A MEMORY-AUGMENTED NEURAL NETWORK BY RESEMBLING HUMAN COGNITIVE PROCESS OF MEMORIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Memorization of long-term information is a core task in sequence learning of neural networks, and inspired by human cognitive process, we propose a sparse memory-augmented neural network (SMANN) to deal with it in this paper, which is composed of a two-layer neural controller and an external memory. In the first layer of the network, the information is divided into segments according to a sparse mask, which preserve immediate memory, and then in the second layer, the segmented information are collected and processed as short-term memory. For alleviation of gradient vanishing problem, constrained LSTM structures are utilized in both of the layers to make the chrono-initializer more reasonable. Lastly, the external memory is used to store long-term information, and its access rate is reduced sharply owing to the sparse mask. In experiments, we evaluate the network and its components like constrained LSTM and the neural controller independently, results on different tasks has demonstrated the superiorities of our networks compared with their counterparts.

1 INTRODUCTION

Capture of long-term dependencies is a core task in sequence learning, and recurrent neural networks (RNNs) have become the standard approach for addressing this learning problem. In one aspect, some RNNs models integrate historical information in the state parameters and propagate them along the cells in a serial order, such as the standard Long Short-Term Memory (LSTM) Lipton et al. (2015), the Gated Recurrent Unit (GRU) Cho et al. (2014) and the JANET Westhuizen & Lasenby (2018), which differ in cell structures and all employ information latching to alleviate gradient vanishing problem; In the other aspect, some adopt distinct network architectures, such as the hierarchical recurrent neural network El Hihi & Bengio (1996) which captures long-term dependencies in multiple time scales, the skip RNN Campos et al. (2017) which learns to skip redundant state updates and shortens the valid path length in long-term memory and the dilated recurrent neural networks Chang et al. (2017) which employs dilated skip connections to capture complex dependencies. Nevertheless, it is still difficult for these methods to learn and retain memory over long distance, and then models simulating human cognitive architecture have been developed recently and provided a promising potential in long-term sequential learning.

Memory-augmented neural networks (MANNs) consist of a neural controller and an external memory, which resemble human cognitive process roughly as we perceive information sequentially and process them in the brain by consulting memory Le et al. (2019). The MANNs models have been demonstrated to have superior performance over recurrent neural networks by many literatures Graves et al. (2016); Le et al. (2018). Despite the promising results, there remains some problems to be solved: (i) the network architecture should be refined so as to simulate human cognitive process more precisely; (ii) as access of external memory is expensive, the steps of reading and writing should be reduced while maintaining high performance.

Richard et.al Atkinson & Shiffrin (1968) proposed a modal model, in which human perceive information by sensory memory (immediate memory) firstly, and then the information selected by human attention will be transferred into short-term memory, lastly it can be stored into the cerebral cortex as semipermanent memory (long-term memory) if repeated for certain times. Besides, long-term

memory can be retrieved and reused in short-term memory Eichenbaum & Cohen (2004); Kandel et al. (2000). In this paper, we propose a sparse memory-augmented neural network (abbreviated as SMANN) to simulate this cognitive process. The network comprises a neural controller with two layers and a external memory: the first layer of the controller is divided into segments according to a sparse mask, in which each segment stores immediate memory; the second layer of the controller collects information from the segments, it integrates short-term memory into the state of the network and transfers processed information to external memory as long-term memory; reading and writing on the external memory are performed via attention mechanism on the mask, as it is sparse, it can reduce the memory access rate sharply. Besides, we propose to use a constrained cell structure in the neural controller to make chrono-initialization on the parameters more reasonable. The experiments are divided into three parts: improved LSTM with chrono-initializer, independent neural controller and the memory-augmented neural network. The results demonstrate the superiority of our models over other recurrent networks or the neural turing machine.

2 Model Architecture

The architecture of our proposed neural network is depicted in Fig.1, where the colored circles represent network cells in the neural controller and the squares represents binary elements in the sparse mask, M indicates the external memory. The first layer of the controller is broken by the mask and the segments store immediate memory; the second layer preserves short-term memory and its states are only updated at certain time steps according to the mask, meanwhile the external memory (long-term memory) is accessed simultaneously. In the following parts of this section, we present the cell structure and the initialization method firstly, and then the neural controller is illustrated in detail, lastly the access mechanism of the external memory is explained briefly.

