Backpropagated plasticity: learning to learn with gradient descent in large plastic neural networks

Thomas Miconi
Uber AI Labs
tmiconi@uber.com

Jeff Clune
Uber AI Labs
jeffclune@uber.com

Kenneth O. Stanley
Uber AI Labs
kstanley@uber.com

Abstract

How can we build agents that keep learning from experience, quickly and efficiently, after their initial training? Here we take inspiration from the main mechanism of learning in biological brains: synaptic plasticity carefully designed by evolution to produce efficient lifelong learning. We show that plasticity, just like connection weights, can be optimized by gradient descent in large (millions of parameters) recurrent networks with Hebbian plastic connections. Applied to the task of arbitrary natural image memorization, recurrent plastic networks with more than two million parameters can be trained to memorize and reconstruct sets of novel, high-dimensional (1,000+ pixels) natural images not seen during training. Surprisingly, the trained networks exhibit highly structured plasticity, in contrast with traditional, homogenous auto-associative networks. Crucially, traditional non-plastic recurrent networks fail to solve this task, and require orders of magnitude more training to partially solve a considerably simpler version of it. In conclusion, backpropagated plasticity may provide a powerful novel approach to the learning-to-learn problem.

1 Introduction: the problem of "learning to learn"

Designing agents that can quickly learn from ongoing experience is the basic problem of meta-learning, or "learning-to-learn" [14]. Several methods exist to address this problem, including endowing networks with external content-addressable memory banks, as in Memory Networks and Neural Turing Machines [3, 13, 10]; augmenting networks with "fast weights" that essentially attend to recently encountered patterns [1] (which can in principle be exploited by adequately trained fixed weights to store any desired memory [11]); or even simply training standard recurrent networks (which, as universal Turing machines, can *in principle* learn any computable function of their inputs) to adequately incorporate past experience in their future response properties [6, 15, 2].

In biological brains, however, long-term learning is thought to occur primarily through *synaptic plasticity* – the strengthening and weakening of connections between neurons as a result of neural activity, as carefully tuned by evolution over millions of years to enable efficient learning during the lifetime of each individual. While multiple forms of synaptic plasticity exist, many of them follow the general principle known as Hebb's rule: if a neuron repeatedly takes part in making another neuron fire, the connection between them is strengthened (often roughly summarized as "neurons that fire together, wire together") [4]. This principle underlies several forms of observed plasticity in animal brains, allowing them to learn from experience and adapt to their environment.

Designing neural networks with plastic connections has long been explored with evolutionary algorithms (see [12] for a recent review). However, given the spectacular results of gradient descent in designing traditional non-plastic neural networks for complex tasks, it would be of great interest to expand backpropagation training to networks with plastic connections. We previously demonstrated the theoretical feasibility and analytical tractability of this approach [9]. Here we show that the

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

approach is successful in training large (millions of parameters) networks for non-trivial tasks, which cannot be solved by conventional non-plastic networks.

2 Backpropagated plasticity: training neural networks with plastic connections using gradient descent

To train plastic networks with backpropagation, a plasticity rule must be specified. Many formulations are possible. Here we chose the simplest formulation that would keep separate plastic and non-plastic (baseline) components for each connection, while guaranteeing stability (that is, avoiding weight explosion caused by the positive feedback inherent to Hebbian learning).

The connection between neurons i and j is structurally determined by two parameters: a baseline fixed weight $w_{i,j}$, and a plasticity coefficient $\alpha_{i,j}$. The plastic component of each synapse is stored in a *Hebbian trace* $\mathrm{Hebb}_{i,j}$, which varies during a lifetime according to ongoing inputs and outputs. In the current implementation, the Hebbian trace is simply a running average of the product of pre- and post-synaptic activity. The total, effective weight of a connection is the sum of the baseline (fixed) weight, plus the Hebbian trace multiplied by the plasticity coefficient. The precise network equations for the output $x_j(t)$ of neuron j are:

$$x_j(t) = \tanh\big\{\sum_{i \in inputs} [w_{i,j}x_i(t-1) + \alpha_{i,j}\mathrm{Hebb}_{i,j}(t)x_i(t-1)]\big\}, \quad (1)$$

$$\mathrm{Hebb}_{i,j}(t+1) = \eta x_i(t-1)x_j(t) + (1-\eta)\mathrm{Hebb}_{i,j}(t). \quad (2)$$

