# Incremental learning with deep neural networks using a test-time oracle

Alexander Gepperth[1] and Ayanava Sarkar[1]

1- University of Applied Sciences Fulda - Dept of Applied Computer Science
Leipzigerstr. 123, 36037 Fulda - Germany

**Abstract**. We present a simple idea to avoid catastrophic forgetting when training deep neural networks (DNNs) on class-incremental tasks. This means that initial training is conducted on a sub-task described by a dataset $D1$, whereas re-training is conducted subsequently, on a sub-task described by a dataset $D2$ that is composed of different classes. As our recent work suggest that DNNs perform very poorly at this problem, we propose a simple extension that proposes an individually trained read-out layer for each sub-task. While this is unproblematic for training, a clustering method is used at test time to determine to which sub-task a sample most likely belongs. Experiments on simple benchmarks derived from MNIST show the effectiveness of this method for which a dedicated TensorFlow implementation is made available.

## 1 Introduction

The context of this article is the susceptibility of deep neural networks (DNN) to an effect usually termed "catastrophic forgetting" or "catastrophic interference" [1]. When training a DNN incrementally, that is, first training it on a sub-task

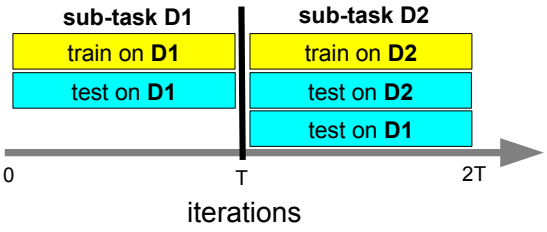


Fig. 1: Basic scheme of incremental training experiments conducted in this article. Initial training is conducted using a sub-task $D1$ for $T$ iterations, followed by retraining on sub-task $D2$ for another $T$ iterations. Both datasets differ in their statistical properties; in this article, we model this by using different classes from the MNIST benchmark for $D1$ and $D2$, or a different spatial permutation of pixels from the same classes, or both. During both training and retraining, performance on the test sets of $D1$ and $D2$ (derived from the MNIST test data) are conducted.

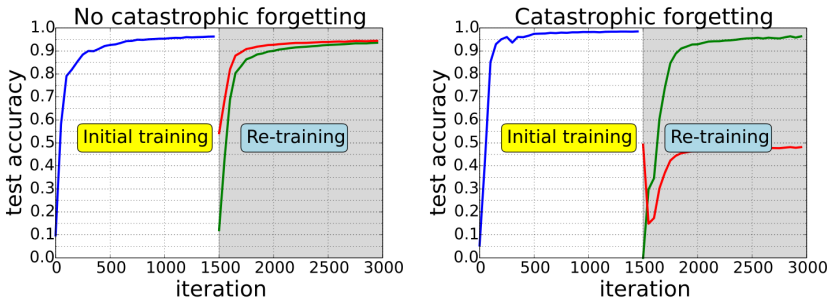


Fig. 2: Two prototypical experiments without (upper diagram) and with (lower diagram) catastrophic forgetting. Each experiment is subdivided into two steps: initial training on sub-task $D1$ (left-hand side, white background) and re-training on sub-task $D2$ (right-hand side, gray background). The dark blue curve indicates test performance on $D1$ during initial training, the red one test performance on $D1 \cup D2$ during re-training. It is the latter that indicates unambiguously whether catastrophic forgetting happens. The green curve indicates test performance on $D2$ during re-training and indicates whether $D2$ has been well learned or not.

$D1$ and subsequently re-training on another sub-task $D2$ whose statistics differ (see Fig. 1 for a visual impression), catastrophic forgetting (CF) implies an abrupt and virtually complete loss of knowledge about $D1$ during re-training. In various forms, knowledge of this effect dates back to very early works on neural networks [1], 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, namely [2, 3, 4]. On the other hand, there are machine learning methods explicitly constructed to avoid catastrophic forgetting [5, 6, 7], although this ability seems to be achieved at the cost of significantly reduced learning capacity, probably because none of these architectures is "deep". In this article, we verify the proposed solutions for DNNs using a wide variety of class-incremental visual problems constructed from the well-known MNIST benchmark [8]. subsectionRelevance of the catastrophic forgetting problem When DNNs are trained on a single (sub-)task $D1$ only, catastrophic forgetting is not an issue. When retraining is necessary with a new sub-task $D2$, one recurs often to the solution to retrain the DNN with samples defining $D1$, plus the samples for $D2$. This heuristic works in many situations, especially when the cardinality of $D1$ is moderate. When $D1$ becomes very large, and many slight additions $D(1+n)$ are required, this strategy is very ineffective or outright infeasible. In addition, 'Big Data' settings in which streaming data are processed (like, e.g., in network intrusion detection[9]) are often subject to concept drift [10] and do not reveal whether incoming data is from a "new" task (with different statistics) or from $D1$. In this case, con-