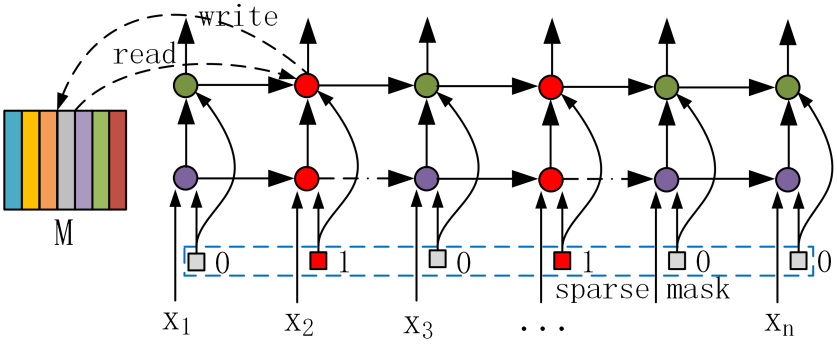


Figure 1: Architecture of SMANN.

2.1 Constrained LSTM with Chrono-initializer

In the neural controller, we adopt a LSTM cell with constrained output, which is accommodated to the chrono-initializer. Our proposed LSTM cell structure is formulated as below:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{h}_t &= \frac{D}{2}\left(\tanh(\mathbf{c}_t + \mathbf{u}) + \tanh(\mathbf{c}_t - \mathbf{u})\right) ,
\end{aligned} \tag{1}$$

where the symbols take the standard meaning as in Lipton et al. (2015), the variable $\mathbf{u}$ is a trainable vector with elements initialized as 0.5, and D is a trainable diagonal matrix with all of the diagonal elements initialized as 1. The standard procedure for LSTMs is to initialize the weights with uniform distribution, to initialize the forget gate bias as one and the rest biases as zero Jozefowicz et al. (2015). Nevertheless, as mentioned in Westhuizen & Lasenby (2018), the problem is that little information would be propagated through a long range of the sequence by this scheme. Thus,

Tallec et.al Tallec & Ollivier (2018) proposed a chrono-initializer, the motivation is to force the model's memory lying approximately in the same range with the sequence's temporal dependencies. Assuming the sequential data have temporal dependencies in a range $[T_l, T_r]$, we start by a leaky RNN, whose time-discretized equation is given by:

$$\mathbf{h}_{t+1} = \alpha \odot \tanh(\mathbf{U}\mathbf{h}_t + \mathbf{W}\mathbf{x}_t + \mathbf{b}) + (\mathbf{1} - \alpha) \odot \mathbf{h}_t \ , \tag{2}$$

which can be transformed to continuous time version by using first order Taylor expansion $h(t + \delta_t) \approx h(t) + \delta_t \frac{\mathrm{d}h(t)}{\mathrm{d}t}$ with time step $\delta_t = 1$:

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = \alpha \odot \tanh(\mathbf{U}\mathbf{h}(t) + \mathbf{W}\mathbf{x}(t) + \mathbf{b}) - \alpha \odot \mathbf{h}(t) \ . \tag{3}$$

As stated in Tallec & Ollivier (2018), the solution of Eqn.3 is $\mathbf{h}(t) = e^{-\alpha(t-t_0)}\mathbf{h}(t_0)$ with $\mathbf{x}(t) = 0$ for $t > t_0$, $\mathbf{b} = 0$ and $\mathbf{U} = 0$, So $\mathbf{h}(t)$ decreases to e^{-1} of $\mathbf{h}(t_0)$ over a time proportional to $T = 1/\alpha$, which can be called characteristic forgetting time of the leaky RNN. It is sensible to have T lying in the same range with the sequential data's temporal dependencies. Nevertheless, T is an unknown prior in LSTM networks, thus we use the gates $\mathbf{f}_t$ and $\mathbf{i}_t$ to learn the dominating parameters $1 - \alpha$ and α respectively. With the values of both inputs and hidden layers zero-centred over time, $\mathbf{f}_t$ will typically center around $\sigma(\mathbf{b}_f)$. Values of $\sigma(\mathbf{b}_f)$ in the desired range $[1 - \frac{1}{T_l}, 1 - \frac{1}{T_r}]$ can be obtained by initializing $\mathbf{b}_f$ as distributed as $\log(\mathcal{U}([T_l, T_r]))$ where $\mathcal{U}$ is the uniform distribution. For the same reason, $\mathbf{i}_t$ will fall into the range of $[\frac{1}{T_r}, \frac{1}{T_l}]$ when $\mathbf{b}_i = -\mathbf{b}_f$. Usually T_l is set to the shortest length of 1 and T_r set to the maximum length in the sequence, then we get:

$$\begin{aligned} \mathbf{b}_f &\sim \log(\mathcal{U}([1, T_{max} - 1])) \\ \mathbf{b}_i &= -\mathbf{b}_f \ , \end{aligned} \tag{4}$$

we can see that each component of $\mathbf{b}_f$ corresponds to one forgetting time of the network, it means that each component of $\mathbf{c}_t$ in Eqn.1 preserves information for a length of the forgetting time. However, the temporal dependencies of the sequence data will not distribute uniformly in the range of sequence length, thus we use a double tanh function to make constraint on the hidden state originated from $\mathbf{c}_t$, which can fitter out useless components and assign weights on the rest ones. Experiments are carried out to provide insights and understanding of this improvement in subsection 4.1.

2.2 Neural Controller with Sparse Mask

The mask (denoted as m) in the neural controller is a one-dimensional vector with binary elements, and it is used to segment the first layer of the controller and determine the state update of the second layer and access of the external memory. If the mask element at time step t equals to 1, the cell connection in the first layer is forced to be broken, and the last two vectors in Eqn.1 change into this formula:

$$\begin{aligned} \mathbf{c'}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c) \\ \mathbf{h}_t &= \frac{D}{2}\left(\tanh(\mathbf{c'}_t + \mathbf{u}) + \tanh(\mathbf{c'}_t - \mathbf{u})\right) \\ \mathbf{c}_t &= (1 - \mathrm{m_t}) \odot \mathbf{c'}_{\mathrm{t}} \ . \end{aligned} \tag{5}$$

Simultaneously, the cell state in the second layer is updated, $\mathbf{c}_t$ and $\mathbf{h}_t$ evolve as:

$$\begin{aligned} \mathbf{c'}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c) \\ \mathbf{c}_t &= \mathrm{m_t} \odot (\mathbf{c'}_{\mathrm{t}}) + (1 - \mathrm{m_t}) \odot \mathbf{c}_{\mathrm{t}-1} \\ \mathbf{h}_t &= \frac{D}{2}\left(\tanh(\mathbf{c}_t + \mathbf{u}) + \tanh(\mathbf{c}_t - \mathbf{u})\right) \ , \end{aligned} \tag{6}$$

where the states are responsible for short-term memory as the information is difficult to be propagated along extreme long distance on account of the vanishing gradient. The mask is generated by the formula as below:

$$\mathrm{m} = f_{binary}(\sigma(\mathrm{m'})) \ , \tag{7}$$

where σ symbolizes sigmoid function, and f_{binary} is a deterministic step function which rounds a vector with elements in $(0, 1)$ to a vector in $\{0, 1\}$. As f_{binary} is a non-differentiable step function,

we use the straight through (ST) estimator to compute the gradients, which is a biased estimator firstly proposed by Hinton et al. Hinton (2012); Bengio et al. (2013). When m' is initialized as zero vector, it will result in a initial mask with all elements equal to 1, which means that the first layer of the controller is broken into segments at every time step and the second layer collects information from below and updates its cell states at the same time steps. Next, in the training process, cells in the first layer will group into lots of clusters as zeros arise in the mask. Finally, if the mask is sparse, several clusters appear and the number of update operation in the second layer along with the access number to the external memory will be reduced sharply. The mask is forced to be sparse by adding a term of Lasso regularization Zhao & Yu (2006); Donoho & Elad (2003) to the loss of the model.

$$L = L_1 + \varepsilon \ell_1(\mathrm{m}) \,, \tag{8}$$

where L symbolizes the total loss, L_1 represents error between outputs and the ground truth, such as cross entropy in classification applications, ℓ_1 denotes ℓ_1-norm, and ε is a scalar set to $1e-6$. In subsection 4.2, the experimental results demonstrate that this simple segmented neural network outperforms other state-of-the-art models, and for convenience, it is abbreviated as SNN.

