Evolving Deep LSTM-based Memory networks using an Information Maximization Objective

Aditya Rawal
Computer Science, University of Texas at Austin
aditya@cs.utexas.edu

Risto Miikkulainen
Computer Science, University of Texas at Austin
and Sentient Technologies, Inc.
risto@cs.utexas.edu

ABSTRACT

Reinforcement Learning agents with memory are constructed in this paper by extending neuroevolutionary algorithm NEAT to incorporate LSTM cells, i.e. special memory units with gating logic. Initial evaluation on POMDP tasks indicated that memory solutions obtained by evolving LSTMs outperform traditional RNNs. Scaling neuroevolution of LSTM to deep memory problems is challenging because: (1) the fitness landscape is deceptive, and (2) a large number of associated parameters need to be optimized. To overcome these challenges, a new secondary optimization objective is introduced that maximizes the information (Info-max) stored in the LSTM network. The network training is split into two phases. In the first phase (unsupervised phase), independent memory modules are evolved by optimizing for the info-max objective. In the second phase, the networks are trained by optimizing the task fitness. Results on two different memory tasks indicate that neuroevolution can discover powerful LSTM-based memory solution that outperform traditional RNNs.

CCS Concepts

•**Computer systems organization** → **Neural Networks;**
•**Computing methodologies** → *Artificial Intelligence;*

Keywords

Algorithms

1. INTRODUCTION

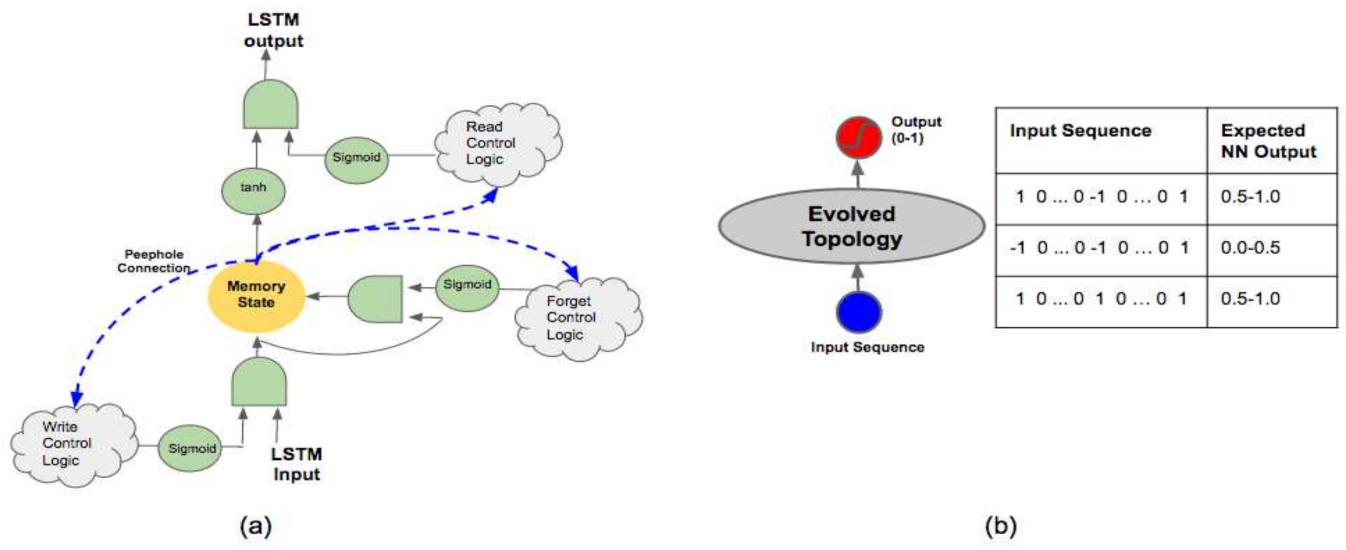
Natural organisms can memorize and process sequential information over long time lags. Chimpanzees and orangutans can recall events that occurred more than a year ago [16]. Long term social memory can provide significant survival benefits. For example, bottlenose dolphins can recognize each other's whistle sounds even after decades [7]. Such a capability allows the dolphin to identify adversaries as well as potential teammates for hunting. First step towards adaptive behavior is to memorize past events and utilize them for future decision making [23]. For example, a group of hyenas, during lion-hyena interactions, modulate their behavior over a period of time through memory-based emotions - transitioning from being fearful initially to becoming risk-taking later [27].

Memory is a key cognitive component and incorporating this capability in artificial agents can make them more realistic [22]. New methods are presented in this paper that can evolve deep sequence processing networks to solve reinforcement-learning (RL) memory tasks with long time-dependencies. Tasks requiring memory can be formally be described as POMDP problems. Traditionally, recurrent neural networks (RNNs) have been the preferred choice for this purpose. However, RNNs leak information and are unable to discover long-term dependencies [11]. Long Short Term Memory (LSTM) [12] successfully overcomes these limitations of RNNs. It consists of memory cells with linear activations. The inflow and outflow of information to and from these cells is controlled by associated input/output gated units. While LSTM networks have been used to achieve strong results in the supervised sequence learning problems such as in speech recognition [10] and machine translation [2], their success in POMDP tasks has been limited [5, 4]. A possible reason is that it is difficult to train LSTM units (including its associated control logic) with weak reward/fitness signal. Also, the number of LSTM units in a network is a parameter that is often manually selected. This approach turns out to be inefficient especially in new problems where the memory depth requirements are not clear.