Thus, depending on the values of $w_{i,j}$ and $\alpha_{i,j}$, a connection can be fully fixed (if $\alpha = 0$), or fully plastic with no fixed component (if $w = 0$), or have both a fixed and a plastic component. The Hebbian trace $\mathrm{Hebb}_{i,j}$ is initialized to zero at the beginning of each lifetime/episode (we use "lifetime" and "episode" interchangeably): it is purely a lifetime quantity. The parameters $w_{i,j}$ and $\alpha_{i,j}$, on the other hand, are the structural parameters of the network that are conserved across lifetimes and optimized by gradient descent between lifetimes (descending the gradient of the error computed over the lifetime), to produce optimal expected performance over a lifetime/episode. While η, the "learning rate" of plasticity, could also be optimized, for simplicity here it is kept fixed at 0.01. This formulation of plasticity is easily implemented in automatic differentiation packages. All experiments reported here use the PyTorch package to compute gradients.

3 Results

To demonstrate the backpropagated plasticity approach, we apply it to the task of memorizing sets of arbitrary patterns (including novel patterns never seen during training), and reconstructing these patterns when exposed to partial, degraded versions of them. Networks that can perform this task are known as content-addressable memories, or auto-associative networks. This task is a useful test because it is known that hand-designed recurrent networks with (usually homogenous) Hebbian plastic connections can successfully solve it for binary patterns [7]. Thus, if backpropagated plasticity is to be of any help, it should be able to automatically solve this task – that is, to automatically design networks that can perform this task just like existing hand-designed networks can.

3.1 Binary patterns

Figure 1 describes an episode in this task. The network is shown a set of 5 binary patterns in succession. Each binary pattern is composed of 1,000 elements, each of which is either 1 or -1. Each pattern is shown for 10 time steps, with 3 time steps of zero input between presentations, and the whole sequence of patterns is presented 3 times in random order (few-shot learning). Then, one of the presented patterns is chosen at random and degraded, by setting half of its bits to zero (i.e. no input). This degraded pattern is then fed as an input to the network. The task of the network is to reproduce the correct full pattern in its outputs, drawing on its memory to complete the missing bits of the degraded pattern.

The architecture (Figure 1) is a fully recurrent neural network with one neuron per pattern element, plus one fixed-output ("bias") neuron, for a total of 1,001 neurons. Input patterns are fed by clamping the value of each neuron to the value of the corresponding element in the pattern, if this value is

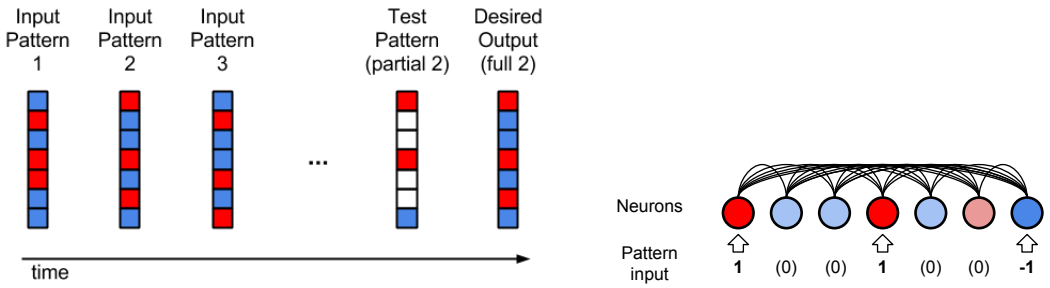

Figure 1: Left: Conceptual description of the task. Right: depiction of the architecture.