2.3 External Memory

The external memory is used for preserving long-term memory in our works. As mentioned before, access of the external memory happens at the time steps corresponding to non-zero elements in the sparse mask. The read vector is a convex combination of the row-vectors in the memory, and write comprises two parts: erase and add Graves et al. (2016). The addressing mechanism adopted in our implementation is content-addressing, in which the content-based system produces a normalised weighting based on cosine similarity. At an access time step t, the read vector is concatenated with the output of the neural controller ($\mathbf{h}_t$ of the second layer), and then passes through a full connected layer to get a final output; in the other case that external memory is not accessed, the outputs are copied just form the previous time steps. Experimental results in subsection 4.3 indicate that the access rate has been reduced sharply by our designed neural controller.

3 Theoretical Analysis

3.1 Advantage in Memory Capacity

The recurrent length is proposed and used to measure the memory capacity of recurrent neural networks Zhang et al. (2016); Chang et al. (2017). The network is better in memory capacity when the mean path length spanning all time steps is shorter. We use $d_i(n)$ to denote the path length, where i is the time index of input node and n means that the output node is located at time step $i+n$. In a regular two-layer recurrent neural network, where the cells are connected in sequence without break, we have $[d_i(n)]_{\mathrm{RNN}} = n$ apparently. However, in our network (SMANN), $[d_i(n)]_{\mathrm{SMANN}}$ may be smaller than n if one or more time steps for updating state and accessing external memory (the red nodes in Fig.1) locate between i and $i+n$. Then, we adopt a mean value as below:

$$\overline{d} = \frac{1}{\ell} \sum_{n=1}^{\ell} \max_{i \in V} d_i(n) \,, \tag{9}$$

where ℓ is the sequence length, V is a set comprising all time steps. For the regular recurrent neural network, we have $[\overline{d}]_{\mathrm{RNN}} = \frac{1+\ell}{2}$, and for SMANN $[\overline{d}]_{\mathrm{SMANN}} \leq \frac{1+\ell}{2}$, it means that the memory capacity of our network has been improved compared with the RNN network.

In another aspect, for RNN the gradient of the objective function is given by:

$$\left[\frac{\partial L}{\partial \mathbf{c}_\tau}\right]_{\mathrm{RNN}} = \frac{\partial L}{\partial \mathbf{c}_T} \prod_{k=\tau}^{T-1} \frac{\partial \mathbf{c}_{k+1}}{\partial \mathbf{c}_k} \,. \tag{10}$$

For SMANN, in the second layer, we have $\frac{\partial \mathbf{c}_{k+1}}{\partial \mathbf{c}_k} = 1$ if the state is not updated according to zero element of the mask, then the gradient can be formulated as:

$$\left[\frac{\partial L}{\partial \mathbf{c}_\tau}\right]_{\mathrm{SMANN}} = \frac{\partial L}{\partial \mathbf{c}_T} \prod_{k=\tau, m(k)=1}^{T-1} \frac{\partial \mathbf{c}_{k+1}}{\partial \mathbf{c}_k} \,, \tag{11}$$

As pointed out by Bengio et al. Bengio et al. (1994); Hochreiter et al. (2001), the norm of each partial derivative must be less than 1 in order to latch information robustly. Thus, we have:

$$\frac{\left[\frac{\partial L}{\partial \mathbf{c}_\tau}\right]_{\mathrm{RNN}}}{\left[\frac{\partial L}{\partial \mathbf{c}_\tau}\right]_{\mathrm{SMANN}}} = \frac{\frac{\partial L}{\partial \mathbf{c}_T}\prod_{k=\tau}^{T-1}\frac{\partial \mathbf{c}_{k+1}}{\partial \mathbf{c}_k}}{\frac{\partial L}{\partial \mathbf{c}_T}\prod_{k=\tau,m(k)=1}^{T-1}\frac{\partial \mathbf{c}_{k+1}}{\partial \mathbf{c}_k}} = \prod_{k=\tau,m(k)=0}^{T-1}\frac{\partial \mathbf{c}_{k+1}}{\partial \mathbf{c}_k}\,, \tag{12}$$

As we know, the trained mask m is sparse. When $T-\tau$ increases, Eqn.12 approaches 0, which means that the gradients of SMANN vanish slower than that of RNN and can be propagated along a further distance. Eventually, the memory capacity can be improved by this neural architecture.

