Catastrophic forgetting: still a problem for DNNs

B. Pfülb, A. Gepperth, S. Abdullah, and A. Kilian

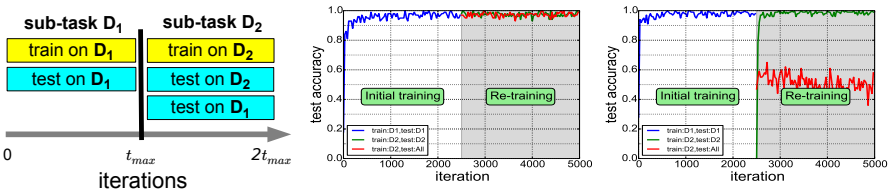
Fulda University of Applied Sciences, Leipzigerstr. 123, 36037 Fulda, Germany
{benedikt.pfuelb, alexander.gepperth, saad.abdullah, andre.kilian}@cs.hs-fulda.de
https://www.hs-fulda.de

Abstract. We investigate the performance of deep neural networks when trained on class-incremental visual problems consisting of initial training, followed by retraining with added visual classes. Catastrophic forgetting (CF) behavior is measured using a new evaluation procedure that aims at an application-oriented view of incremental learning. In particular, it imposes that model selection must be performed on the initial dataset alone, as well as demanding that retraining control be performed only using the retraining dataset, as initial dataset is usually too large to be kept. Experiments are conducted on class-incremental problems derived from MNIST, using a variety of different DNN models, some of them recently proposed to avoid catastrophic forgetting. When comparing our new evaluation procedure to previous approaches for assessing CF, we find their findings are completely negated, and that none of the tested methods can avoid CF in all experiments. This stresses the importance of a realistic empirical measurement procedure for catastrophic forgetting, and the need for further research in incremental learning for DNNs.

Keywords: DNN · catastrophic forgetting · incremental learning

1 Introduction

The context of this article is the susceptibility of DNN to an effect usually termed "catastrophic forgetting" or "catastrophic interference" [2]. When training a DNN incrementally, that is, first training it on a sub-task D_1 and subsequently retraining on another sub-task D_2 whose statistics differ (see Fig. 1), CF implies an abrupt and virtually complete loss of knowledge about D_1 during retraining. In various forms, knowledge of this effect dates back to very early works on neural networks [2], of which modern DNNs are a special case. Nevertheless, known solutions seem difficult to apply to modern DNNs trained in a purely gradient-based fashion. Recently, several approaches have been published with the explicit goal of resolving the CF issue for DNNs in incremental learning tasks, illustrated in [3,5,10]. On the other hand, there are "shallow" machine learning methods explicitly constructed to avoid CF (reviewed in, e.g., [9]), although this ability seems to be achieved at the cost of significantly reduced learning capacity. In this article, we test the recently proposed solutions for DNNs using



(a) Training Scheme (b) without CF (c) with CF

Fig. 1: Scheme of incremental training experiments (see Fig. 1a) and representative outcomes without and with CF (see Figs. 1b and 1c). Initial training with sub-task D_1 for t_{max} iterations is followed by retraining on sub-task D_2 for another t_{max} iterations. During training (white background) and retraining (grey background), test performance is measured on D_1 (blue curves), D_2 (green curves) and $D_1 \cup D_2$ (red curves). The red curves allow to determine the presence of CF by simple visual inspection: if there is significant degradation w.r.t. the blue curves, then CF has occurred.

a variety of class-incremental visual problems constructed from the well-known MNIST benchmark [6]. In particular, we propose a new experimental protocol to measure CF which avoids commonly made [3,5,10,7] implicit assumptions that are incompatible with incremental learning in applied scenarios.

1.1 Application relevance of catastrophic forgetting

When DNNs are trained on a single (sub-)task D_1 only, catastrophic forgetting is not an issue. When retraining is necessary with a new sub-task D_2, one often recurs to retraining the DNN with all samples from D_1 and D_2 together. This heuristic works in many situations, especially when the cardinality of D_1 is moderate. When D_1 becomes very large, however, or many slight additions $D_{(1+n)}$ are required, this strategy becomes unfeasible, and an incremental training scheme (see Fig. 1a) must be used. Thus, the issue of catastrophic forgetting becomes critically important, which is why we wish to assess, once and for all, where DNNs stand with respect to CF.