In this work, NEAT (Neuroevolution of Augmenting Topologies) [24] algorithm is extended to incorporate LSTM cells (NEAT-LSTM). Since NEAT algorithm can evolve network topologies, it can discover the correct amount of memory units for the task. NEAT-LSTM outperform RNNs in two distinct memory tasks. However, NEAT-LSTM solutions do not scale as the memory requirement of the task increases. To overcome this problem, a secondary objective is used that maximizes the information stored in the LSTM units. The LSTM network is first evolved during the pre-training phase with this unsupervised objective to capture and store relevant features from the environment. Subsequently, during the task fitness optimization phase, the stored LSTM features are utilized to solve the memory task. This approach yields LSTM networks that are able to solve deeper memory problems.

Six sections follow this introduction. Section 2 provides

GECCO'16, July 20-24, 2016, Denver, Colorado, USA.
© 2016 ACM. ISBN TBA.
DOI: 10.475/123_4



Input Sequence	Expected NN Output
1 0 ... 0 -1 0 ... 0 1	0.5-1.0
-1 0 ... 0 -1 0 ... 0 1	0.0-0.5
1 0 ... 0 1 0 ... 0 1	0.5-1.0

Figure 1: (a) The LSTM unit architecture: A single LSTM unit is comprised of three components: (1) green-colored multiplication gates and sigmoid/tanh activation function, (2) yellow-colored internal memory state with linear activation, and (3) blue-colored peephole connections. There are three multiplication gates (write, forget and read) that control the flow of information through the LSTM unit. The peephole connections allow the internal memory state of the LSTM unit to be probed. The peripheral control logic (shown in gray cloud-shaped boxes) and the peephole connection weights are modified during the course of evolution. (b) The Sequence-Classification Task: This is a binary classification task where the network has to count whether the number of 1s in the input sequence exceeds the number of -1s (0s are ignored). At the start of evolution, the network consists of one input node and one output node. NEAT evolves hidden layer topology. A network requires memory in order to succeed in this task. Example of input-output sequence values are shown in the table.

a brief background on NEAT and LSTM and describes the related work in building agent memory. Two memory tasks are defined in Section 3. These tasks are used to compare the performance of different algorithms. In Section 4, NEAT is extended to include LSTM (NEAT-LSTM) and its performance is compared against RNN. The challenges in scaling the memory solutions are also analyzed and a new unsupervised objective is presented to solve this problem. Future directions are discussed in section 5 and section 6 provides a summary of the work.

2. BACKGROUND AND RELATED WORK

Recurrent neural network (RNN) can be used as policy controller or function approximators to store information from the past in their hidden states. However, scaling RNNs to problems that require deeper memory is challenging because RNNs can leak information over time. Recently, Long Short Term Memory (LSTM) has achieved success in solving supervised sequence processing tasks such as speech recognition [10] and machine translation [2]. LSTMs overcomes the limitation of RNNs by storing information in special linear unit (memory cells) and controlling their inflow and outflow through gating units.

2.1 LSTM

LSTM include three types of control gates: write control that determines the input to the memory state (with linear activation), forget gate that controls how much of the stored memory value is transferred to the next time step, and output gate which regulates the output of the memory cell. In addition, LSTM units can include extra peephole connections to probe the internal memory state. The peephole connections allow the LSTM gates to be modulated based on value stored in the internal memory state. The output activation function is tanh. Structure of a single LSTM unit is depicted in Figure 1a.

LSTM have been used as both policy controller and function approximator to solve RL problems. For example, LSTM based network was used as a function approximator in the robot navigation task [4]. In this work, the input sequence to LSTM was pre-processed to capture salient information from the environment. An unsupervised event extraction was performed by classifying stream of inputs into a variable number of distinct classes. Any change in input stream class is considered an event, and is fed into a RL model consisting of LSTM function approximator. One drawback of this method is that it can ignore sequential information that remains fixed during a trial but changes across trials. In [28], policy gradient algorithm was applied to train LSTM networks resulting in deeper memories for POMDP tasks. Bayer et al.[5] evolved custom LSTM memory cells using mutation operators. These custom cells are then manually instantiated to construct LSTM network for solving the T-maze problem. Their result suggests that evolving LSTMs can lead to interesting solutions to POMPD problems that are difficult to solve otherwise. Yet the evolved memory is not deep enough to be useful in real-world AI tasks (like deep T-maze or modeling hyena emotions). New methods are required to scale the evolution of LSTM to such tasks. Powerful neuroevolutionary techniques (like NEAT) is one

