Writing to the Hopfield Memory via Training a Recurrent Network

Han Bao[1,2], Richong Zhang[1,2(✉)], Yongyi Mao[3], and Jinpeng Huai[1,2]

[1] SKLSDE, School of Computer Science and Engineering, Beihang University, Beijing, China
[2] Beijing Advanced Institution on Big Data and Brain Computing, Beihang University, Beijing, China
{baohan,zhangrc}@act.buaa.edu.cn, huaijp@buaa.edu.cn
[3] School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada
ymao@uottawa.ca

Abstract. We consider the problem of writing on a Hopfield network. We cast the problem as a supervised learning problem by observing a simple link between the update equations of Hopfield network and recurrent neural networks. We compare the new writing protocol to existing ones and experimentally verify its effectiveness. Our method not only has a better ability of noise recovery, but also has a bigger capacity compared to the other existing writing protocols.

Keywords: Hopfield network · Writing protocol · Recurrent network

1 Introduction

Different from conventional memories, associative memories [1] store information in a distributed manner and without an addressing mechanism. With associative memories, message retrival is content based, which resembles a "recalling" process in response to a query that may represent a partial/noisy version of a stored message. E.g., if the message "The capital of France is Paris" is stored in the memory (superimposed with other messages), then by feeding the memory with "The capital of France is Lyon", the original message may be recovered. Such content-addressable memories are seen as closer to the way human brains store and retrieve information and are arguably an important research subject in artificial intelligence.

Among associative memory models, Hopfield network [9] was conceptualized in 1982. In application, Hopfield neural network was appeared on different fields,

This work is supported partly by China 973 program (No. 2015CB358700), by the National Natural Science Foundation of China (No. 61772059, 61421003), and by the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC) and State Key Laboratory of Software Development Environment (No. SKLSDE-2018ZX-17).

A. C. Nayak and A. Sharma (Eds.): PRICAI 2019, LNAI 11671, pp. 241–254, 2019.
https://doi.org/10.1007/978-3-030-29911-8_19

like image processing and optimization [3], economic load dispatch [13], classification task [18], digit recognition [5], social network [4]. Besides, the theoretical studies [7,15,19] and model modification like [11,12,16] have attracted many research interests.

An N-neuron Hopfield network [9] is an undirected weighted graph on N nodes, where each node is referred to as a neuron [14], and the (symmetric) weight matrix W specifies the connectivity strength between neurons. (Negative weights are allowed.) Each neuron i is associated with a state variable x_i taking values in $\{\pm 1\}$. The network has a state-update function $f_W\colon \{\pm 1\}^N \to \{\pm 1\}^N$, defined as

$$f_W(\mathbf{x}) = \text{sign}(W\mathbf{x}), \tag{1}$$

where the sign function is acting element-wise on its input vector, i.e., for all $i \in \{1, \ldots, N\}$, the i-th entry of $\text{sign}(\mathbf{r})$ is equal to 1 if $r_i > 0$ and is equal to -1 otherwise.[1] Using the state-update function, the network updates its state $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ according to

$$\mathbf{x}^{\text{new}} = f_W(\mathbf{x}^{\text{old}}) \tag{2}$$

As an associative memory, a Hopfield network stores a message set $\mathcal{M} = \{\mathbf{x}^1, \ldots, \mathbf{x}^M\} \subseteq \{\pm 1\}^N$ by encoding the message in the weight matrix W. To retrieve a message, when presented with a query/probe $\mathbf{p}$, the network initializes its state vector as $\mathbf{x} = \mathbf{p}$, iterates the update Eq. (2) until convergence or a maximum number of iteration is reached, and declares the state vector as the retrieved message. Usually, the probe $\mathbf{p}$ is a noisy/partial version of one of the message set $\mathcal{M}$ and it is desirable that the retrieved message is equal to the correct message, even when the probe is a severely corrupted version of the message.

The problem of *writing to a Hopfield memory* is then that of designing a weight matrix W for a given message set $\mathcal{M}$. A method of writing is referred to as a *writing protocol*. We note that in the literature, writing protocols are also referred to as "learning rules". In this work, we use the term "writing" instead of "learning" to avoid confusion with another supervised learning problem that will be introduced in a later part of the paper.

