv preprint arXiv:1810.12488.

Kashyap, S., & Datta, D. (2015). Process parameter optimization of plastic injection molding: A review. *International Journal of Plastics Technology*, *19*(1), 1–18. https://doi.org/10.1007/s12588-015-9115-2.

Kingma, D. P., & Ba, J. (2017) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526. https://doi.org/10.1073/pnas.1611835114.

Kiyasseh, D., Zhu, T., & Clifton, D. A. (2020). Clops: Continual learning of physiological signals. arXiv preprint arXiv:2004.09578.

Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., & Díaz-Rodríguez, N. (2020). Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, *58*, 52–68. https://doi.org/10.1016/j.inffus.2019.12.004.

Li, X., Jia, X., Yang, Q., & Lee, J. (2020). Quality analysis in metal additive manufacturing with deep learning. *Journal of Intelligent Manufacturing*, *31*(8), 2003–2017. https://doi.org/10.1007/s10845-020-01549-2.

Lopez-Paz, D., & Ranzato, M. (2017). Gradient episodic memory for continual learning. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 6470–6479). Curran Associates Inc., Red Hook, NY, USA, NIPS'17.

Maschler, B., Vietz, H., Jazdi, N., & Weyrich, M. (2020). Continual learning of fault prediction for turbofan engines using deep learning with elastic weight consolidation. In *2020 25th IEEE international conference on emerging technologies and factory automation (ETFA)* (pp. 959–966). https://doi.org/10.1109/ETFA46521.2020.9211903.

Lenga, M., Schulz, H.,& Saalbach, A. (2020). Continual learning for domain adaptation in chest x-ray classification. *Medical Imaging with Deep Learning*, pp. 413–423.

McDonnell, M. D. T., Arnaldo, D., Pelletier, E., Grant-Jacob, J. A., Praeger, M., Karnakis, D., et al. (2021). Machine learning for multi-dimensional optimisation and predictive visualisation of laser machining. *Journal of Intelligent Manufacturing*, *32*(5), 1471–1483. https://doi.org/10.1007/s10845-020-01717-4.

McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.

Meyes, R., Donauer, J., Schmeing, A., & Meisen, T. (2019). A recurrent neural network architecture for failure prediction in deep drawing sensory time series data. *Procedia Manufacturing*, *34*, 789–797. https://doi.org/10.1016/j.promfg.2019.06.205.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191.

Parisi, G. I., Tani, J., Weber, C., & Wermter, S. (2017). Lifelong learning of human actions with deep neural network self-organization. *Neural Networks*, *96*, 137–149. https://doi.org/10.1016/j.neunet.2017.09.001.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, *113*, 54–71. https://doi.org/10.1016/j.neunet.2019.01.012.

Pomponi, J., Scardapane, S., Lomonaco, V., & Uncini, A. (2020). Efficient continual learning in neural networks with embedding regularization. *Neurocomputing*, *397*, 139–148. https://doi.org/10.1016/j.neucom.2020.01.093.

Pytorch. (2020). Deep learning framework. https://pytorch.org/, Retrieved 2020 Dec 01.

Rebuffi, S. A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 5533–5542). https://doi.org/10.1109/CVPR.2017.587.

Rus, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., & Hadsell, R. (2016). Progressive neural networks. arXiv preprint arXiv:1606.04671.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., & Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. In: Dy, J., & Krause, A. (Eds.), Proceedings of machine learning research, PMLR (Vol. 80, pp. 4528–4537).

Shin, H., Lee, JK., Kim, J., & Kim, J. (2017). Continual learning with deep generative replay. In *Proceedings of the 31st international conference on neural information processing systems, NIPS'17* (pp. 2994–3003). Curran Associates Inc., Red Hook, NY, USA.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning. In: Kůrková V., Manolopoulos, Y., Hammer, B., Iliadis, L., & Maglogiannis, I. (Eds.), *Artificial neural networks and machine learning—ICANN 2018* (pp. 270–279). Cham: Springer. https://doi.org/10.1007/978-3-030-01424-7_27.

Tercan, H., Guajardo, A., Heinisch, J., Thiele, T., Hopmann, C., & Meisen, T. (2018). Transfer-learning: Bridging the gap between real and simulation data for machine learning in injection molding. *Procedia CIRP*, *72*, 185–190. https://doi.org/10.1016/j.procir.2018.03.087.

Tercan, H., Guajardo, A., & Meisen, T. (2019). Industrial transfer learning: Boosting machine learning in production. In *Proceedings, 2019 IEEE 17th international conference on industrial informatics (INDIN)* (pp. 274–279). IEEE. https://doi.org/10.1109/INDIN41052.2019.8972099.

Tian, H., Ren, D., Li, K., & Zhao, Z. (2021). An adaptive update model based on improved long short term memory for online prediction of vibration signal. *Journal of Intelligent Manufacturing*, *32*(1), 37–49. https://doi.org/10.1007/s10845-020-01556-3.

van Garderen, K., van der Voort, S., Incekara, F., Smits, M., & Klein, S. (2019) Towards continuous learning for glioma segmentation with elastic weight consolidation. In *International conference on medical imaging with deep learning*

Wang, J., Ma, Y., Zhang, L., Gao, R. X., & Wu, D. (2018). Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems*, *48*, 144–156. https://doi.org/10.1016/j.jmsy.2018.01.003.

Wong, J. M. (2016). Towards lifelong self-supervision: A deep learning direction for robotics. arXiv preprint arXiv:1611.00201.

Yuan, X., Li, L., Wang, Y., Yang, C., & Gui, W. (2020). Deep learning for quality prediction of nonlinear dynamic processes with variable attention-based long short-term memory network. *The Canadian Journal of Chemical Engineering*, *98*(6), 1377–1389. https://doi.org/10.1002/cjce.23665.

Zellinger, W., Grubinger, T., Zwick, M., Lughofer, E., Schöner, H., Natschläger, T., et al. (2020). Multi-source transfer learning of time series in cyclical manufacturing. *Journal of Intelligent Manufacturing*, *31*(3), 777–787. https://doi.org/10.1007/s10845-019-01499-4.

Zhao, K., Jiang, H., Wu, Z., & Lu, T. (2020). A novel transfer learning fault diagnosis method based on manifold embedded distribution alignment with a little labeled data. *Journal of Intelligent Manufacturing*, *152*(2), 107393. https://doi.org/10.1007/s10845-020-01657-z.

Zhizhong, L., & Hoiem, D. (2018). Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(12), 2935–2947. https://doi.org/10.1109/TPAMI.2017.2773081.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.