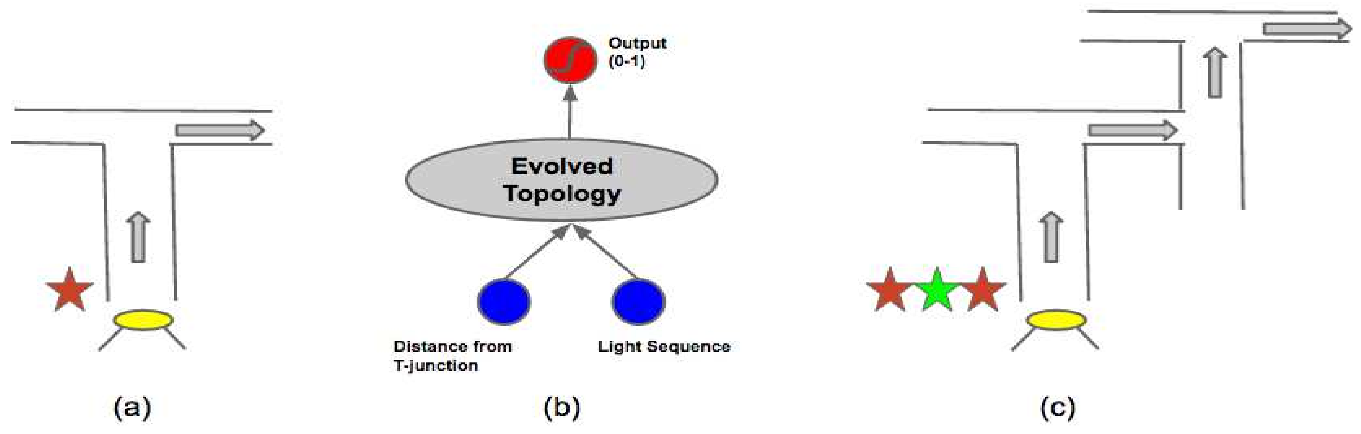


Figure 2: (a) Sequence Recall in simple T-maze: At the beginning of each trial, the agent observes one light. As the agent moves forward in the aisle towards the T-junction, it no longer has access to the light. The color of the light (red/green) indicates the direction (right/left) that the agent should take at T-junction in order to reach the goal. Therefore, to be successful, the agent needs to memorize the color of the light that was shown at the start. (b) Network architecture: In this task, the network has two inputs: one input represents the distance to the T-junction and the second input is the light sequence (active only during the first few time steps) (c) Deep T-maze: In the Deep T-maze, there are multiple input lights at the start corresponding to the multiple T-junctions. In order to be successful, the agent is required to recall the input light sequence in the correct order at each T-junction. This task requires deeper memory than the simple T-maze.

candidate approach to achieve complex memory solutions. NEAT can evolve both the topology and weights of a LSTM network in a non-parametric manner.

2.2 Problem of Deception

From an optimization perspective, since the problem of evolving memory is deceptive, extra objectives (to promote solution diversity) can be used to overcome deception [15, 18]. In such approaches, the evolutionary optimization problem is often cast as a multi-objective problem with two objectives - primary objective (task fitness) and secondary diverstiy objective (like novelty search). However, there are no guarantees that such diversity objectives can aid the learning algorithm to capture and store useful historical information from the environment. Since the secondary diversity objective is unrelated to the task fitness, the network can also undergo unsupervised pre-training to optimize this objective. One such unsupervised objective is presented in this paper that maximizes the total theoretic information stored in the LSTM network.

2.3 Unsupervised Training of LSTM

In supervised learning domain, unsupervised pre-training of neural networks to initialize its parameters has been shown to improve the overall network task performance significantly. This idea of pre-training the network with unsupervised objective can also be extended to the RL domain. However, not much literature exists on the topic of unsupervised pre-training of LSTM for RL POMDP tasks. On a related note, [13] performed unsupervised clustering of musical data using LSTM networks. In this work, Binary Information Gain Optimization (BINGO, [21]) was used as an unsupervised objective. BINGO maximizes the information gained from observing the output of a single layer network of logistic nodes, interpreting their activity as stochastic binary variables. One limitation of BINGO is that it searches only for uncorrelated binary features (thus the solution ends up having zeros and ones in equal parts), which limits the amount of information the network can store. Instead, LSTM features with maximal information are evolved in this work. Specifically, the LSTM networks are evolved (using NEAT) to first extract and store independent (and highly informative) real-valued features. These features are then later used to solve the memory task. Storing independent features in the LSTM ensures that the network has maximal information from a theoretic perspective. This information-theoretic objective to train LSTM network is similar to the info-max approach published in [6]. However, the main difference is that there is no underlying assumption on the number of mixed features and the linearity of the mixture. Next, the NEAT algorithm is described in brief.

2.4 NEAT

NEAT is a neuroevolution method that has been successful in solving sequential decision making tasks[24, 25, 15]. The evolved network is used as a policy controller that receives sensory inputs and outputs agent actions at each time step. NEAT begins evolution with a population of simple networks and complexifies the network topology into diverse species over generations, leading to increasingly sophisticated behavior. As new genes are added through mutations, they are assigned unique historical markings. During crossover, genes with the same historical markings are recombined to produce offsprings. The population of networks is divided into different species based on number of shared historical markings. Speciation protects structural innovations in networks by reducing competition among different species. The historical markings and speciation thus allow NEAT to construct complex task-relevant features. Memory can be introduced into the network by adding recurrent connections through mutations. NEAT is an ideal choice to evolve networks with memory in a non-parametric manner.