Several writing protocols for the Hopfield network are proposed in [10,17] and [8], which are often motivated by practical considerations and the notion of writing capacity. Roughly speaking, the writing capacity is the maximum number of messages that can be written to the memory so that reliable retrieval is possible. Depending on the choice of the message, the amount of noise allowed in the probe, and the precise definition of reliable retrieval, several expressions of writing capacity have been derived, as briefly discussed in the next section. The main drawback of exist writing protocols is either lack of theoretical explanation or

[1] More generally, function f_W may take the form $f_W(\mathbf{x}) = \text{sign}(W\mathbf{x} + \mathbf{b})$, where $\mathbf{b}$ is an off-set or threshold vector. In the context of this paper, dropping this offset term $\mathbf{b}$ is without loss of generality.

limitation of capacity. To overcome this drawback, we start with interpretability and propose a new writing protocol which has a bigger capacity. We will describe it in detail on Sect. 3.

2 Existing Writing Protocols

In this section we give a brief review on some existing writing protocols for writing a set of messages $\mathcal{M} = \{\mathbf{x}^1, \ldots, \mathbf{x}^M\}$ into a Hopfield memory of size N. Hopfield's original writing protocol [9], referred to as **HOP** hereafter, accounts for each message by adding its outer-product to the weight matrix, i.e., the weight matrix is defined as

$$W = \sum_{m=1}^{M} \left[\mathbf{x}^m \cdot (\mathbf{x}^m)^T - I_N\right], \tag{3}$$

where I_N is the $N \times N$ identity matrix.

With the energy of a state vector $\mathbf{x} \in \{\pm 1\}^N$ defined as

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T W \mathbf{x}, \tag{4}$$

it follows that each message $\mathbf{x}^m \in \mathcal{M}$ is a local minimum of the energy function. (To simplify notation, the dependency of the energy on W is implicit.) Furthermore, the iterative update under Eq. (1) seeks such local minima, i.e.,

$$E(\mathbf{x}) \geq E(f_W(\mathbf{x}))$$

for any $\mathbf{x} \in \mathbb{R}^N$. That is, each message $\mathbf{x^m} \in \mathcal{M}$ is a stable fixed point in the retrieving dynamics.

The **HOP** protocol enjoys two practical properties. Namely, it is local, i.e., W_{ij} depends only on the information available to the i and j-th neurons, and it is incremental, i.e., the writing of a new message depends only on the old W and the new message. The writing capacity of **HOP** protocol for random messages is equal to $\frac{N}{2 \ln N}$ for error free recovery (except for a vanishing fraction of messages) [15] and is approximately $0.14N$ when a vanishing fraction of errors is tolerated [9]. (See [15] for exact definitions and more notions of capacity.)

The pseudo-inverse (**PSE**) protocol is proposed in [10] and is shown to attain a writing capacity of N. In this protocol the (i, j)-th entry of the weight matrix is defined as

$$W_{ij} := \frac{1}{N} \sum_{m=1}^{M} \sum_{m'=1}^{M} x_i^m (Q^{-1})_{mm'} x_j^{m'}, \tag{5}$$

where $Q_{mm'} = \frac{1}{N} \sum_{k=1}^{N} x_k^m x_k^{m'}$ and $(Q^{-1})_{mm'}$ is the (m, m')-th entry of the matrix Q^{-1}.

While the **PSE** achieves a larger writing capacity compared to **HOP**, it loses some of the practically appealing aspects of the **HOP** protocol. A local

and incremental protocol with writing capacity $\frac{N}{\sqrt{2 \ln N}}$ is constructed in [17]. We refer to this protocol as **STO**. In this protocol, the (i, j)-th element of matrix W is computed recursively by

$$W_{ij}^m = W_{ij}^{m-1} + \frac{1}{N} x_i^m x_j^m - \frac{1}{N} x_i^m h_{ji}^m - \frac{1}{N} h_{ij}^m x_j^m, \tag{6}$$

where $W_{ij}^0 = 0$, $h_{ij}^m = \sum_{k=1, k \neq i,j}^N W_{ik}^{m-1} x_k^m$, and $m = 1, 2, \ldots, M$. The resulting matrix W^M is then taken as W.

A more recent protocol is the minimum probability flow (**MPF**) protocol [8].

$$K(W) = \sum_{m=1}^{M} \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x}^m)} \exp\left(\frac{E(\mathbf{x}^m) - E(\mathbf{x}')}{2} \right), \tag{7}$$

where $\mathcal{N}(\mathbf{x})$ is the set of state vectors within Hamming distance one from $\mathbf{x}$ and E is the energy of a state as in (4). (Recall that E depends on W.) In effect, the protocol works by designating a W that not only assigns the messages to local minima of the energy function E, but also shapes the region around such local minima so that no state within Hamming distance one from a message attains the same energy level as the message. (Assuming the distance between any two messages in $\mathcal{M}$ is larger than two.)