3.2 Sparsity of the Mask

As mentioned before, the mask is used to divide long-term memory into immediate memory fragments. Assuming that the non-zero elements distribute averagely in the mask, the distance among them is denoted as x, then we have:

$$\begin{aligned}
&\max_{i\in V} d_i(n) = x + \lfloor\frac{n}{x}\rfloor - 1 \\
&\overline{d} = \frac{1}{\ell}\sum_{n=1}^{\ell}\max_{i\in V} d_i(n) = \frac{1}{\ell}\sum_{n=1}^{\ell}\Big(x + \lfloor\frac{n}{x}\rfloor - 1\Big) \\
&\quad\approx \frac{1}{\ell}\sum_{n=1}^{\ell}\Big(x + \frac{n}{x} - 1\Big) = x + \frac{\ell+1}{2x} - 1\,,
\end{aligned} \tag{13}$$

where $\lfloor . \rfloor$ is a floor function. When $x = \sqrt{\frac{\ell+1}{2}}$, $\overline{d}$ is minimized, which means that the network achieves the best ability of memorization by this setting of the mask. As for the MNIST dataset with period of $\ell = 784$, the optimal value of x approximates 20. The experimental results in subsection 4.2 and 4.3 show that the non-zero elements in the resulted mask is approximately uniform (will be influenced by the content of sequence data) and the number equals to a near-optimal value.

4 Experiments

The performance evaluation is conducted mainly on four kinds of tasks: sequential classification, copy, add and frequency discrimination. The dataset of MNIST comprising $60,000$ training samples and $10,000$ testing ones, in which each sample is reshaped as an 784-dimensional row vector. Also a permuted MNIST (pMNIST) is adopted, as stated by Arjovsky et al. Arjovsky et al. (2016), the pixels are permuted to lengthen temporal dependencies and increase difficulty for recurrent neural networks to learn.

In the add task, the networks are feed with sequences of $(value, marker)$ tuples, and aim to generate addition of the values marked with 1 while ignoring others marked with 0. The synthetic dataset is composed of $50,000$ sequences in which 70 percents are used for training, 10 percents are for validation and the rest is for testing. The values of each sequence are drawn from $\mathcal{U}[-0.5, 0.5]$ with $\mathcal{U}$ denotes uniform distribution, and the first marker is randomly placed among the first 10% of the length and the second one is placed among the last 10%, thus the useful information for adding has interval of at least 80% of the total length, which is beneficial for testing of long-term memory. Mean Squared Error (MSE) between the output and the ground truth is adopted as loss function.

In the copy task, $50,000$ sequences are generated and divided as training, validation and testing sets by the same proportion as in the add task. The sequence includes $I + 21$ items in which each indicates a category selected from $\{a_i\}_{i=0}^{9}$. Specifically, the first 10 items are used for copy and sampled uniformly from $\{a_i\}_{i=1}^{8}$, and then the following I items are filled with zeros, next, the item located at $I + 11$ is set to 9, which acts as a delimiter telling the network starts to copy the first ten items, finally, the last 10 items are also filled with zeros. Consequently, the ground truth contains $I + 11$ items of zeros and ten ones copied from the head of the input sequence. The networks aim to minimize the average cross entropy of the category predictions.

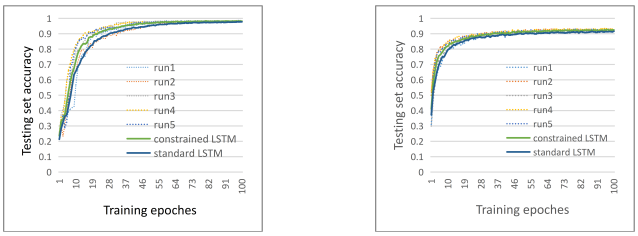
The task of frequency discrimination is to discriminate the sinusoids with two kinds of periods. Different from the experimental setting in Campos et al. (2017), where the data is generated on

the fly, our constructed dataset has $30,000$ sequences drawn from sinusoids with length of certain milliseconds, in which half of the sinusoids have periods P in range of $\mathcal{U}(1,2)$ milliseconds and the other half have periods in range of $\{\mathcal{U}(0,1) \cup \mathcal{U}(2,3)\}$ milliseconds Neil et al. (2016); Campos et al. (2017). Moreover, each sequence has a random phase shift drawn from $\mathcal{U}(0,P)$. As sinusoid is continuous signal, the amplitudes are sampled and saved at intervals of 1.0 milliseconds.

In the following experiments, all the networks are trained by Adam Kingma & Ba (2014) optimizer with a learning rate of 0.001 for 100 epoches, dropout rate of 0.1 is used on the output of networks, and the batch size is set to 200. Moreover, the model parameters with the minimum validation loss are saved and used for testing.