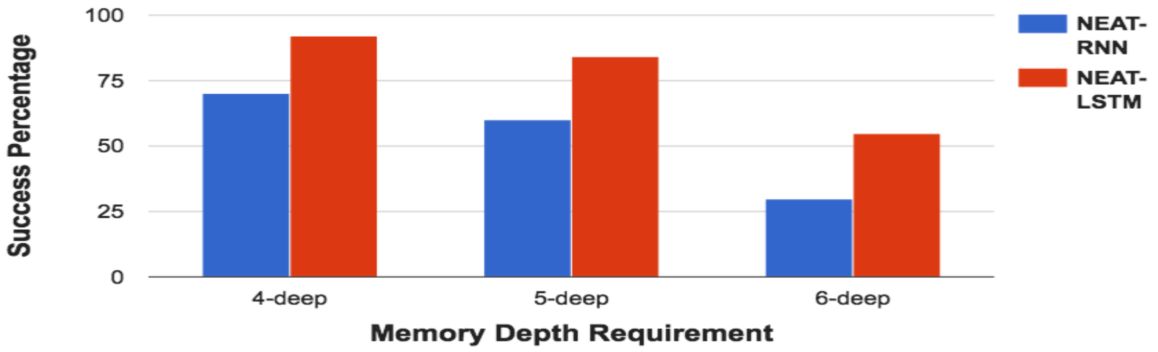


Figure 3: NEAT-LSTM vs. NEAT-RNN comparison on the Sequence-Classification task. The memory depth requirement is varied on the x-axis. The y-axis values represents the success rate of each method in 50 runs. The performance of NEAT-RNN and NEAT-LSTM is comparable for 4-deep sequence classification. As the task depth is increased to 6-deep, NEAT-LSTM significantly outperforms NEAT-RNN. Successful solutions to the sequence-classification task should be able to retain an aggregate of previous inputs and should also continuously update this aggregate with new incoming inputs. The performance results indicate that LSTM-based networks can memorize information over longer intervals of time as compared to RNNs.

3. MEMORY TASKS

This section describes two memory tasks. Both the tasks are situated in a discrete maze where the agent moves one-step at each time-step of the trial. The first task, sequence classification, is a binary classification problem of streaming input. The second task, sequence recall, requires the agent to recall a previously provided instruction input and use it to make future decisions (turn left or right at the T-junction). In both the tasks, the agent is required to store and to utilize the past events in order to be successful in the task.

3.1 Sequence Classification

This is a binary classification task, where given an input sequence of 1 and -1 (interleaved with 0s), the network needs to determine whether it received more 1s than -1s. The number of interleaved 0s in the input sequence is random and ranges between (10-20 time steps) . This task can be visualized as a maze positioning task. An agent situated at the center of a one-dimensional maze is provided instructions to move either left/right. It is expected to move left (west) when its input is -1 and move right (east) when its input is 1. When its input is zero, it is not expected to move. At the end of the sequence, the agent needs to identify whether it is on the right side of the maze or the left side i.e. has it taken mores steps towards east than west. Note, that input 0 does not affect this decision but only serves to confuse the agent. The number of turns (left/right) that the agent takes during a trial is defined as the depth of the task. For example, a sequence of four turns (total four 1/-1 inputs interleaved with 0 in the input sequence) is termed as 4-deep. Different sequence classification experiments are carried out by varying the depth of the task (4/5/6 deep). The network architecture and example input-output combinations are shown in Figure 1b.

3.2 Sequence Recall

In the simple T-Maze, at the beginning, an agent receives an instruction stimulus (like red/green light) (see Figure 2a). The agent then travel a corridor until it reaches the T-junction. At the junction, the path splits into two branches (left/right) and the agent needs to take the correct branch in order to reach the reward. The position of the reward is indicated by the instruction stimulus it received at the beginning. For example, red light can indicate presence of reward on the right-branch and green light can indicate the reward on left branch. A successful agent thus can only maximize collecting the reward by memorizing and utilizing its stimulus instruction. T-Maze has been widely used as a benchmark problem for evaluating agents with memory [3, 5, 15, 18]. This simple T-maze (with one T-junction) can then be extended to a more complex deep T-maze which consists of a sequence of independent T-junctions (Figure 2c.). Here, the agent receives a sequence of ordered instructions (one corresponding to each T-junction decision) at the start of trial and it has to utilize the correct instruction at every T-junction in order to reach the goal. Risi et al. [19] used one such T-maze extension (double T-maze) to test plastic neural networks. The distracted sequence recall task used in [17] is another variation of the deep T-maze recall, but it uses supervised training to learn the LSTM parameters. However, the approach presented in this paper uses a weak fitness signal (proportional to the number of correctly recalled input instructions) to train the memory network. Scaling the memory depth of network to recall long sequences in such RL settings has been a challenge.

4. EXPERIMENTS

In each experiment, a population of 100 networks is evolved using NEAT for 15,000 generations. Each individual in the population is evaluated on all possible input sequences. Deeper tasks therefore require more evaluations than shallow ones. The length of each trial also depends on the task-depth. For example, a 4-deep sequence classification task consists of at least four time steps (corresponding to four 1/-1) and 40 time steps of 0 inputs interleaved between 1/-1. During evolution, out of a total fitness of 100, the network receives a fraction for correctly predicting a part of the problem. For example, in a 4-deep sequence classification problem, if the agent correctly predicts its position in the maze (left or right side) after two turns, then it receives 50 fitness points. It is expected that such partial reward will

