INTELLIGENT SYNAPSES FOR MULTI-TASK AND TRANSFER LEARNING

Friedemann Zenke[*], **Ben Poole**[*], **Surya Ganguli**
Neural Dynamics and Computation Lab, Stanford University
{fzenke,benpoole,sganguli}@stanford.edu

ABSTRACT

Deep learning has led to remarkable advances when applied to problems in which the data distribution does not change over the course of learning. In stark contrast, biological neural networks exhibit continual learning, solve a diversity of tasks simultaneously, and have no clear separation between training and evaluation phase. Furthermore, synapses in biological neurons are not simply real-valued scalars, but possess complex molecular machinery that enable non-trivial learning dynamics. In this study, we take a first step toward bringing this biological complexity into artificial neural networks. We introduce *intelligent synapses* which are capable of accumulating information over time, and exploiting this information to efficiently protect old memories from being overwritten as new problems are learned. We apply our framework to learning sequences of related classification problems, and show that it dramatically reduces catastrophic forgetting while maintaining computational efficiency.

1 INTRODUCTION

Deep learning has become an indispensable asset for applied machine learning that rivals human performance in a variety of tasks (LeCun et al., 2015). However, building systems that can simultaneously solve many tasks and continually learn over long timescales remains a challenging open problem. One of the major difficulties for continual learning in both biological and machine learning is acquiring knowledge needed to solve new tasks without forgetting what was learned on earlier tasks (Fusi et al., 2005; Lahiri & Ganguli, 2013; Benna & Fusi, 2016; Goodfellow et al., 2013). In this work, we develop an efficient online framework for regularizing neural networks that preserves the structure that is important for solving earlier tasks. This enables networks to learn sequences of tasks without forgetting, and improves generalization in transfer learning.

Prior approaches to alleviating forgetting in machine learning have primarily focused on architectural changes and functional regularization. Architectural approaches to catastrophic forgetting alter the architecture of the network to reduce interference between tasks without altering the objective function. For example, freezing subsets of weights (Razavian et al., 2014), fine-tuning from old weights (Donahue et al., 2014; Yosinski et al., 2014), altering nonlinearities (Srivastava et al., 2013; Goodfellow et al., 2013), or copying and augmenting networks (Rusu et al., 2016). Functional approaches alter the objective by adding a regularization term that penalizes changes in the input-output function of the neural network. In Li & Hoiem (2016); Jung et al. (2016), the log probabilities or hidden units are constrained to be close to their values for the old parameters. While these approaches explicitly preserve aspects of the input-output mapping for the old task, they can be costly to compute.

The approach we take in this work is *structural* regularization, which adds data-independent penalties to the parameters of a model. Structural approaches to catastrophic forgetting can be far more efficient than functional approaches as they do not depend on data to be evaluated.

[*] authors contributed equally

2 INTELLIGENT SYNAPSES

Core to our approach is the assumption that synapses that were important for solving previous tasks should not be altered, while synapses that were not important can be modified. Given the importance Ω_k for each synapse k, we can add a regularization term to the loss $L(\boldsymbol{\theta})$ that penalizes changes in synaptic strength relative to some baseline $\tilde{\boldsymbol{\theta}}_k$ proportional to their importance:

$$\tilde{L}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + c\sum_k \Omega_k \left(\boldsymbol{\theta}_k - \tilde{\boldsymbol{\theta}}_k\right)^2 . \tag{1}$$

While this framework supports many rules for choosing the importance of each synapse, we turned to a local importance measure that assigns credit to each synapse for improvements in the global objective. For small changes in parameter space, we can represent the improvement in the objective using a linear approximation:

$$L(\boldsymbol{\theta}(t) + \boldsymbol{\delta}(t)) - L(\boldsymbol{\theta}(t)) \approx \sum_k g_k(\boldsymbol{\theta}(t))\boldsymbol{\delta}_k(t) \, , \tag{2}$$

where $g_k(\boldsymbol{\theta}(t)) \equiv \frac{\partial L}{\partial \theta_k}\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}(t)}$ is the gradient of the loss with respect to synapse k at time t, and $\boldsymbol{\delta}_k(t)$ is the change in the synapticic weight $\boldsymbol{\theta}_k$ at time t (for gradient descent, $\boldsymbol{\delta}_k(t) = -\eta g_k(\boldsymbol{\theta}(t))$). Synapses that have a large gradient and experience large changes contribute the most to decreasing the loss, whereas synapses with a small gradient and/or small change contribute less. Integrating these contributions over the course of training allows individual synapses to incrementally build up an estimate of their importance, ω_k. In the limit of infinitesimal updates with $\boldsymbol{\theta}'_k(t) \equiv \boldsymbol{\delta}_k(t)$, the sum of the per-synapse importances, ω_k, can be written as the path integral over the gradient field:

$$\int_C \boldsymbol{g}(\boldsymbol{\theta}(t))d\boldsymbol{\theta} = \int_{t_0}^{t_1} \boldsymbol{g}(\boldsymbol{\theta}(t)) \cdot \boldsymbol{\theta}'(t)dt = \sum_k \int_{t_0}^{t_1} g_k(\theta_k(t))\delta_k(t)dt \equiv -\sum_k \omega_k. \tag{3}$$