4.1 Single-layer Constrained LSTM

Here the constrained LSTM is compared with the standard LSTM Lipton et al. (2015) to verify its improvement in performance, and the two networks are both composed of a single layer of 128 units. The comparison is performed in sequential classification, add and copy tasks. The mean values and standard deviations of testing accuracies [%] on MNIST and pMNIST are exhibited in Table.1. For the add and copy tasks, comparisons are conducted on sequences with length of 120, and losses of MSE and cross entropy are also displayed as form of $\mathrm{mean} \pm \mathrm{std}$. All the results are generated by five different runs. The resulted curves on the MNIST and pMNIST datasets are depicted in Fig.2, where the different results of the constrained LSTM over five runs are drawn with dashed lines and the averaging results of two compared networks are drawn with solid lines. Apparently, our proposed LSTM works better than the standard version.



(a) MNIST (b) pMNIST

Figure 2: Testing accuracies of single-layer LSTMs over training epochs.

Table 1: Results of testing accuracies and MSE[a].

Model \ Task	MNIST	pMNIST	Add	Copy
Standard LSTM[b]	97.4 ± 0.49	91.3 ± 0.55	2.34e-5±5.34e-6	0.1246±0.0172
Constrained LSTM[b]	98.2 ± 0.33	92.4 ± 0.50	2.21e-5±6.55e-6	0.1138±0.0125

[a] In the tables of this paper, 1e-n indicates 1×10^{-n}.
[b] a single layer of 128 units is used.

4.2 Independent Neural Controller

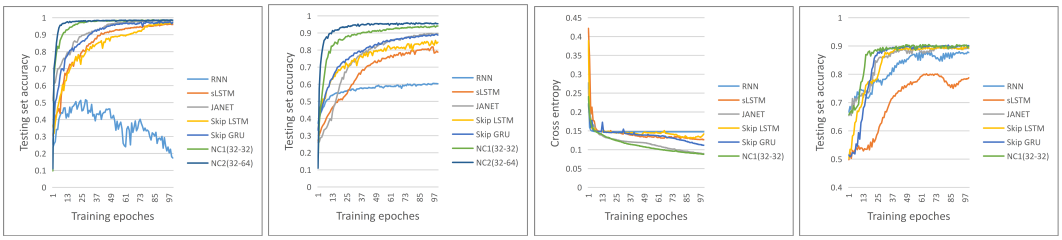
In this part, the network of neural controller is evaluated independently from the assistance of external memory. The comparative networks of the RNN, standard LSTM, JANET, skip LSTM and skip GRU all have two hidden layers of 32 units, and their results are obtained by five independent runs. For other networks, we cited the stated results in the papers directly, and all of them have one single layer of 128 units unless specially stated. The testing accuracies on the MNISTs are presented in Table.2, where results in the upper part are displayed in the form of $\mathrm{mean} \pm \mathrm{std}$, and others are cited mean values. Our neural controller (NC_1) with two hidden layers of 32 units behaves better than the comparative models except for the tLSTM and BN-LSTM, which both have much more amount of parameters than NC_1. Assume that a LSTM layer has n_1 inputs and n_2 hidden units, then we

Table 2: Results of testing accuracies.

Model \ Dataset	Accuracy on MNIST	Accuracy on pMNIST
RNN	54.9 ± 5.56	60.6 ± 22.76
Standard LSTM Lipton et al. (2015)	96.1 ± 0.99	80.3 ± 1.14
JANET Westhuizen & Lasenby (2018)	97.9 ± 0.24	89.9 ± 0.66
Skip LSTM Campos et al. (2017)	96.8 ± 0.39	85.8 ± 0.86
Skip GRU Campos et al. (2017)	97.6 ± 0.33	89.2 ± 0.73
NC_1 (32-32)	98.6 ± 0.21	94.4 ± 0.45
NC_2 (32-64)	99.0 ± 0.12	**95.6** ± 0.28
tLSTM[a] He et al. (2017)	**99.2**	94.9
BN-LSTM[b] Cooijmans et al. (2016)	99.0	95.4
sTANH-RNN Zhang et al. (2016)	98.1	94.0
uRNN Arjovsky et al. (2016)	95.1	91.4
iRNN Le et al. (2015)	97.0	82.0
TANH-RNN Le et al. (2015)	35.0	35.0

[a] actually three layers with 100 hidden units are used.
[b] one single layer with 100 hidden units.