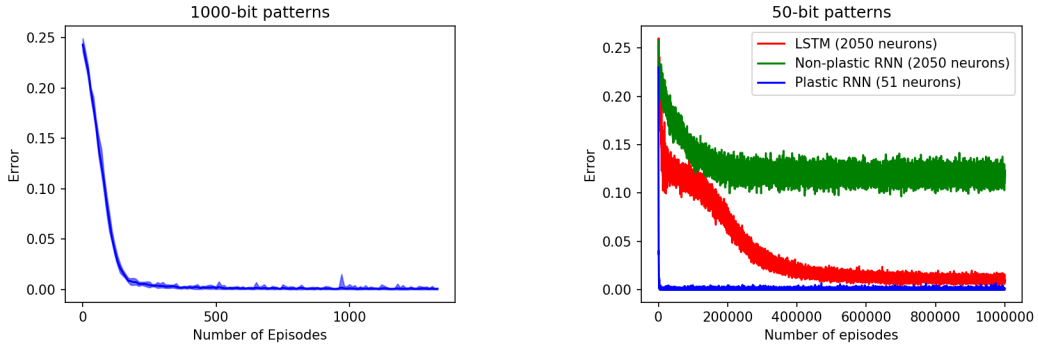

Figure 2: Left: Learning curve for 1,000-bit pattern memorization (10 runs shown: shaded area indicates minimum and maximum loss, thick curve indicates mean loss). Right: Learning curve for typical runs for 50-bit patterns, using a non-plastic RNN with 2050 neurons (green curve), an LSTM with 2050 neurons (red curve), and a backpropagated plastic-weight network with the same parameters but only 51 neurons (blue curve).

not zero (i.e. 1 or -1); for zero-valued inputs in degraded patterns, the corresponding neurons do not receive pattern input, and get their inputs solely from lateral connections, from which they must reconstruct the correct, expected output values. Outputs are read directly from the activation of the neurons. The network's performance is evaluated only on the final time step, by computing the loss as the summed squared error between the final network output and the correct expected pattern (that is, the non-degraded version of the degraded input pattern). The gradient of this error over the $w_{i,j}$ and $\alpha_{i,j}$ coefficients is then computed by backpropagation, and these coefficients are optimized through an Adam solver [8] with learning rate 0.001. Note that the network has two trainable parameters (w and α) for each connection, summing up to $1{,}001 \times 1{,}001 \times 2 = 2{,}004{,}002$ trainable parameters.

Figure 2 (left) shows the result of a typical training run. Error (defined as the proportion of bits that have the wrong sign) converges to a low, residual value (<1%) within about 200 episodes.

3.2 The importance of being plastic: a comparison with non-plastic recurrent networks

In principle, this task (like any computable tasks) could be solved by a non-plastic recurrent network, although the non-plastic networks will require additional neurons to store inputs. However, despite much exploration, we were unable to succeed in solving the original task with a non-plastic RNN or LSTM [5]. We could only succeed by reducing the pattern size to 50 bits (down from 1,000), showing only 2 patterns per episode (rather than 5), and presenting them for only 3 time steps. The best results required adding 2000 extra neurons (for a total of 2050 neurons). Training error over episodes is shown in Figure 2 (right). For the non-plastic RNN, the error slowly decreases over millions of episodes (green curve). The LSTM solves the task, imperfectly, after about 500,000 episodes (red curve). For comparison, the blue curve shows the exact same problem, architecture, and parameters, but restoring plastic connections (η=.05 to account for faster presentations). The network solved the task very quickly, reaching mean error below .01 within 2,000 episodes.

Thus, for this specific task, plastic recurrent networks seem considerably more powerful than LSTMs. Although this task is known to be well-suited for plastic recurrent networks [7], this result raises the