1.2 Approach of the article

In all experiments, we consider class-incremental learning scenarios divided into two training steps on disjunct sub-tasks D_1 and D_2, as outlined in Sect. 1 and visualized in Fig. 1. Both training steps are conducted for a fixed number of iterations, with the understanding that in practice retraining would have to be stopped at some point by an appropriate criterion before forgetting of D_1 is complete. The occurrence of forgetting is quantified using classification performance on all test samples from $D_1 \cup D_2$ at the time retraining is stopped (see Fig. 1 for a visual impression). In contrast to previous works, our experiments take into account how (class-)incremental learning works in practice:

- D_2 is not available at initial training
- D_1 is not available at retraining time as it might be very large.

Tab. 1: Overview over 6 DNN models used in this study. They are obtained by combining the concept of Dropout (D) with the basic DNN models: fully-connected (fc), convolutional (conv), LWTA and EWC.

concept \ model	fc	conv	LWTA	EWC
with Dropout	D-fc	D-conv	✗	D-EWC (EWC)
without Dropout	fc	conv	LWTA-fc (LWTA)	✗

This training paradigm (which we term "realistic") has profound consequences, most importantly that initial model selection has to be performed using D_1 alone, which is in contrast to previous works on CF in DNNs [3,5,10], where $D_1 \cup D_2$ is used for model selection purposes. Another consequence is that the decision on when to stop retraining has to be taken based on D_2 alone.
In order to reproduce earlier results, we introduce another training paradigm which we term "prescient", where both D_1 and D_2 are known at all times, and which aligns well with evaluation methods in recent works. As classifiers, we use typical DNN models like fully-connected- (fc), convolutional- (conv), LWTA-based- (fc-LWTA) and DNNs based on the EWC model (EWC). Most of these can be combined with the concepts of Dropout (D, [4]). An overview of possible combinations is given in Tab. 1.
For all models, hyperparameter optimization is conducted in order to ensure that our results are not simply accidental.

1.3 Related work on CF in DNNs

In addition to early works on CF in connectionist models [2], new approaches specific to DNNs have recently been unveiled, some with the explicit goal of preventing catastrophic forgetting [3,5,10,7]. The work presented in [3] advocates the popular Dropout method as a means to reduce or eliminate CF, validating their claims on tasks derived a randomly shuffled version of MNIST [6] and a Sentiment Analysis problem. In [10], a new kind of competitive transfer function is presented which is termed LWTA (Local Winner Takes All). In a very recent article [5], the authors advocate determining the hidden layer weights that are most "relevant" to a DNNs performance, and punishing the change of those weights more heavily during retraining by an additional term in the energy functional. Experiments are conducted on random data, randomly shuffled MNIST data as in [3,10], and on a task derived from Deep Q-learning in Atari Games [8]. Even more recently, authors in [7] propose the so-called incremental moment matching (IMM) technique which suggests an alignment of statistical properties of the DNN between D_1 and D_2 which is not included here, because it inherently requires knowledge of D_1 at re-training time to select the best regularization parameter(s).

2 Methods

The principal dataset this investigation is based on is MNIST [6]. Despite being a very old benchmark, and a very simple one, it is still widely used, in particular in

recent works on incremental learning in DNNs [3,5,7,10]. It is used here because we wish to reproduce these results, and also because we care about performance in class-incremental settings, not offline performance on the whole dataset. As we will see, MNIST-derived problems are more than a sufficient challenge for the tested algorithms, so it is really unnecessary to add more complex ones (but see Sect. 4 for a more in-depth discussion of this issue).

2.1 Learning tasks

As outlined in Sect. 1.2, incremental learning performance of a given model is evaluated on several datasets constructed from the MNIST dataset. The model is trained successively on two sub-tasks (D_1 and D_2) from the chosen dataset and it is recorded to what extend knowledge about previous sub-tasks is retained. The precise way the sub-tasks of all datasets are constructed from the MNIST dataset shall be described below.

Exclusion: D5-5 These datasets are obtained by randomly choosing 5 MNIST classes for D_1, and the remaining 5 for D_2. To verify that results do not depend on a particular choice of classes, we create a total of 8 datasets where the partitioning of classes is different (see Tab. 2).