The value of the integral in Eq. 3 is given by the difference in loss between the start and end point of the training trajectory, $\boldsymbol{\theta}(t_0)$ and $\boldsymbol{\theta}(t_1)$ respectively. Thus ω_k can be interpreted as an additive per-synapse contributions to decreasing the total training loss: $L(\boldsymbol{\theta}(t_1)) - L(\boldsymbol{\theta}(t_0)) = -\sum_k \omega_k$.

In practice, we solve a sequence of tasks indexed by μ, and compute the importance measure Ω_k^μ (used in Eq. 1) by summing over a scaled version of the importances for all previous tasks, ν:

$$\Omega_k^\mu = \sum_{\nu<\mu} \frac{\omega_k^\nu}{(\Delta_k^\nu)^2 + \xi} \, , \quad \text{with} \quad \Delta_k^\nu \equiv \theta_k(t^\nu) - \theta_k(t^{\nu-1}) \tag{4}$$

The scaling ensures consistency of units, and the dampening parameter ξ bounds the expression in cases for which parameter changes are small ($\Delta_k^\nu \ll \xi$).

A typical training cycle in the benchmark then proceeds as follows. Initially all Ω_k^μ are set to zero and the network is trained on Task 1. During training ω_k^μ is estimated as a running sum $\omega_k = \sum_i g_k^i \delta_k^i$ over mini batches. For simplicity we updated the baseline variables Ω_k^μ and θ_k at the end of each task. More specifically, at the end of training on Task 1 we set $\tilde{\theta}_k \rightarrow \theta_k$ and set $\Omega_k^\mu = \sum_{\nu<\mu} \omega_k^\nu$. After updating Ω_k^μ, the synaptic variables ω_k are reset to zero. The strength paramter c was tuned manually and typically chosen in the range of 1.

The approach presented here is similar in spirit to elastic weight consolidation (EWC) (Kirkpatrick et al., 2016) in that the form of the regularization is identical, and important parameters are pulled back stronger towards their reference weight. However, in contrast to EWC, our method can be computed online over the course of training with minimal overhead as it only relies on a running sum of the products of gradients and updates. EWC relies on the Fisher information to identify the importance of each weight which has to be computed during a separate phase after learning parameters for each task. Computing the Fisher can also be costly in high-dimensional output spaces, as it requires an expectation over samples from the output of the model.

3 RESULTS

We evaluated our approach for multi-task learning on split MNIST and permuted MNIST, and transfer learning from MNIST to MNIST-bg and between subsets of classes on CIFAR-10.

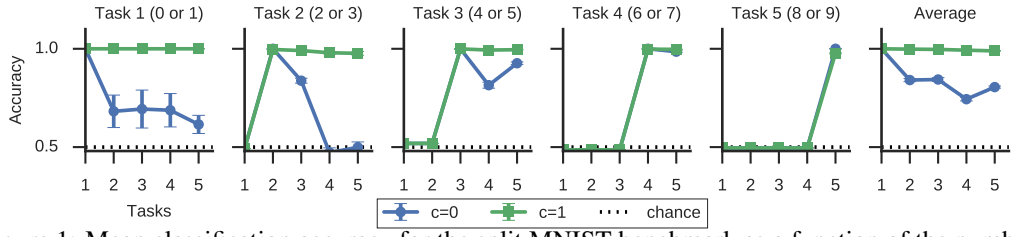

Figure 1: Mean classification accuracy for the split MNIST benchmark as a function of the number of tasks. Each task is a binary classification task between two MNIST digits. We use the same network to jointly solve all tasks but have a separate linear class readout form the final hidden layer for each task. The first five panels show classification accuracy on each task as a function of the number of consecutive tasks. The rightmost panel shows the average accuracy for all seen tasks, which is computed as the average over all previous tasks. Blue lines ($c = 0$) correspond to fine-tuning with no regularization, while green lines ($c = 1$) correspond to intelligent synapses with a strength of 1. Error bars correspond to SEM (n=10).

For multi-task learning, we find that intelligent synapses greatly reduce forgetting and retain strong performance while learning up to 10 tasks (Fig. 1,2). These results were consistent across training and validation error, and were comparable to the results reported in Kirkpatrick et al. (2016).

