ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

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

Alexander Gepperth[1] and Saad Abdullah Gondal[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 readout layer for each sub-task. While this is unproblematic for training, a clustering method (the oracle) is used at test time to determine which sub-task a sample most likely belongs to. 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 avoidance of an effect usually termed "catastrophic forgetting" or "catastrophic interference" [2] in deep neural networks (DNNs) using the so-called multiple readout layer (MRL) technique. When training a DNN incrementally, that is, first training it on a sub-task $D1$ and subsequently re-training on another sub-task $D2$ whose statistics differ (see Fig. 2 for a visual impression), catastrophic forgetting (CF) implies an abrupt and virtually complete loss of knowledge about $D1$ during re-training. A common workaround is to retrain the DNN with samples defining $D1$, plus the samples for $D2$. This 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,

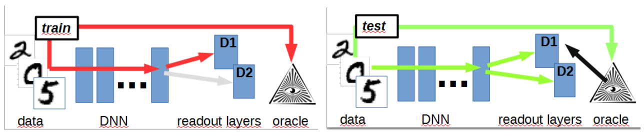

Fig. 1: Illustration of the proposed MRL technique. At training time (left), the DNN is trained normally, the readout layer to train being determined by the sub-task, while at the same time training the oracle. At test time (right), it is the (now fully trained) oracle that determines which readout layer is used to obtain a classification.

making the catastrophic forgetting issue critically important. This motivated us to look into simple ways DNNs could be augmented to perform what is required. We provide here a proof-of-concept for the proposed MRL technique (see Fig. 1) using a set of simple class-incremental visual problems constructed from the well-known MNIST benchmark [5].

Related work In various forms, knowledge of the catastrophic forgetting 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, new approaches specific to DNNs have been unveiled[3, 10, 4, 7, 8, 6, 9], some with the explicit goal of preventing catastrophic forgetting[3, 4, 7, 8, 6, 9], while others [10] just suggest that their methods induce a greater "structural stability".In [4] the authors advocate determining the hidden layer weights that are most "relevant", and punishing the change of those weights more heavily during re-training. In [8], newly trained filters are constrained to be linear combinations of existing ones, thus guaranteeing unimpaired performance on the original problem. The incremental moment matching technique proposed in [6] aims, in a framework of Bayesian neural networks, at matching the moment of the posterior distribution of the network for the old and the new task. In the context of object detection architecture, [9] proposed to limit catastrophic forgetting by modifying the loss function.The iCaRL model proposed in [7] addresses class-incremental learning in an essentially prototype-based architecture, with a focus on managing/updating the class-specific prototypes in an incremental fashion.

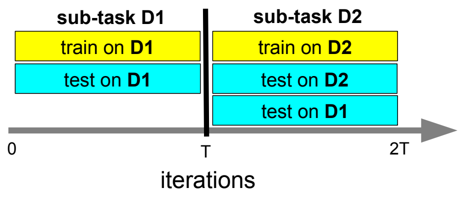

Fig. 2: Basic scheme of incremental training experiments conducted in this article. Initial training is conducted using a sub-task $D1$ for T iterations, followed by retraining on sub-task $D2$ for another T iterations. Both datasets differ in their statistical properties; in this article, we model this 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.

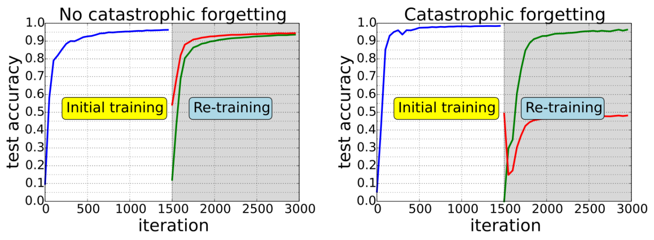

Fig. 3: Two prototypical experiments without (upper diagram) and with (lower diagram) catastrophic forgetting. Each experiment is subdivided into initial training on sub-task $D1$ (left-hand side, white background) and re-training on sub-task $D2$ (right-hand side, gray background). The dark blue curve indicates test performance on $D1$ during initial training, the red one test performance on $D1 \cup D2$ during re-training. The latter indicates unambiguously whether catastrophic forgetting happens. The green curve gives test performance on $D2$ during re-training and shows whether $D2$ has been well learned or not.

1.1 Models

We use TensorFlow/Python [1] 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/esann18.tar.gz`. The proposed MRL technique (see Fig. 1) could be in principle mounted 'on top' of any neural network. Here, we use a 'normal' fully-connected (FC) feed-forward MLP with two hidden layers, 2 softmax readout layers SM_i trained using cross-entropy, and the (optional) application of dropout (D) and ReLU operations after each hidden layer. The MRL technique proposes 2 readout layers here instead of a single one. Furthermore, a k-means clustering KM_i implementing the oracle is associated to every readout layer. Its structure can thus be summarized as Input-FC1-D-ReLU-FC2-D-ReLu-FC3-SM_i+KM_i. The choice of i for a particular model follows simple rules:

- During training, the readout layer is defined by the currently trained sub-task (one readout layer per sub-task). K-means clustering corresponding to this sub-task is performed on $D1$ in order to recognize samples coming from it later, realizing the oracle
- During testing, the oracle determines to which sub-task a sample belongs. This is achieved by running all clustering methods KM_i on the current test sample $\vec{x}$. If there is a clustering method KM_{i^*} one of whose cluster centers has the smallest distance to $\vec{x}$, the corresponding readout layer RL_{i^*} is used for generating a classification decision.