Exclusion: D9-1 We construct these datasets in a similar way as D5-5, selecting 9 MNIST classes for D_1 and the remaining class for D_2. In order to make sure that no artifacts are introduced, we create three datasets (D9-1a, D9-1b and D9-1c) with different choices for D_1 and D_2, see Tab. 2.

Permutation: DP10-10 This is the dataset used to evaluate incremental retraining in [3,5,10], so results can directly be compared. It contains two sub-tasks, each of which is obtained by permuting each 28 x 28 image in a random fashion that is different between, but identical within, sub-tasks. Since both sub-tasks contain 10 MNIST classes, we denote this dataset by DP10-10, the "P" indicating permutation, see Tab. 2.

2.2 Models

We use TensorFlow/Python to implement or re-create all models used in this article. The source code for all experiments is available at `https://gitlab.informatik.hs-fulda.de/ML-Projects/CF_in_DNNs`.

Fully connected deep network Here, we consider a "normal" fully-connected (FC) feed-forward MLP with two hidden layers, a softmax (SM) readout layer trained using cross-entropy, and the (optional) application of Dropout (D) and

Tab. 2: MNIST-derived datasets (DS) used in this article. All partitions of MNIST into D_1 and D_2 are non-overlapping. For the DP10-10 dataset, the classes are identical for D_1 and D_2 but pixels are permuted in D_2 as described in the text.

DS / part.	D5-5								D9-1			DP10-10
	D5-5a	D5-5b	D5-5c	D5-5d	D5-5e	D5-5f	D5-5g	D5-h	D9-1a	D9-1b	D9-1c	
D_1 classes	0-4	02468	34689	02567	01345	03489	05678	02368	0-8	1-9	0,2-9	0-9
D_2 classes	5-9	13579	01257	13489	26789	12567	12349	14579	9	0	1	0-9

ReLU operations after each hidden layer. Its structure can thus be summarized as In-FC1-D-ReLU-FC2-D-ReLU-FC3-SM. In case more hidden layers are added, their structure is analogous.

ConvNet A convolutional network inspired by [1] is used here, with two hidden layers and the application of Dropout (D), max-pooling (MP) and ReLU after each layer, as well as a softmax (SM) readout layer trained using cross-entropy. It structure can thus be stated as In-C1-MP-D-ReLU-C2-MP-D-ReLU-FC3-SM.

EWC The Elastic Weight Consolidation (EWC) model has been recently proposed in [5] to address the issue of CF in incremental learning tasks. We use a TensorFlow-implementation provided by the authors that we integrate into our own experimental setup; the corresponding code is available for download as described. The basic network structure is analogous to that of fc models.

LWTA Deep learning with a fully-connected Locally-Winner-Takes-All (LWTA) transfer function has been proposed in [10], where it is also suggested that deep LWTA networks have a significant robustness when trained incrementally with several tasks. We use a self-coded TensorFlow implementation of the model proposed in [10]. Following [10], the number of LWTA blocks is always set to 2. The basic network structure is analogous to that of fully-connected models.

Dropout Dropout, introduced in [4] and widely used in recent research on DNNs, is a special transfer function that sets a random subset of activities in each layer to 0 during training. It can, in principle, be applied to any DNN and thus can be combined with all previously listed models except EWC (already incorporated) and LWTA (unclear whether this would be sensible as LWTA is already a kind of transfer function).

2.3 Experimental procedure

The procedure we employ for all experiments is essentially the one given in Sect. 1.2, where all models listed in Sect. 2.2 and Tab. 1 are applied to a subset of class-incremental learning tasks described in Sect. 2.1. For each experiment, characterized by a pair of model and task, we conduct a search in model parameter space for the best model configuration, leading to multiple runs per experiment, each run corresponding to a particular set of parameters for a given model and a given task.

Each run lasts for $2t_{\max}$ iterations and is structured as shown in Fig. 1, initially training the chosen model first on sub-task D_1 and subsequently on sub-task D_2, each time for $t_{\max}$ iterations. Classification accuracy, measured at iteration t, on a test set $\mathcal{B}$ while training on a train set $\mathcal{A}$, is denoted $\chi(\mathcal{A}, \mathcal{B}, t)$. For a thorough evaluation, we record the quantities $\chi(D_1, D_1, t < t_{\max})$, $\chi(D_2, D_2, t \geq t_{\max})$ and $\chi(D_2, D_1 \cup D_2, t \geq t_{\max})$. Finally, the best-suited parameterized model must be chosen among all the runs of an experiment. We investigate two strategies for doing this, corresponding to different levels of knowledge at training and retraining time during a single run. As detailed in Sect. 1.2, these are the strategies which we term "prescient" and "realistic". The "prescient" evaluation strategy (see Alg. 1) corresponds to an a priori knowledge of sub-task D_2 at initial training time, as well as to a knowledge about D_1 at retraining time.