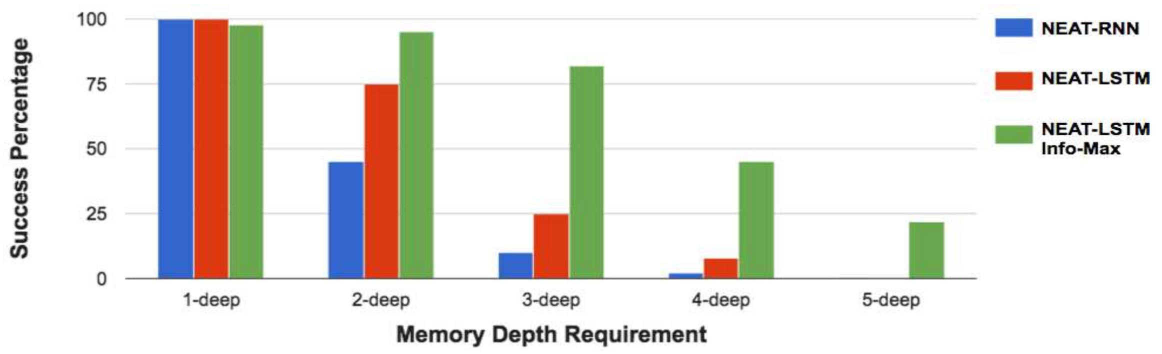


Figure 4: NEAT-LSTM, NEAT-RNN and NEAT-LSTM Info-max comparisons on Sequence-Recall task. Success percentage of each method is plotted for T-maze with varying depth. Both NEAT-RNN and NEAT-LSTM can quickly find the solution to simple one-step T-maze. As the maze becomes deeper, NEAT-LSTM outperforms NEAT-RNN. However, beyond 3-deep, the problem becomes too complex for both the methods. A solution to the deep T-maze problem requires memorizing the input light sequence in its correct order for several hundred time-step. NEAT-LSTM Info-max can successfully find solutions for even 5-deep recall. This suggests that pre-training of LSTM networks with unsupervised Info-max objective results in the capture of useful information from the environment that can later be used to solve the memory task.

shape the network towards evolving optimal behavior. Data is collected from 50 runs of each experiment type and the average success rate (percentage of runs that yield solution with maximum fitness) is measured to compare performance of different methods. At each time step, the network under evaluation is activated once. The value at the input of a node is propagated to its output in a single time step. Therefore, the number of time steps it takes for the network input to reach the output is equal to the shortest path between input and output. This setup is critical in ensuring that the network captures the input sequence values and their order correctly in the recall task. Further details on experimental setup, results and evolved network can be found on the demo page: http://nn.cs.utexas.edu/?deepmemory

4.1 Experiment 1: Comparing RNNs vs. LSTM

First, RNNs are evolved using standard NEAT algorithm (labeled as NEAT-RNN). A user-defined parameter controls the probability (set to 0.1 in these experiments) of adding a new recurrent link. The recurrent links can be either self-loops or a longer loop consisting of several intermediate nodes. Next, NEAT is extended to evolve LSTM-based networks that are compared with RNNs.

4.1.1 Method: NEAT-LSTM

Standard NEAT algorithm can add new nodes (sigmoid or rectified linear units) through mutation. In this work, NEAT is extended such that during its search process, it can add new LSTM units (probability of adding a new LSTM unit is 0.01). On being first instantiated, LSTM gates have default values - always write, always read and always forget. This setting ensures that initially, newly added LSTM units do not affect the functionality of the existing network. Each new instantiation of a LSTM unit is associated with the corresponding addition of a minimum of six network parameters - three connections from external logic to the control gates (depicted as cloud shaped gray boxes in Figure 1a) and three peephole connections (blue-colored links in Figure 1a). During the course of evolution, the existing parameters can be modified/removed and new ones can be added to suit the requirements of the task. No recurrent connections are allowed except the recurrency that exists within the LSTM unit.

4.1.2 Results

As shown in Figure 3, both NEAT-RNN and NEAT-LSTM can solve the 4-deep sequence classification task easily. As memory depth requirement increases to six, their success rate gradually decreases. NEAT-LSTM significantly outperforms NEAT-RNN in all the cases (t-test; $p<0.01$). The sequence-classification task is relatively simpler than the sequence-recall task. The agent is required to update and store its internal state with each new valid input (1/-1). Therefore, the successful networks have simple architecture (consisting of only a few recurrent neurons in the case of NEAT-RNN and a single LSTM in the case of NEAT-LSTM).

In the sequence-recall task, both NEAT-RNN and NEAT-LSTM quickly find the solution to one-step T-maze (Figure 4). This result is expected and it matches the outcomes of previous papers that focused on this problem ([15, 28]). However, as the T-maze becomes deeper, the successful solutions are required to store input light sequence information for longer durations and in its correct order. Therefore, beyond 3-deep T-maze, the problem becomes too complex for both NEAT-RNN and NEAT-LSTM to solve.