As the strategy uses one readout layer per sub-task, we need only two readout layers for the present time. MRL can be turned off by always selecting readout layer RL_i and clustering algorithm KM_i as RL_1 and KM_1. This case constitutes

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

the baseline of a normal fully-connected DNN, which our proposed MRL/oracle technique should outperform.

2 Datasets

The principal dataset this investigation is based on is MNIST[5]. 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[3, 10, 4]. As we will see, MNIST-derived class-incremental problems are more than a sufficient challenge for DNNs so it is really unnecessary to add more complex ones.

2.0.1 Permutation: DP10-10

This is the dataset used to evaluate incremental retraining in [3, 10, 4], so results can directly be compared to those in [3, 10, 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.

2.0.2 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).

2.0.3 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 Experiments

The basic experimental paradigm is very simple: we test the baseline model against the MRL/oracle technique (see Sec. 1.1) on all datasets described in Sec. 2. Evaluation measure is the best test error on $D1 \cup D2$ during the complete re-training period. It is imperative to consider the union of both sub-tasks for evaluating a model, since a strong drop on $D1$ test performance could just as easily be counteracted by an even faster rise in $D2$ test performance. For

task	D9-1a	D9-1b	D9-1c	D5-5a	D5-5b	D5-5c	DP10-10
acc. in %	97	98	99	96	97	97	99

Table 1: Performance of the oracle separating D1/D2 samples at test time.

task	D9-1a	D9-1b	D9-1c	D5-5a	D5-5b	D5-5c	DP10-10
acc./oracle	96.0	98	93.5	94.5	94.5	94.5	93.75
acc./baseline	55.5	65.1	79.7	41.1	45.3	48.8	88.2

Table 2: Performance of the MRL model with oracle as compared to the baseline model, a fully connected DNN. All accuracies are given in percent.

simplicity, we fix the topology of the used networks to two hidden layers of 400 neurons. We do not discuss topology variations here but just state that they did not change the gist of the results although of course performance is impacted. We always used an Euclidean distance measure and 500 cluster centers for all clustering methods. A performance baseline is always the best performance of any model during initial training. As in initial training, the used readout layer is always taken to be 1, all models reduce to a simple fully-connected DNN in this case. Another interesting quantity for evaluation is the classification accuracy of the clustering methods, or to put it in another way: how well can the oracle determine the provenance of a single test sample?

3.1 Accuracy of the oracle

The accuracy of the oracle is given in Tab. 1 and shows that, with the chosen parameters, a near-perfect oracle can be achieved at least on MNIST, largely independently of the partitioning of the classes.

3.2 Accuracy of the MRL model with oracle

As can be seen from Tab. 2 and Fig. 4, MRL with an oracle outperforms the baseline model of a fully-connected DNN by a large margin. Furthermore, we

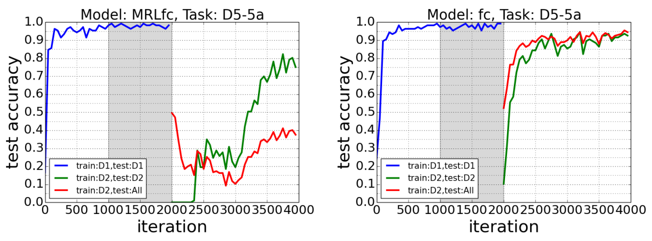

Fig. 4: Graphical representation of class-incremental learning progress. Left: baseline model, right: MRL with oracle. All curves indicate test accuracies.

can see that MRL with oracle maintains a stable performance on $D1 \cup D2$ independently of retraining time, removing the necessity to find the exact right moment to stop retraining. We observe as well that the baseline model does well on the task DP10-10, however this seems to be a task that does not incur catastrophic forgetting in any model (see [3, 10, 4]), so it should not be taken too seriously.

4 Discussion

We presented a simple technique to give fully-connected DNNs the ability to learn incrementally. A modest evaluation is conducted on MNIST to give a proof-of-concept which is all that is intended here. All evidence is purely empirical, and we can offer no proofs why and when our model should perform well, other than stating that the readout layer is the layer closest to the source o the gradient in a DNN, and that it is therefore most subject to change when statistics change. Protecting this layer in particular therefore seems a sensible thing to do. Next steps will obviously include tests on more challenging datasets, as well as a rigorous parameter search, as well as an extension to other types of DNNs such as CNNs.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

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

[3] I. J. Goodfellow, M. Mirza, X. Da, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgeting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[4] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835, 2017.

[5] 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.

[6] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems*, pages 4655–4665, 2017.

[7] S.-A. Rebuffi, A. Kolesnikov, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proc. CVPR*, 2017.

[8] A. Rosenfeld and J. K. Tsotsos. Incremental learning through deep adaptation. *arXiv preprint arXiv:1705.04228*, 2017.

[9] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. *arXiv preprint arXiv:1708.06977*, 2017.

[10] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber. Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318, 2013.