Both assumptions are difficult to reconcile with incremental training in applied scenarios, as detailed in Sect. 1.2. We use this strategy here to compare our results to previous works in the field [3,5,10]. In contrast, the "realistic" evaluation strategy (see Alg. 2) assumes no knowledge about future sub-tasks (D_2) and furthermore supposes that D_1 is unavailable at retraining time due to its size (see Sect. 1.2 for the reasoning). It is this strategy which we propose for future investigations concerning incremental learning.

2.4 Hyperparameters and model selection

For runs from all experiments, not involving CNNs, the parameters that are varied are: number of hidden layers $L \in \{2, 3\}$, hidden layer sizes $S \in \{200, 400, 800\}$, learning rate during initial training $\epsilon_{D_1} \in \{0.01, 0.001\}$, and learning rate during retraining $\epsilon_{D_2} \in \{0.001, 0.0001, 0.00001\}$. Based on the parameter set $\mathcal{P} \subseteq L \times S \times \epsilon_{D_1} \times \epsilon_{D_2}$, all models are evaluated, respectively are model-specific hyperparameters used or supplanted. For experiments using CNNs, we fix the topology to a form known to achieve good performances on MNIST as an exhaustive optimization of all relevant parameters would prove too time-consuming in this case, and vary only the ϵ_{D_1} and ϵ_{D_2} as detailed before. For EWC experiments, the importance parameter λ of the retraining run is fixed at $1/\epsilon_{D_2}$, this choice is nowhere to be found in [5] but is used in the provided code, which is why we adopt it. For LWTA experiments, the number of LWTA blocks is fixed to 2 in all experiments, corresponding to the values used in [10]. Dropout rates, if applied, are set to 0.2 (input layer) and 0.5 (hidden layers), consistent with the choices made in [3]. For CNNs, only a single Dropout rate of 0.5 is applied for input and hidden layers alike. The length $t_{\max}$ of training / retraining period is empirically fixed to 2500 iterations, each iteration using a batch size of 100 ($\text{batch}_{\text{size}}$). The Momentum optimizer provided by TensorFlow is used for performing training, with a momentum parameter $\mu = 0.99$.

2.5 Reproduction of previous results by prescient evaluation

In this experiment, we wish to determine whether it is possible to find a parameterization for a given DNN model and task when there is a perfect knowledge about and availability of the initial and future sub-tasks. Applying the models listed in Sect. 2.2 to the tasks described in Sect. 2.1, and using the experimental procedure detailed in Sect. 2.3, we obtain the results summarized in Tab. 3 (applying the "prescient" evaluation of Alg. 1). We can state the following insights: first of all, we can reproduce the basic results from [3] using the fc model on DP10-10, which avoids catastrophic forgetting (contrarily to the conclusions drawn in this paper: these authors consider the very modest decrease in performance to be catastrophic forgetting). This is however very specific to this particular task, and in fact all models except EWC exhibit blatant catastrophic forgetting behavior particularly on the D5-5 type tasks, while performing adequately if not perfectly on the D9-1 tasks. EWC performs well on these tasks as well, so we can state that EWC is the only tested algorithm that avoids CF

for all tasks when using prescient evaluation. Another observation is that the use of Dropout, as suggested in [3], does not seem to significantly improve matters. The LWTA method performs a little better than fc, D-fc, conv and D-conv but is surpassed by EWC by a very large margin.

