Evolving Neural Turing Machines

Rasmus Boll Greve, Emil Juul Jacobsen, and Sebastian Risi
IT University of Copenhagen
Copenhagen, Denmark
{ragr,ejuu,sebr}@itu.dk

Abstract

Instead of training a *Neural Turing Machine* (NTM) with gradient descent [1], in this work NTMs are trained through an evolutionary algorithm. Preliminary results suggest that this setup can greatly simplify the neural model, generalizes better, and does not require accessing the entire memory content at each time-step. We show preliminary results on a simple copy and T-Maze learning task.

1 Introduction

An important component of our human cognition is the ability to store and use information during inference. However, while much progress has been made in machine learning [2, 3], how to best augment artificial agents with a similar long-term memory component remains elusive. In recent work in this direction, Graves et al. [1] extended the capabilities of an artificial neural network (ANN) by combining it with an external memory bank. Because the architecture of their *Neural Turing Machine* (NTM) is differentiable end-to-end it can be trained through gradient descent and was able to learn simple algorithms such as copying, sorting and recall from example data. However, the differentiable memory requires the NTM to access the entire memory content at each step, which can be prohibitively slow for larger memory banks and will likely not scale-up to large and more difficult problems. Others have tried to overcome this obstacle by combining gradient descent methods with reinforcement learning methods, so far with only limited success [4].

This study presents preliminary results training a NTM through a neuroevolutionary (NE) approach, in which both the topology and the weights of a network are determined through artificial evolution. We replicate the copy tasks from Graves et al. [1] but also show that the NE approach can be applied directly to allow an agent to solve a T-Maze learning tasks. Because we are not restricted to models that are differentiable end-to-end, the evolutionary NTM is significantly simpler than the original NTM, generalises perfectly to longer sequences on the copy task, and is not limited by a fixed-size memory bank. Importantly, alternative NTM implementations might help to elucidate which components are a necessity for an AI with deeper reasoning skills to emerge, and which are just a byproduct of a particular training method.

2 Evolutionary Neural Turing Machine (ENTM)

In this paper, the NTMs are evolved with NEAT [5]. NEAT is an algorithm that evolves increasing large ANNs, and has shown promise in a variety of complex control tasks. It starts with a population of simple networks and then adds complexity over generations by adding new nodes and connections through mutations. By evolving ANNs in this way, the topology of the network does not need to be known a priori; NEAT searches through increasingly complex networks to find a suitable level of complexity. Because it starts simply and gradually adds complexity, it tends to find a solution network close to the minimal necessary size.

Training a NTM through evolution allows the system to be simpler and less restrictive (Figure 1) than in the original setup [1]. The components of the *Evolutionary Neural Turing Machine* (ENTM) include the same as in the original NTM architecture: an artificial neural network (ANN) controller and an external memory bank. However, our model has a theoretically infinite memory bank that can store vectors of size M at each memory location. Additionally, it only has a single combined head, which is responsible for both writing and reading. The ANN interacts with the external environment through its inputs and outputs but it can also perform selective read/write operations, shift the current read from/write to position in memory, and perform content-based addressing. The number of neural network inputs and outputs corresponds to the vector size M, plus additional outputs for control (e.g. shift, etc.). In more detail, the ENTM performs the following four steps:

1. Write: A write interpolation parameter determines how much of the existing memory vector at the current head location should be interpolated with the given write vector: $\boldsymbol{M}_{t+1}(h) \longleftarrow \boldsymbol{M}_t(h) \cdot (1 - w_t) + \boldsymbol{a}_t \cdot w_t$, where $\boldsymbol{M}_t(h)$ is the memory vector at the head's location at time t, w_t is the write parameter and $\boldsymbol{a}_t$ is the write vector.

2. Content Jump: If the ANN jump output is higher than a given threshold (0.5 in this paper) a content-based jump of the head is performed to the position on the tape which is most similar to the write vector. In the current implementation this is based on an Euclidean distance function.

3. Shift: The controller can perform a shift, which moves the memory head relative to its current location. The ANN has three shift outputs, where the highest activated output determines if the head should be moved one position to the left, one position to the right or remain at the current position.

4. Read: After the possible content jump and shift, the neural network is automatically given the content of the memory vector at the head's final location as input.

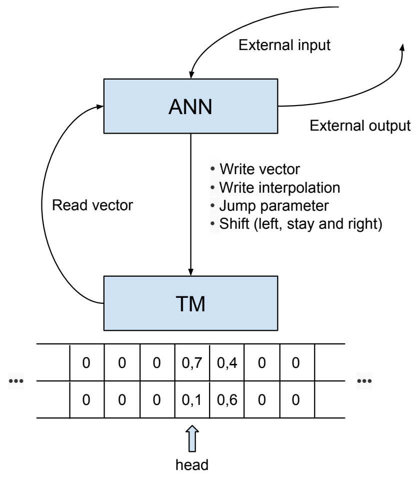


Figure 1: Evolvable Neural Turing Machine.