tinuous retraining is required, and many settings require that old knowledge be retained, at least for a defined time. If DNNs are employed in cases as outlined here, the issue of catastrophic forgetting becomes critically important, which is why we wish to asses, once and for all, where DNNs stand w.r.t. this issue.

*Related work* Recently, new approaches specific to DNNs have been unveiled[2, 3, 4], some with the explicit goal of preventing catastrophic forgetting[2, 4], while others [3] just suggest that their methods induce a greater "structural stability" while carefully avoiding the term "catastrophic forgetting". The work presented in [2] advocates the popular "dropout" method as a means to reduce or eliminate catastrophic forgetting, validating their claims on tasks derived a randomly shuffled version of MNIST[8] and a Sentiment Analysis problem. In [3], a new kind of competitive transfer function is presented which is termed LWTA ("local winner takes all"). This article also remarks that most forgetting happens in the readout layers of a DNN, and that maintaining separate readout layers for each sub-task can alleviate catastrophic forgetting. This idea is generalized in this article to the concept of multiple readout layers (MRL) in DNNs. Again, tests are conducted on two aforementioned problems. Lastly, in a very recent article [4] 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 re-training by an additional term in the energy functional. Experiments are conducted on random data, randomly shuffled MNIST data as in [2, 3], and on a task derived from Deep Q-learning in Atari Games [11].

## 2 Methods

## 3 Datasets

The principal dataset this investigation is based on is MNIST[8]. 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 deep networks[2, 3, 4]. 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 algorithm so it is really unnecessary to add more complex ones.

#### *3.0.1 Permutation: DP10-10*

This is the dataset used to evaluate incremental retraining in [2, 3, 4], so results can directly be compared to those in [2, 3, 4]. It contains two sub-problems, each of which is obtained by permuting each 28x28 image in a random fashion that is different between, but identical within, sub-problems. Since both sub-problems contain 10 MNIST classes, we denote this dataset by DP10-10, the 'P' indicating permutation.

#### *3.0.2 Exclusion: D5-5*

This dataset contains two sub-problems that are obtained by randomly choosing 5 MNIST classes for the first sub-problem, and the remaining 5 for the second, which leads to the identifier D5-5. For simplicity, we choose the classes as 0,1,2,3,4 and 5,6,7,8,9. To verify that results do not depend on this particular choice of classes, we create two additional datasets where the partitioning of classes is 0,2,4,6,8 –vs– 1,3,5,7,9 (D5-5b) and 3,4,6,8,9 –vs– 0,1,2,5,7 (D5-5c).

#### *3.0.3 Exclusion: D9-1*

We construct this dataset (containing two sub-problems) in a similar way as D5-5, selecting MNIST classes 0–8 for the first sub-problem and the remaining class 9 for the second one. In order to make sure that no artifacts are introduced by the arbitrary choice of the second sub-problem, we create two additional datasets (D9-1b and D9-1c) where the second sub-problem contains MNIST class 0 and 1, respectively.

### 3.1 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 www.gepperth.net/alexander/downloads/iclr18.tar
We mainly 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 ReLU operations after each hidden layer. Its structure can thus be summarized as Input-FC1-D-ReLU-FC2-D-ReLu-FC3-SM.

## 4 Experiments

## 5 Discussion

## References


[1] RM French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4), 1999.

[2] Ian J Goodfellow, Mehdi Mirza, Xia Da, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgeting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[3] Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318, 2013.

[4] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835, 2017.

[5] A Gepperth and C Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 2015. accepted.

[6] S. Vijayakumar and S. Schaal. Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high-dimensional spaces. In *International Conference on Machine Learning*, 2000.

[7] M. Butz, D. Goldberg, and P. Lanzi. Computational complexity of the xcs classifier system. *Foundations of Learning Classifier Systems*, 51, 2005.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[9] Leandro L Minku and Xin Yao. Ddd: A new ensemble approach for dealing with concept drift. *IEEE transactions on knowledge and data engineering*, 24(4):619–633, 2012.

[10] A Gepperth and B Hammer. Incremental learning algorithms and applications. 2016.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.