2.6 Realistic evaluation

This experiment imposes the much more restrictive/realistic evaluation, detailed in Sect. 2.3 and Alg. 2, essentially performing initial training and model selection only on D_1 and retraining only using D_2. It is this or related schemes that would have to be used in typical application scenarios, and thus represents the principal subject of this article. The performances of all tested DNN models on all of the tasks from Sect. 2.1 are summarized in Tab. 4. Plots of experimental results over time for the D-fc and EWC models are given in Figs. 2 to 5. The results show a rather bleak picture where only the EWC model achieves significant success for the D9-1 type tasks while failing for the D5-5 tasks. All other models do not even achieve this partial success and exhibit strong CF for all tasks. We can therefore observe that a different choice of evaluation procedure strongly impacts results and the conclusions which are drawn concerning CF in DNNs. For the realistic evaluation condition, which in our view is much more relevant than the prescient one used in nearly all of the related work on the subject, CF occurs for all DNN models we tested, and partly even for the EWC model. As to the question why EWC performs well for all of the D9-1 type task in contrast to the D5-5 type tasks, one might speculate that the addition of five new classes, as opposed to one, might exceed EWC's capabilities of protecting the weights most relevant to D_1. Various different values of the constant λ governing the contribution of Fisher information in EWC were tested but with very similar results.

3 Discussion of results and principal conclusions

From our experiments, we draw the following principal conclusions:

- CF should be investigated using the appropriate evaluation paradigms that reflect application conditions. At the very least, using future data for model selection is inappropriate, which leads to conclusions that are radically different from most related experimental work, see Sect. 1.3.

Data: model m, sub-tasks D_1 & D_2, parameter set $\mathcal{P}$
Result: quality of best model $q^*_{m_{\boldsymbol{p}}}$
initialize $q^*_{m_{\boldsymbol{p}}} \leftarrow -1$
foreach *parameters* $\boldsymbol{p} \in \mathcal{P}$ **do**
  initial training of $m_{\boldsymbol{p}}$ on D_1 for $t_{\max}$ iterations
  for $t \leftarrow 0$ **to** t_{max} *iterations* **do** // *retraining of* $m_{\boldsymbol{p}}$ *on* D_2
    update $m_{\boldsymbol{p}}$ on D_2 using batch$_{\text{size}}$; $q_{m_{\boldsymbol{p}},t} \leftarrow \chi(D_2, D_1 \cup D_2, t)$
    if $q_{m_{\boldsymbol{p}},t} > q^*_{m_{\boldsymbol{p}}}$ **then** $q^*_{m_{\boldsymbol{p}}} \leftarrow q_{m_{\boldsymbol{p}},t}$
return $q^*_{m_{\boldsymbol{p}}}$

Alg. 1: The *prescient evaluation* strategy.

Tab. 3: Results for **prescient evaluation**. Please note that the performance level of complete catastrophic forgetting (i.e., chance-level classification after retraining with D_2) depends on the dataset considered: for the D5-5 dataset it is at 0.5, whereas it is at 0.1 for the D9-1 datasets. The rightmost column indicates the DP10-10 task which is solved near-perfectly by all models.

dataset / model	D5-5								D9-1			DP10-10
	D5-5a	D5-5b	D5-5c	D5-5d	D5-5e	D5-5f	D5-5g	D5-5h	D9-1a	D9-1b	D9-1c	
EWC	0.92	0.92	0.91	0.93	0.94	0.94	0.89	0.93	1.00	1.00	1.00	1.00
fc	0.69	0.63	0.58	0.65	0.61	0.58	0.61	0.69	0.87	0.87	0.86	0.97
D-fc	0.58	0.60	0.61	0.66	0.61	0.54	0.63	0.64	0.87	0.87	0.85	0.96
conv	0.51	0.50	0.50	0.50	0.50	0.50	0.51	0.49	0.89	0.89	0.87	0.95
D-conv	0.51	0.50	0.50	0.50	0.50	0.50	0.50	0.49	0.81	0.84	0.87	0.96
LWTA	0.66	0.68	0.64	0.73	0.71	0.62	0.68	0.71	0.88	0.91	0.91	0.97