have $\mathbf{x}_t \in \mathbb{R}^{n_1}$, $\{\mathbf{c}_t, \mathbf{h}_t\} \in \mathbb{R}^{n_2}$, $\mathbf{W}_j \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{U}_j \in \mathbb{R}^{n_2 \times n_2}$ and $\mathbf{b}_j \in \mathbb{R}^{n_2}$. As for a single layer standard LSTM, we have $j = \{i, o, f, c\}$ corresponding to four fully connected layers and the total number of parameters is $4(n_1 n_2 + n_2^2 + n_2)$; in our neural controller, $j = \{i, f, c\}$ as the output gate is omitted. Then it is not too hard to calculate that the three-layer tLSTM with 100 hidden units has roughly 2×10^5 parameters, the BN-LSTM with the same hidden units approximately has 4×10^4 and the two-layer NC_1 has no more than 1×10^4. Additionally, the neural controller can perform better with larger layer sizes, the two-layer NC_2 ranks the same with BN-LSTM on the MNIST and performs much better than others on the pMNIST while its parameter number is roughly 2×10^4. The resulted curves on the MNISTs are depicted in the left and middle parts of Fig.3, apparently we can see that the accuracy rates of our neural controller climb more shapely than others and achieve the highest values eventually.



(a) MNIST (b) pMNIST (c) Copy task (d) Frequency

Figure 3: Testing accuracies on MNISTs and frequency data over training epoches are depicted in (a,b) and (d) respectively, and curves of cross entropy in copy task are drawn in (c).

Experimental results on another three tasks are presented in Table.3, and also the curve results of copy and frequency discrimination tasks are provided in Fig.3(c,d). Our neural controller achieves the best in add and copy, and behaves similarly with some of others on the frequency data.

4.3 Sparse Memory-augmented Neural Network

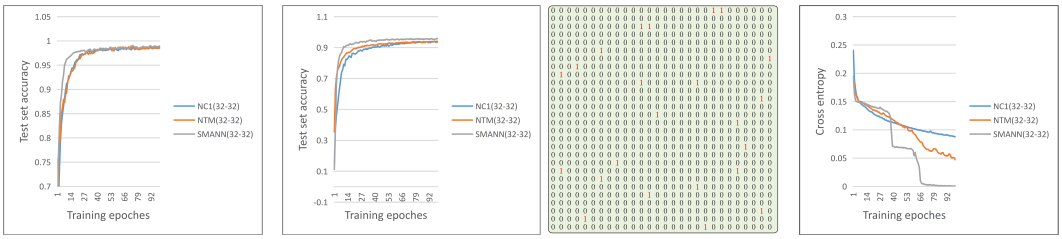
We evaluate our designed neural network with external memory (SMANN) here. The neural controller (NC) and a neural turing machine (NTM) are adopted for comparison, which both comprise neural networks of two layers with 32 hidden units. The results on the MNISTs are depicted in Fig.4, where SMANN has a faster ascent curves and achieves better accuracies than others. Fig.4(c) is a sparse mask (with 23 nonzero elements) generated on the pMNIST, which indicates that the access

Table 3: The experimental results of add, copy and frequency discrimination tasks.

Model \ Task	Add (len[a] = 120)	Copy (len = 120)	Frequency (len = 120)
RNN	7.40e-4 ± 3.35e-4	0.1475 ± 0	86.4 ± 1.42
sLSTM	3.29e-5 ± 1.02e-5	0.1263 ± 7.81e-4	81.2 ± 6.65
JANET	1.08e-5 ± 1.10e-6	0.0886 ± 1.66e-3	89.8 ± 0.39
Skip LSTM	2.88e-5 ± 2.71e-6	0.1296 ± 1.27e-3	90.1 ± 0.36
Skip GRU	1.76e-5 ± 1.05e-5	0.1147 ± 1.51e-2	90.4 ± 0.26
NC_1	9.32e-6 ± 2.26e-6	0.0873 ± 3.61e-3	90.4 ± 0.23

[a] len indicates the sequence length.

rate to the external memory is reduced sharply to nearly 3 percents compared with a neural turing machine, in which the memory access is needed in every time step. Fig.4(d) is a resulted curves on copy task, and SMANN has a significant improvement compared with others. The evaluation between NTM and SMANN are presented in Table.4, SMANN behaves better than NTM apparently, but they both perform not very well in add task compared with models without external memory, we will explore it deeply in the future.