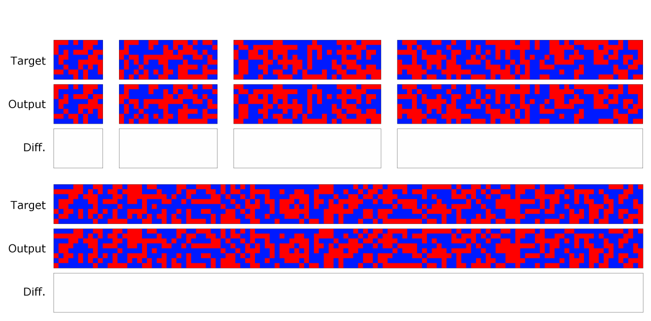
3 Experiments

The ENTM is tested on two different tasks: a copy task [1] and reward-based T-Maze scenario. We compare the ENTM model (Section 2) to a more complex architecture (*Diff-ENTM*), which more closely resembles the original NTM but is also trained with NEAT. The Diff-ENTM has a fixed memory size while the ENTM will expand the memory when addressing a previously unseen location. While the Diff-ENTM calculates a weighting over all memory locations determining how much each location will be effected by the write, the ENTM writes to a single location and uses an interpolation parameter to blend between the current memory content and the one produced by the previous time-step. Both approaches allow for shifting and content jumps; however, while they are part of the weighting calculation for Diff-ENTM, they constitute simple operations to the discrete read/write head in the ENTM. The Diff-ENTM has read and write performed by separate heads defined by different control parameters. A head in ENTM is responsible for both writing and reading, which reduces the number of parameters significantly.

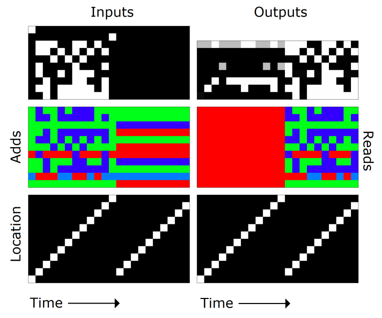
Experimental Parameters. The size of each population is 300. The maximum number of generations is 10,000. Sexual offspring (50%) does not undergo mutation and asexual offspring (50%) has a 0.6 probability of link weight mutation. The copy task has a 0.05 chance of link addition, 0.02 chance of removing a connection and 0.005 chance of node addition. For the T-Maze, the chance of link and node addition is 0.02, and the chance of removing a connection is 0.05. These parameters were found through preliminary experimentation. All results are averaged over ten independent evolutionary runs.

3.1 Copy Task

In the copy task [1] the neural network has to store and recall a long sequence of random binary vectors. The network is given a sequence of random bit vectors, a single bit to indicate the start of the exercise, and a single bit to signal the end of the stimulation phase and the beginning of the recall phase. The fitness of a network is calculated by how well it can recall stored sequences. If the produced bit vector has a match m of at least 25% to the target output, it receives a score of



(a) Successful Copy



(b) Memory Additions

Figure 2: Inputs and outputs for the ENTM solving the copy task with sequences of size 10, 20, 30, 50 and 120 are shown in (a). Memory additions and focus during a single test sequence are shown in (b). The top plots show the input to and outputs from the network respectively. The middle two plots show the vectors written to and read from the ENTM. The bottom plots shows the location in memory that the TM is focused at when writing (left) and reading (right). Note that the memory is "infinite" in both directions, and only the memory locations used by the network are shown.

$s = 1.33(m - 25)$. The final fitness is calculated by summing over the scores for each vector. To encourage solutions that generalise well, each ENTM is evaluated on 50 separate iterations of sequences with random lengths between one and ten. We evolved ENTMs for solving copy tasks with bit vector sizes of one, two, four and eight. The memory vector size was set to the bit vector size plus three extra locations for potential redundancy and control mechanisms that the ENTMs might require to solve the task.

The results show that the smaller the bit vector size the faster evolution finds a solution. The ENTM finds a solution in nine out of ten runs for the one bit version, in eight runs for the two bits, fives runs for four bits, and in one run for eight bits. If successfully, the ENTMs solves the one bit version in 77 generations on average, the two bit version in 400 generations, and four in 704 generations. The more complex Diff-ENTM performs significantly worse ($p < 0.05$ according to the Student's t-test) and only finds a solution in a single run for a vector size of one.