4.2 Experiment 2: Scaling NEAT-LSTM

In the harder task of sequence recall, the agent needs to store the entire input sequence in correct order. Incremental evolution approach was applied in order to solve deeper recall problems (>4-deep). Networks were first evolved for 2-, 3-, and 4-deep recall problems; these networks were then used as a starting point (in the NEAT algorithm) to solve the more complex problems of increasing depth (>4-deep). However, this approach did not yield much success. The problem may be that as the length of the input sequence increases, the number of parameters to be evolved also increase (with each additional LSTM units). Also, the incremental evolution approach requires detailed knowledge of the prob-

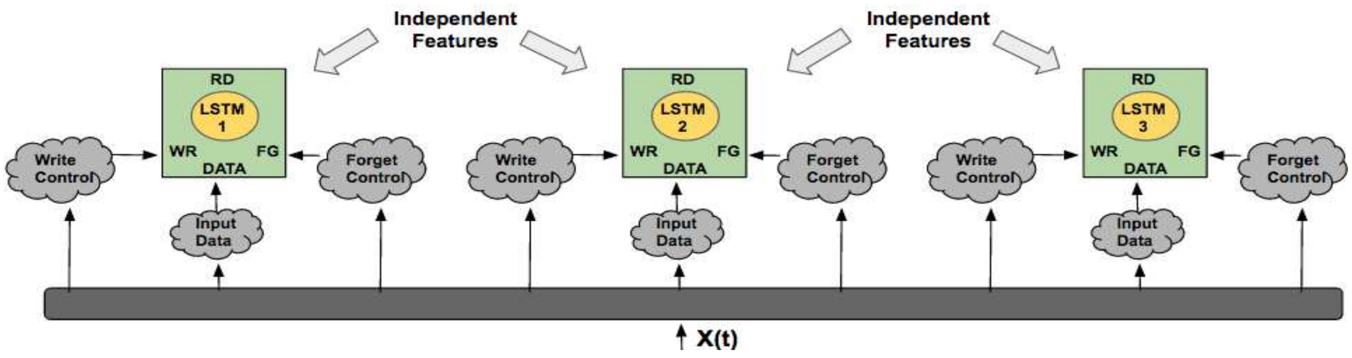


Figure 5: Unsupervised Pre-training using the Information Maximization objective. Highly informative, independent LSTM features are incrementally added by modifying the write control, forget control and input data logic using NEAT. Unsupervised pre-training is carried out until evolution stops discovering independent features or at the end of 10000 generations. By the end of this pre-training phase, salient information observed by the agent is captured and stored in the LSTM network. The pre-training aids evolution of deeper memory solutions in two ways. First, it reduces the problem of deception by directing evolutionary search towards landscapes that provide new information. Second, by incrementally adding independent features, it avoids the problem of training a large number of LSTM parameters simultaneously.

lem domain that is not inline with the goal of this paper (i.e. to solve the memory task with limited domain knowledge).

4.2.1 *Method: Information Maximization Objective*

One way to overcome the problem of deception is by ensuring that evolution discovers unique time-dependencies by not re-discovering existing information. In solving POMDP tasks that require memory, the agent can benefit by capturing salient historical information from the environment and storing it in its neural-network controller. As the agent moves in the environment over multiple trials, it comes across a lot of information. For example, it can observe the wall (which are mostly static across trials) or it can observe a binary light (red/green in T-maze), which can provide possible clues for the location of the goal. It is difficult to discern which information should be stored for later use (blinking light) and which should be discarded (static walls). One solution could be to store inputs (in their native form or in combination with other inputs) such that the total information stored in the network is maximized. From a theoretical perspective, the information stored in a set of random variables is maximized when their joint entropy is maximized. The joint entropy of two random variables X and Y can be defined as,

$$H(X, Y) = H(X) + H(Y) - I(X, Y) \tag{1}$$

$$H(X) = \sum P(X) \log P(X) \tag{2}$$

where $H(X)$, $H(Y)$ are the individual entropies of X and Y respectively and $I(X, Y)$ is the mutual information between X and Y. Thus, to maximize the information stored in the network, the individual entropy of its hidden unit activations should be maximized and their mutual information minimized. The mutual information can be expressed as Kullback-Leibler divergence:

$$I(X, Y) = \sum P(X, Y) \log \frac{P(X, Y)}{P(X)P(Y)} \tag{3}$$

Random variables with zero mutual information are statistically independent. Computing and storing highly informative, maximally independent features (i.e. features with high individual entropy and low mutual information) in the network is the unsupervised objective that will be used for NEAT (Info-max). The features are stored in the hidden LSTM units, since these units can retain information over several time steps. Evolving independent features allows for pre-training the network and only augments the NEAT-LSTM approach as outlined in Section 4.1.1. The new learning algorithm now consists of two steps: The first one is an unsupervised objective phase where the independence of LSTM hidden units is maximized (see Figure 5), and the second is an RL phase where the fitness objective is maximized (Figure 6).

A feature vector is constructed by concatenating the memory state values (yellow-colored circle in Figure 1) of the LSTM unit from each neural network activation across different trials. There is one distinct feature vector corresponding to each LSTM hidden unit. To compute entropy and mutual information, the feature vectors are treated as random variables. The real-valued features are partitioned into 10 equal-sized bins, and a histogram is constructed by counting the number of elements in each bin. Entropy and pairwise mutual information (an approximation of total mutual information) of feature histograms is then calculated using equation 2 and equation 3 respectively.