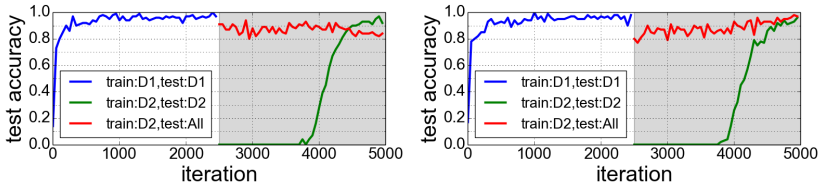
Data: model m, sub-tasks D_1 & D_2, parameter set $\mathcal{P}$
Result: quality of best model $q^*_{m_{\boldsymbol{p}}}$
initialize $q^*_T \leftarrow -1$
forall the *parameters* $\boldsymbol{p} \in \mathcal{P}$ **do** *//determine best model parameter training* D_1
  for $t \leftarrow 0$ **to** t_{max} *iterations* **do**
    update of $m_{\boldsymbol{p}}$ on D_1 using batch$_{\text{size}}$; $q_{m_{\boldsymbol{p}},t} \leftarrow \chi(D_1, D_1, t)$
    if $q_{m_{\boldsymbol{p}},t} > q^*_T$ **then** $q^*_T \leftarrow q_{m_{\boldsymbol{p}},t}$; $m^*_{\boldsymbol{p}} \leftarrow m_{\boldsymbol{p}}$
initialize $q^*_{m_{\boldsymbol{p}}} \leftarrow -1$
forall the *retraining learning rates* $\epsilon \in \epsilon_{D_2}$ **do**
  initialize $q^*_R \leftarrow -1$
  for $t \leftarrow 0$ **to** t_{max} *iterations* **do** *//retraining of* $m^*_{\boldsymbol{p}}$ *on* D_2
    update $m^*_{\boldsymbol{p}}$ on D_2 with learning rate ϵ; $q_{m_{\boldsymbol{p}},t} \leftarrow \chi(D_2, D_2, t)$
    if $q_{m_{\boldsymbol{p}},t} > q^*_R$ **then** $q^*_R \leftarrow q_{m_{\boldsymbol{p}},t}$
  $t_E \leftarrow \arg\min_t(q_{m_{\boldsymbol{p}},t} > 0.99 \cdot q^*_R)$; $q_{m_{\boldsymbol{p}}} \leftarrow \chi(D_2, D_1 \cup D_2, t_E)$ **if** $q_{m_{\boldsymbol{p}}} > q^*_{m_{\boldsymbol{p}}}$ **then** $q^*_{m_{\boldsymbol{p}}} \leftarrow q_{m_{\boldsymbol{p}}}$
return $q^*_{m_{\boldsymbol{p}}}$

Alg. 2: The *realistic evaluation* strategy.

- using a realistic evaluation paradigm, we find that CF is still very much a problem for all investigated methods.
- in particular: Dropout is not effective against CF; neither is LWTA.
- the permuted MNIST task can be solved by almost any DNN model in almost any topology. So all conclusions drawn from using this task should be revisited.

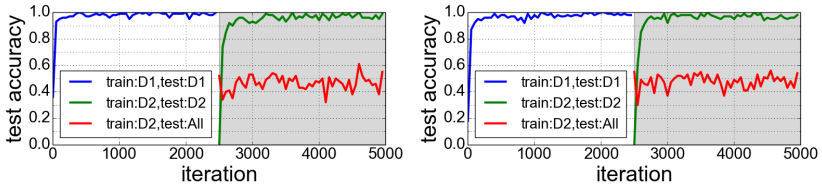
Tab. 4: Results for **realistic evaluation**. Please note that the performance level of total catastrophic forgetting (i.e., chance-level classification after retraining with D_2) depends on the dataset: for the D5-5 dataset it is at 0.5, whereas it is at 0.1 for the D9-1 datasets. The rightmost column indicates the DP10-10 task ("permuted MNIST") which is again solved near-perfectly by all models.

dataset / model	D5-5								D9-1			DP10-10
	D5-5a	D5-5b	D5-5c	D5-5d	D5-5e	D5-5f	D5-5g	D5-5h	D9-1a	D9-1b	D9-1c	
EWC	0.48	0.56	0.62	0.52	0.58	0.58	0.55	0.53	0.82	0.91	0.97	0.99
fc	0.47	0.49	0.50	0.50	0.48	0.49	0.50	0.49	0.15	0.10	0.23	0.97
D-fc	0.47	0.50	0.50	0.50	0.49	0.49	0.50	0.49	0.52	0.10	0.16	0.96
conv	0.48	0.50	0.50	0.50	0.49	0.50	0.51	0.49	0.29	0.33	0.11	0.95
D-conv	0.48	0.50	0.50	0.50	0.45	0.50	0.50	0.49	0.24	0.22	0.14	0.96
LWTA	0.47	0.50	0.50	0.50	0.49	0.49	0.51	0.49	0.48	0.29	0.66	0.97