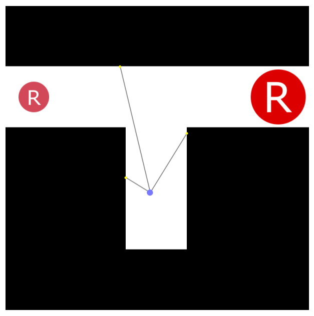
The ENTM solutions were tested on their ability to generalize to sequences longer than those encountered during evolution. Two champions from the four and eight bit experiments were evaluated on sequences of 10, 20, 50, 100 and 1,000 non-empty bit vectors. In contrast to the results by Graves et al. [1], the evolved solutions generalize perfectly even for very large sequences (Figure 2a). The ENTM continually performs a left shift while writing the input to memory. When reaching the delimiter input it performs a content-jump to the original position and continues shifting left while reading the memory back to the output neurons (Figure 2b).

Because NEAT starts simply and gradually adds nodes and connections, it is able to discover a sparsely connected champion network with only a single hidden neuron. This evolved network has a considerably smaller size than the original NTM, which is fully connected with 100 hidden neurons and a total of 17,162 parameters [1].

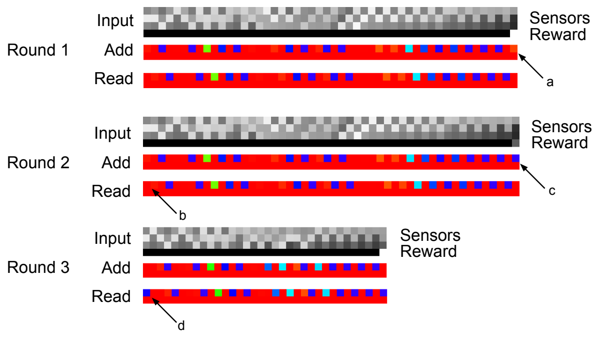
3.2 A Dynamic, Reward-based Scenario: Continuous T-Maze

The second domain investigated in this paper is the T-Maze task [6]. It is often studied in the context of operant conditioning of animals and therefore makes a good test domain for architectures involving memory. The T-Maze consists of two arms that either contain a high or low reward (Figure 3a). The simulated robot begins at the bottom of the maze and its goal is to navigate to the reward position. This procedure is repeated many times during the robots lifetime. When the position of the high reward sometimes changes, the robot should alter its strategy accordingly to explore the other arm of the maze in the next trial and remember the new position in the future.

The agent has three sensors that measure the distance to walls and an additional reward input that is only activated once the agent reaches either a low (0.1) or high (1.0) reward. The agent moves



(a) Continuous T-Maze



(b) Memory Usage

Figure 3: (a) The challenge for the robot in the T-Maze domain is to remember the location of the high reward from one round to the next. (b) The graph shows the values written to and read from memory by a ENTM solution during three consecutive rounds. The high reward goal is located to the right for round 1 and to the left for rounds 2 and 3. The ENTM never performs a shift or content-jump and only uses one memory location. In the first round the agent turns right and collects the high reward goal. Notice how the ENTM writes a low (red) value at point *a*, which is read back again at point *b*. In round 2 it performs another right turn, this time encountering a low reward which results in a higher value (blue) being written to memory (point *c*). This value is read back at point *d* at the start of round 3, which enables the ENTM to change its behavior and now turn left at the junction.

forward with a constant speed and turns by $45° R$, where R is the turn output scaled into the range $[-1, 1]$. The fitness is calculated as the total reward collected over all rounds, where the agent receives a reward of ten for collecting the high reward and a reward of one for the low reward. No reward is given if the agent collides with a wall or moves more than 100 steps without reaching a goal. The memory vector size is set to two at each location, which is large enough to potentially allow for key/value like usage or storage of a value for each goal in a single memory location.

Both ENTM and Diff-ENTM found a solution (a network that requires minimal exploration when the reward switches) in all ten runs. ENTM took on average 35 generations ($\sigma = 14$), and Diff-ENTM 37 generations ($\sigma = 19$). A generalisation test (testing on 10,000 randomly chosen scenarios with 45 rounds maximum and up to eight switches) showed that the ENTM solutions generalize perfectly while the Diff-ENTM solutions make a few errors when more switches are introduced. The memory usage of an evolved ENTM solution is shown in Figure 3b (see caption for details).

4 Conclusion

This paper has demonstrated the ability of an evolved Neural Turing Machine to successfully remember arbitrarily long sequences of up to eight bits and control a robot in a T-Maze learning task. In the future it will be interesting to extend this approach to more complex domains that require deeper reasoning and larger neural structures.

References

[1] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[2] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Net.*, vol. 61, pp. 85–117, 2015.

[3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *arXiv preprint arXiv:1504.00702*, 2015.

[4] W. Zaremba and I. Sutskever, "Reinforcement learning neural turing machines," *arXiv preprint arXiv:1505.00521*, 2015.

[5] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[6] A. Soltoggio, J. A. Bullinaria, C. Mattiussi, P. Dürr, and D. Floreano, "Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios," in *Proc. of Alife XI*, pp. 569–576, MIT Press, 2008.