Evolving multiple independent features simultaneously can be challenging. This problem can be broken down using incremental evolution [9] (without the need for domain knowledge). Independent features (with their values stored in individual LSTM units) can be discovered one at a time using NEAT. Since the primary goal of this work is to build networks with memory, one simplifying assumption can be introduced: environment is not dynamic, i.e. it does not change over time. With this assumption, independent features once discovered and stored in LSTM need not change over a period of time. The stationarity assumption also entails that during the unsupervised training phase, only the write control logic, forget control logic and input data logic of the LSTM unit need to be modified. Once the independent features have been discovered, no more changes to the

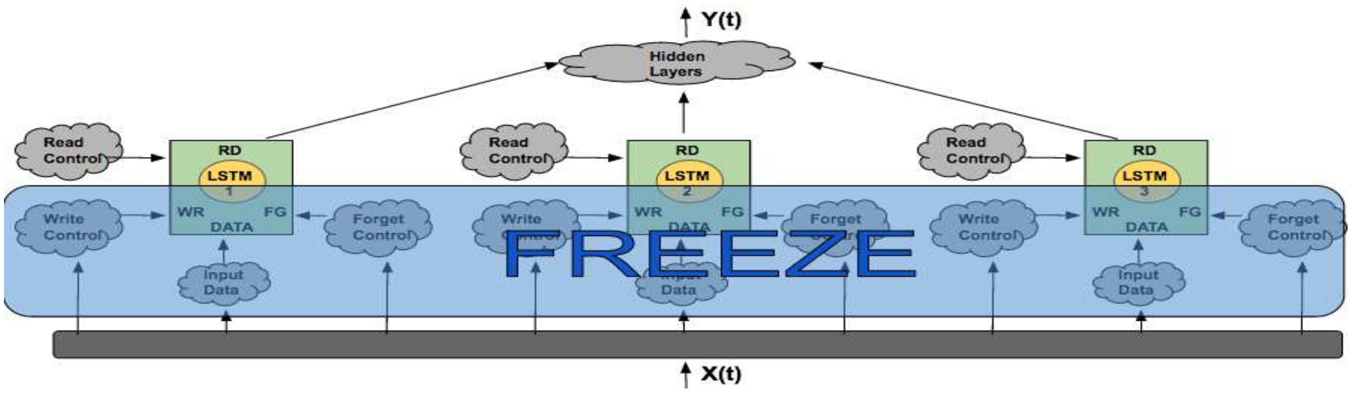


Figure 6: RL training using the fitness objective. During the RL phase, the stored/memorized features are utilized to solve the memory task. The write control, forget control and input data logic of the LSTM units (that store independent features) is frozen and the read control logic is evolved using NEAT. NEAT can add new hidden layers over the top of existing LSTM network.

write, forget and input logic of the LSTM units are required (i.e. they are frozen). For the remainder of evolutionary search, NEAT can only add outgoing connections from the frozen network (to facilitate re-use of frozen logic). Often, there exists multiple solutions to the problem of finding networks with maximum information. Some of these network solutions could be large. To bias NEAT towards evolving smaller solutions during unsupervised objective optimization, a regularization factor is introduced that penalizes larger networks. The size of the evolved network (equal to the number of network connections) is weighted by a regularization parameter (value varies between 0 and 1), and the resulting penalty term is subtracted from the unsupervised objective value. Unsupervised training is stopped either when NEAT cannot find any more independent features or at end of 10000 generations.

Subsequently, during the RL phase, the LSTM outputs are provided as extra inputs (in addition to the sensor inputs) to the NEAT algorithm. NEAT evolves the read control logic of the frozen LSTM units to utilize the stored features appropriately as deemed fit for the task. This approach makes neuroevolution computationally more tractable.

4.2.2 *Results*

NEAT-LSTM with the information maximization objective (NEAT-LSTM Info-max) was evaluated on sequence-recall task, since it is the harder of the two memory tasks. As shown in Figure 4, NEAT-LSTM and NEAT-RNN outperform NEAT-LSTM Info-max in the 1-deep task. This is probably because NEAT-LSTM and NEAT-RNN are powerful enough to quickly find a solution to the shallow memory problem. As the memory depth requirement increases however, NEAT-LSTM Info-max consistently outperforms NEAT-LSTM and NEAT-RNN (t-test; $p<0.01$). NEAT-LSTM Info-max is able to solve 5-deep sequence recall problem about 20% of the time.

5. DISCUSSION AND FUTURE WORK

In the sequence-recall task, the networks evolved using NEAT-LSTM Info-max method can preserve information (in correct order) over hundreds of time steps. This result suggests that pre-training of LSTM network with information maximization objective facilitates the capture of useful information from the environment. To confirm this hypothesis, pairwise mutual information was computed between LSTM feature vectors and the network inputs. Since each input light in the input sequence is independent of the other, maximum-information objective should often yield solutions such that different LSTM units capture information from distinct previous inputs. This was often found to be true. The info-max objective drives the network evolution towards gathering maximum possible information. Note that unlike novelty search, info-max is not open-ended since it depends on the richness of observable information in the environment. Further comparisons between Info-max and novelty search on memory tasks can be found at the demo page (link provided in Section 4) . While the idea of maximizing agent information is proposed to increase the depth of the agent's memory, it can indirectly lead to exploratory behaviors. Artificial curiosity [20] is one such concept, where the agent is explicitly rewarded for exploring areas in the environment that provide more information.