(a) MNIST (b) pMNIST (c) Mask on pMNIST (d) Copy Task

Figure 4: Testing accuracies on MNISTs and cross entropy in the copy task.

Table 4: Comparison of NTM and SMANN.

Model \ Task	MNIST	pMNIST	Add (len = 120)
NTM(32-32)	98.6 ± 0.17	93.6 ± 0.26	7.04e-5 ± 5.49e-6
SMANN(32-32)	98.9 ± 0.14	95.9 ± 0.23	6.89e-5 ± 4.02e-6
– –	Copy (len = 120)	Frequency (len = 120)	Frequency (len = 500)
NTM(32-32)	0.0468 ± 3.98e-2	90.7 ± 0.32	86.9 ± 0.33
SMANN(32-32)	5.27e-4 ± 1.02e-3	91.0 ± 0.27	89.3 ± 0.31

5 Conclusion

In this paper, we propose a sparse memory-augmented neural network to simulate human memory process. In the bottom layer of the neural controller, information is divided into segments via a mask, and then short-term information are collected and memorized in the top layer, constrained LSTMs are utilized in both of the layers in consideration of chrono-initializer. The external memory is accessed according to the sparse mask and the access rate is reduced to about 3 percents compared to conversional neural turing machine, which can accelerate the speed of training and running the whole network as access to external memory is time-consuming. Experiments are conducted as three parts: the constrained LSTM, the neural controller and the complete model. The two components are demonstrated to behave better than their counterparts, and the model with an external memory improves further and outperforms the neural turing machine with less memory access.

REFERENCES

Martín Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 1120–1128, 2016. URL http://jmlr.org/proceedings/papers/v48/arjovsky16.html.

R. C. Atkinson and R. M. Shiffrin. Human memory: A proposed system and its control processes 1. *Psychology of Learning & Motivation*, 2:89–195, 1968.

Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*, 2017.

Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 77–87, 2017.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Tim Cooijmans, Nicolas Ballas, César Laurent, and Aaron C. Courville. Recurrent batch normalization. *CoRR*, abs/1603.09025, 2016. URL http://arxiv.org/abs/1603.09025.

David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ?1 minimization. *Proceedings of the National Academy of Sciences*, 100(5): 2197–2202, 2003.

Howard Eichenbaum and Neal J Cohen. *From conditioning to conscious recollection: Memory systems of the brain*. Number 35. Oxford University Press on Demand, 2004.

Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems*, pp. 493–499, 1996.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538 (7626):471, 2016.

Zhen He, Shaobing Gao, Liang Xiao, Daxue Liu, Hangen He, and David Barber. Wider and deeper, cheaper and faster: Tensorized lstms for sequence learning. In *Advances in neural information processing systems*, pp. 1–11, 2017.

Geoffrey Hinton. Neural networks for machine learning. *Coursera video lecture 9c*, 2012.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on International Conference on Machine Learning*, 2015.

Eric R Kandel, James H Schwartz, Thomas M Jessell, Department of Biochemistry, Molecular Biophysics Thomas Jessell, Steven Siegelbaum, and AJ Hudspeth. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Hung Le, Truyen Tran, Thin Nguyen, and Svetha Venkatesh. Variational memory encoder-decoder. In *Advances in Neural Information Processing Systems*, pp. 1508–1518, 2018.

Hung Le, Truyen Tran, and Svetha Venkatesh. Learning to remember more with less memorization. *arXiv preprint arXiv:1901.01347*, 2019.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941, 2015. URL http://arxiv.org/abs/1504.00941.

Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in neural information processing systems*, pp. 3882–3890, 2016.

Corentin Tallec and Yann Ollivier. Can recurrent neural networks warp time? *International Conference on Learning Representations*, 2018.

Jos Van Der Westhuizen and Joan Lasenby. The unreasonable effectiveness of the forget gate. *CoRR, abs/1804.04849, 2018.*, 2018. URL http://arxiv.org/abs/1804.04849.

Saizheng Zhang, Yuhuai Wu, Tong Che, Zhouhan Lin, Roland Memisevic, Ruslan Salakhutdinov, and Yoshua Bengio. Architectural complexity measures of recurrent neural networks. *CoRR*, abs/1602.08210, 2016. URL http://arxiv.org/abs/1602.08210.

Peng Zhao and Bin Yu. On model selection consistency of lasso. *Journal of Machine learning research*, 7(Nov):2541–2563, 2006.