For transfer learning, we compared our approach to the standard feature extraction approach (freezing all weights except the final readout layer), fine tuning, and training from scratch. We find that when transferring to a task with a small amount of data, intelligent synapses improve validation accuracy over all these approaches (Fig. 3).

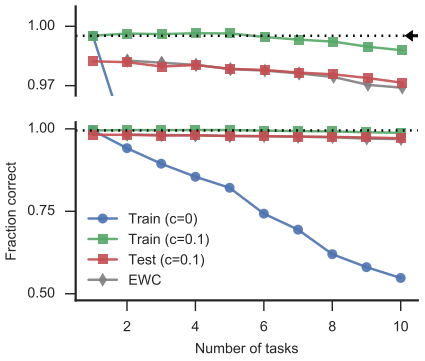

Figure 2: Intelligent synapses retain performance on permuted MNIST task. Each task is created by randomly permutating pixels. Classification accuracy on all seen tasks as a function of number of tasks. With no regularization, the network quickly forgets old tasks (blue), while intelligent synapses (green) and elastic weight consolidation (grey, extracted from Kirkpatrick et al. (2016)) retain performance over many tasks. The top panel is a zoom-in on the top section of the graph, with the dotted line showing initial training accuracy on a single task, and the black arrow showing training accuracy when trained on all tasks simultaneously.

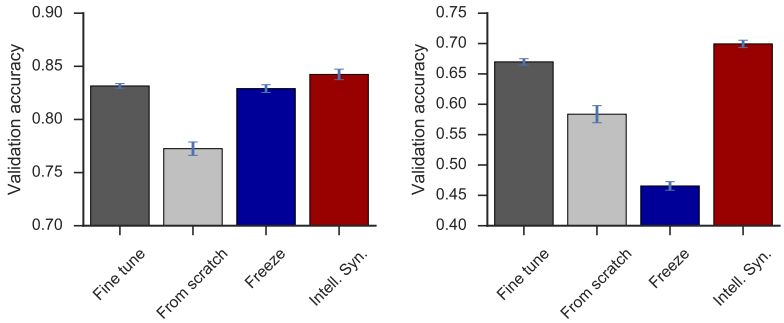

Figure 3: Intelligent synapses improve transfer learning. **Left:** Validation accuracy when transferring from all data of CIFAR 10 classes 0-7 to binary classification between CIFAR-10 class 8 vs 9 using 50 examples per class. The network was a convolutional neural network. Chance performance is 0.5. **Right:** Transfer from normal MNIST to MNIST-bg (background images) using only 10 labeled examples from each of the 10 classes of MNIST-bg. Chance performance is 0.1. Error bars correspond to SEM (n=5).

REFERENCES

Marcus K. Benna and Stefano Fusi. Computational principles of synaptic memory consolidation. *Nat Neurosci*, advance online publication, October 2016. ISSN 1097-6256. doi: 10.1038/nn.4401. URL http://www.nature.com/neuro/journal/vaop/ncurrent/full/nn.4401.html.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference in Machine Learning (ICML)*, 2014.

Stefano Fusi, Patrick J. Drew, and Larry F. Abbott. Cascade models of synaptically stored memories. *Neuron*, 45(4):599–611, February 2005. ISSN 0896-6273. doi: 10.1016/j.neuron.2005.02.001. URL http://www.ncbi.nlm.nih.gov/pubmed/15721245.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv:1312.6211 [cs, stat]*, December 2013. URL http://arxiv.org/abs/1312.6211. arXiv: 1312.6211.

Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting Learning in Deep Neural Networks. *arXiv:1607.00122 [cs]*, July 2016. URL http://arxiv.org/abs/1607.00122. arXiv: 1607.00122.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *arXiv:1612.00796 [cs, stat]*, December 2016. URL http://arxiv.org/abs/1612.00796. arXiv: 1612.00796.

Subhaneil Lahiri and Surya Ganguli. A memory frontier for complex synapses. In *Advances in Neural Information Processing Systems*, volume 26, pp. 1034–1042, Tahoe, USA, 2013. Curran Associates, Inc. URL http://papers.nips.cc/paper/4872-a-memory-frontier-for-complex-synapses.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 0028-0836. doi: 10.1038/nature14539. URL http://www.nature.com/nature/journal/v521/n7553/full/nature14539.html.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pp. 614–629. Springer, 2016.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 806–813, 2014.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv:1606.04671 [cs]*, June 2016. URL http://arxiv.org/abs/1606.04671. arXiv: 1606.04671.

Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Juergen Schmidhuber. Compete to Compute. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2310–2318. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5059-compete-to-compute.pdf.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.