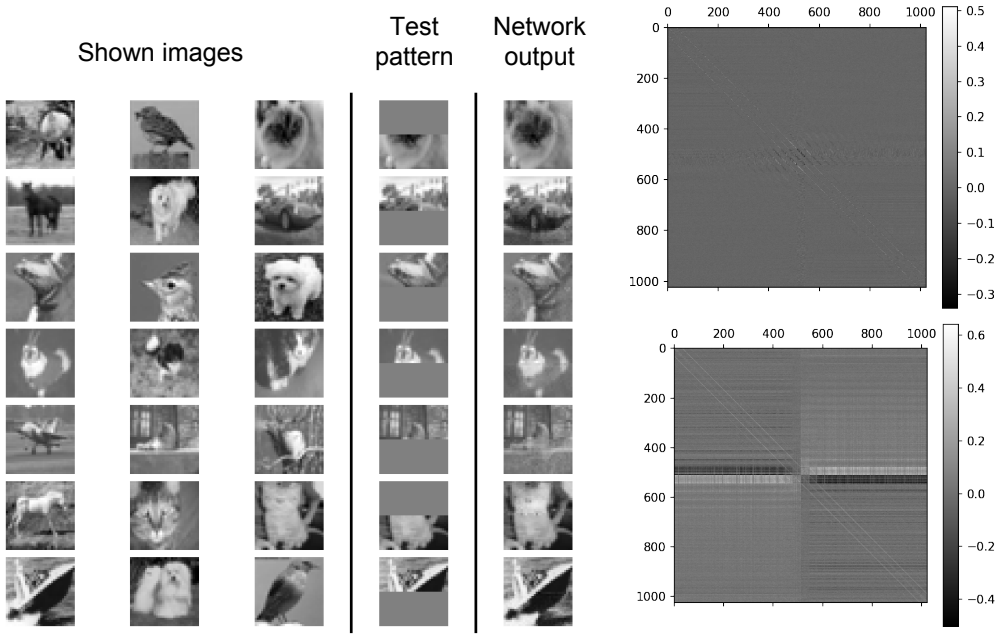

Figure 3: Left: Typical image reconstruction results from a withheld test set (not seen during training). Each row is a full episode. Right: matrices of baseline weights $w_{i,j}$ (top) and plasticity coefficients $\alpha_{i,j}$ (bottom) after training. Each column describes the input to a single cell, and vertically adjacent entries describe inputs from horizontally adjacent pixels in the image. Notice the significant structure present in both matrices (best viewed electronically by zooming).

question of which other domains might benefit from the backpropagated plasticity approach over current LSTM models.

3.3 Natural images

As a more challenging test, we applied our method to the problem of memorizing natural images with graded pixel values, which contain much more information per element. Images are from the CIFAR-10 database, which contains 60,000 images of size 32 by 32 pixels (i.e. 1,024 pixels in total), converted to grayscale pixels between 0 and 1.0. The architecture is largely similar to the one described above, with 1,025 neurons in total, leading to $2 \times 1{,}025 \times 1025 = 2{,}101{,}250$ parameters. Each episode included 3 pictures, shown 3 times (in random order each time) for 20 timesteps each time, with 3 time steps of zero input between image presentations. To prevent a trivial solution consisting in simply reconstructing each missing pixel as the average of its neighbors (which the high autocorrelation of natural images might make viable), images are degraded by zeroing out one full contiguous half of the image (either top or bottom half).

Figure 3 (left) shows the behavior of the trained network on a withheld test set of images not seen during training. The last column shows the final output of the network, i.e. the reconstructed image. The model has successfully learned to perform the non-trivial task of memorizing and reconstructing previously unseen natural images.

Figure 3 (right) also shows the final matrices of weights and plasticity coefficients produced by training. The plasticity matrix (bottom) shows considerable structure, in contrast to the homogenous plasticity of traditional Hopfield networks [7]. Some of the structure (diagonal lines) is related to the high correlation of neighboring pixels, while other aspects (alternating bands near the midsection) result from the choice to use half-field zeroing in test images.

In conclusion, these preliminary results demonstrate the feasibility of training structural plasticity in million-parameter recurrent networks, massively improving performance over non-plastic networks on some tasks. This largely unexplored paradigm, with strong inspiration from biological learning mechanisms, offers an intriguing new direction for time-dependent learning in general, and meta-learning in particular.

References

[1] Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4331–4339. Curran Associates, Inc., 2016.

[2] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. 2016, arXiv:1611.02779.

[3] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. October 2014, arXiv:1410.5401.

[4] Donald O. Hebb. The organization of behavior: a neuropsychological theory. 1949.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Sepp Hochreiter, A Younger, and Peter Conwell. Learning to learn using gradient descent. *Artificial Neural Networks—ICANN 2001*, pages 87–94, 2001.

[7] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*. 2015.

[9] T. Miconi. Backpropagation of hebbian plasticity for continual learning. In *NIPS Workshop on Continual Learning*, 2016.

[10] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with Memory-Augmented neural networks. 19 May 2016, arXiv:1605.06065.

[11] J. Schmidhuber. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In Stan Gielen and Bert Kappen, editors, *ICANN '93: Proceedings of the International Conference on Artificial Neural Networks Amsterdam, The Netherlands 13–16 September 1993*, pages 460–463. Springer London, London, 1993.

[12] Andrea Soltoggio, Kenneth O. Stanley, and Sebastian Risi. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. 2017, arXiv:1703.10371.

[13] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-To-End memory networks. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[14] Sebastian Thrun and Lorien Pratt. Learning to learn. In Sebastian Thrun and Lorien Pratt, editors, *Learning to Learn*, chapter Learning to Learn: Introduction and Overview, pages 3–17. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[15] Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. 2016, arXiv:1611.05763.