When neuroevolution is used to discover features, a rich feature set may accumulate. Recently, feature accumulation through evolution has been successful in solving both supervised learning problems and RL problem. For example, Szerlip et al. [26] used novelty search as an unsupervised objective to train HyperNEAT networks for MNIST digit recognition problem. Koutnik et al. [14] used a similar diversity objective (during unsupervised training) to train Convolutional Networks for simulated car racing. The Info-max objective presented in this paper is customized for memory tasks and therefore, should work better in POMDP problems. The idea of maximizing information gain for advancing the complexity of behaviors is biologically plausible as well [1].

6. CONCLUSIONS

Incorporating memory into artificial agents in a non-parametric manner is a challenging problem [8]. As a solution to this problem, LSTM networks are evolved using NEAT (NEAT-LSTM). NEAT discovers the appropriate number of LSTM units suitable for a given task. Evaluation on two memory tasks indicate that LSTM networks outperform RNNs in shallow POMDP tasks. To further scale the evolution of LSTM to deeper memory problems, a new information maximization (Info-max) objective is devised. The LSTM networks are pre-trained by optimizing for this unsupervised objective. During the course of pre-training over several generations, LSTM units incrementally capture and stored unique features from the environment. After this pre-training phase, the LSTM network is evolved to solve the memory task. The strong performance of NEAT-LSTM Info-max on deep sequence recall task indicates its utility in building generic AI agents that can solve both MDPs and POMDPs.

7. ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under grant DBI-0939454 and National Institutes of Health under grant 1R01GM105042.

8. REFERENCES

[1] C. Adami. The use of information theory in evolutionary biology,. *Annals NY Acad. Sci.*, 1256:49–65, 2012.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *In ICLR*, 2015.

[3] B. Bakker. Reinforcement learning with long short-term memory. In *Advances in Neural Information Processing Systems 14*, pages 1475–1482, 2002.

[4] B. Bakker, V. Zhumatiy, G. Gruener, and J. Schmidhuber. A robot that reinforcement-learns to identify and memorize important previous observations. In *In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2003*, pages 430–435, 2003.

[5] J. Bayer, D. Wierstra, J. Togelius, and J. Schmidhuber. Evolving memory cell structures for sequence learning. In *Proc. ICANN*, pages 755–764, 2009.

[6] A. J. Bell and T. J. Sejnowski. An information-maximisation approach to blind separation and blind deconvolution. *Neural Computation*, pages 1129–1159, 1995.

[7] J. Bruck. Decades-long social memory in bottlenose dolphins. *Proceedings of the Royal Society B: Biological Sciences*, 280, (2013).

[8] F. Doshi. The infinite partially observable markov decision process. In *NIPS,2009*, 2009.

[9] F. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342, 1997.

[10] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *In Proc. 31st ICML*, pages 1764–1772, (2014).

[11] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press*, 2001.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[13] M. Klapper-Rybicka, N. N. Schraudolph, and J. Schmidhuber. Unsupervised learning in lstm recurrent neural networks. In *ICANN*, pages 684–691. Springer-Verlag, 2001.

[14] J. Koutnik, J. Schmidhuber, and F. Gomez. Evolving deep unsupervised convolutional networks for vision-based reinforcement learning. In *In Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, (GECCO 2014)*, pages 541–548, 2014.

[15] J. Lehman and R. Miikkulainen. Overcoming deception in evolution of cognitive behaviors. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2014)*, Vancouver, BC, Canada, July 2014.

[16] G. Martin-Ordas, D. Berntsen, and C. J. Memory for distant past events in chimpanzees and orangutans. *Current Biology*, 23(15):1438–1441, (2013).

[17] D. D. Monner and J. A. Reggia. A generalized lstm-like training algorithm for second-order recurrent neural networks. *Neural Networks*, 25:70–83, 2012.

[18] C. Ollion, T. Pinville, and D. Stephane. With a little help from selection pressures: evolution of memory in robot controllers. In *In Artificial Life, volume 13*, pages 407–414, 2012.

[19] S. Risi, C. E. Hughes, and K. O. Stanley. Evolving plastic neural networks with novelty search. *Adaptive Behavior*, 18(6):470–491, 2010.

[20] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990-2010). In *IEEE Transactions on Autonomous Mental Development*, volume 2(3).

[21] N. N. Schraudolph and T. J. Sejnowski. Unsupervised discrimination of clustered data via optimization of binary information gain. *Advances in Neural Information Processing Systems*, 5:499–506, 1993.

[22] J. Schrum, I. Karpov, and R. Miikkulainen. Ut2: Human-like behavior via neuroevolution of combat behavior and replay of human traces. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG 2011)*, pages 329–336, 2011.

[23] K. O. Stanley, B. Bryant, and R. Miikkulainen. Evolving adaptive neural networks with and without adaptive synapses. In *Proceedings of the 2003 Congress on Evolutionary Computation*, Piscataway, NJ, 2003. IEEE.

[24] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[25] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[26] P. A. Szerlip, G. Morse, J. K. Pugh, and K. O. Stanley. Unsupervised feature learning through divergent discriminative feature accumulation. In *In Proc. of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[27] H. Watts and K. E. Holekamp. Interspecific competition influences reproduction in spotted hyenas. *Journal of Zoology*, 276(4):402–410, (2008).

[28] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber. Recurrent policy gradients. *Logic Journal of IGPL*, 18(2):620–634, (2010).