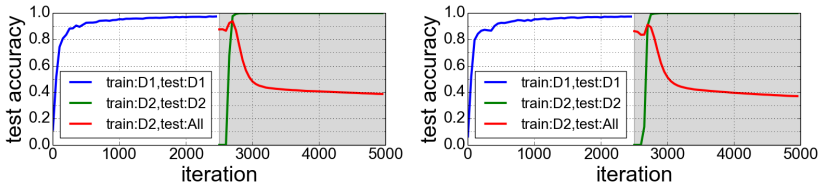
Task: D9-1a Task: D9-1c

Fig. 2: Best EWC runs on D9-1 datasets in the **realistic evaluation** condition.



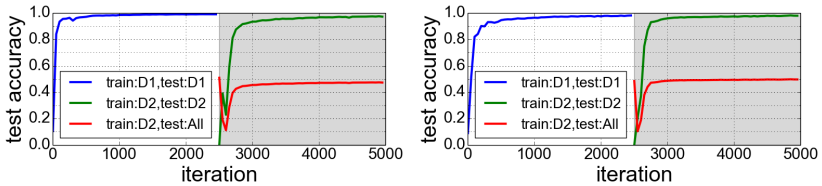
Task: D5-5a Task: D5-5h

Fig. 3: Best EWC runs on D5-5 datasets in the **realistic evaluation** condition.



Task: D9-1b Task: D9-1c

Fig. 4: Best D-fc runs on D9-1 datasets in the **realistic evaluation** condition.



Task: D5-5a Task: D5-5c

Fig. 5: Best D-fc runs on D5-5 datasets in the **realistic evaluation** condition.

– EWC seems to be partly effective but fails for all of the D5-5 tasks, indicating that it is not the last word in this matter.

We write that EWC "seems to be partly effective", meaning it solves some incremental tasks well while it fails for others. So we observe that there is no guarantee that can be obtained from a purely empirical validation approach such as ours; yet another type of incremental learning task might be solved perfectly or not at all. This points to the principal conceptual problem that we see when investigating CF in DNNs: there is no theory that might offer any guarantees. Such guarantees could be very useful in practice, the most interesting one being how to determine a lower bound on performance loss on $D_1 \cup D_2$, without having

access to D_1, only to the network state and D_2. Other guarantees could provide upper bounds on retraining time before performance on $D_1 \cup D_2$ degrades.

4 Future work

The issue of CF is a complex one, and correspondingly our article and our experimental procedures are complex as well. There are several points where we made rather arbitrary choices, e.g., when choosing the constant $\mu = 0.99$ in the realistic evaluation Alg. 2. The results are affected by this choice although we verified that the trend is unchanged. Another weak point is our model selection procedure: a much larger combinatorial set of model hyper-parameters should be sampled, including Dropout rates, convolution filter kernels, number and size of layers. This might conceivably allow to identify model hyperparameters avoiding CF for some or all tested models, although we consider this unlikely. Lastly, the use of MNIST might be criticized as being too simple: this is correct, and we are currently doing experiments with more complex classification tasks (e.g., SVHN and CIFAR-10). However, as our conclusion is that none of the currently proposed DNN models can avoid CF, this is not very likely to change when using an even more challenging classification task (rather the reverse, in fact).

References

1. Ciresan, D.C., Meier, U., Masci, J., Maria Gambardella, L., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. vol. 22, p. 1237. Barcelona, Spain (2011)
2. French, R.: Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences (4) (1999)
3. Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A., Bengio, Y.: An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211 (2013)
4. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)
5. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences p. 201611835 (2017)
6. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Intelligent Signal Processing. IEEE Press (2001)
7. Lee, S.W., Kim, J.H., Jun, J., Ha, J.W., Zhang, B.T.: Overcoming catastrophic forgetting by incremental moment matching. In: Advances in Neural Information Processing Systems. pp. 4655–4665 (2017)
8. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature (7540), 529–533 (2015)

9. Sigaud, O., Salaün, C., Padois, V.: On-line regression algorithms for learning mechanical models of robots: a survey. Robotics and Autonomous Systems **59**(12), 1115–1129 (2011)
10. Srivastava, R.K., Masci, J., Kazerounian, S., Gomez, F., Schmidhuber, J.: Compete to compute. In: Advances in neural information processing systems (2013)