Our approach to writing on Hopfield networks resembles that of the **MPF**, where in essence we locate the messages as local minima of the energy function and shape the region around such minima. We rely on a simple connection to recurrent neural networks, which casts the writing problem as a supervised learning problem. Contrast to **MPF**, the Hamming radius within which the energy function is shaped is not limited to one, and can be viewed a tunable parameter via the choice of the training set, which will be described in detail in experiments section.

3 TRN Writing Protocol

We regard the writing to a Hopfield network as that of training a recurrent neural network. A *recurrent neural network* (RNN) of length L is a dynamic system that takes a sequence $(\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^L)$ as an input and generates a sequence $(\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^L)$ as an output according to

$$\mathbf{s}^t = f(\mathbf{s}^{t-1}, \mathbf{u}^t) \tag{8}$$

$$\mathbf{y}^t = g(\mathbf{s}^t, \mathbf{u}^t), \tag{9}$$

where $\mathbf{s}^t$ is referred to as the *state* of the network, the first equation as the *state-update equation*, and the second as the *output equation*. Graphically, the structure of a recurrent unit is shown in Fig. 1.

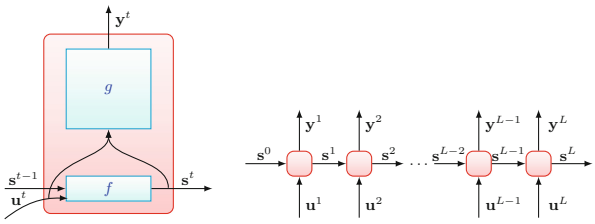


Fig. 1. Generic recurrent neural network of length L. Left: one time step. Right: L time steps.

One of the earliest RNNs [6], often referred to as the Vanilla RNN, specializes the state-update equation to

$$\mathbf{s}^t = \tanh(W\mathbf{s}^{t-1} + U\mathbf{u}^t + \mathbf{b}), \tag{10}$$

where the tanh function is applied element-wise, W and U are matrices of appropriate sizes, and $\mathbf{b}$ is a vector of appropriate size. The output function of Vanilla RNN may vary depending on applications.

Note that in the degenerate case when $\mathbf{u}^t = \mathbf{0}$, for all t, and $\mathbf{b} = \mathbf{0}$, Eq. (10) reduces to (1) with the exception that the sign function in (1) is replaced with the tanh function in (10). As such, a sensible construction of a training set from the set of messages so that, after training, the messages are local minima of a proper choice of a loss function may be viewed as a writing protocol as discussed below.

Given a set of message $\mathcal{M} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^M\} \subseteq \{\pm 1\}^N$ as in the introduction, a training set can be constructed as $\{(\mathbf{x}^1, \mathbf{p}^{1,k}), \ldots, (\mathbf{x}^M, \mathbf{p}^{M,k}) \mid k = 1, \ldots, K\}$, i.e., for each message we generate K probes, which can be understood as some corrupted versions of the message. (Different noise levels are tried out in the experiments section.) Since the objective is to retrieve a target message $\mathbf{x}^m$ from its probe $\mathbf{p}^{m,k}$, we define the loss function as following,

$$D(W) = \sum_{m=1} \sum_{k=1} \|\mathbf{x}^m - f_W^L(W\mathbf{p}^{m,k})\|^2 \tag{11}$$

where f_W^L is the tanh function applied L-times to $W\mathbf{p}^{m,k}$, $\|\cdot\|$ is the euclidean norm. The objective is to minimize the loss function over W, to which end we use the stochastic gradient descent (SGD) [2] to train the RNN. We take the resulting W as the Hopfield weight matrix and refer to the above procedure of obtaining W as the **TRN** writing protocol. The algorithm is summarized in Algorithm 1.

4 Theoretical Results

Two performance metrics, the Message Error Rate (MER) and the Bit Error Rate (BER) are used to evaluate the performance of each protocol. Namely, we define

Algorithm 1. TRN Protocol

Input: A set of messages $\mathcal{M} = \{\mathbf{x}^1, \ldots, \mathbf{x}^M\}$
Output: Weight matrix W

Construct a training set with K probes for each message
Initialize W(Gaussian distribution with 0 mean and 1 variance), $r = 1$, $A = M * K$ is training set size, and B is minibach size
while $r < 1000$ (maximum number of iterations) **do**
  Set$S = 0$, Random sample a minibach $\{(\mathbf{x}^B, \mathbf{p}^B)\}$
  for $i = 1, ..., B$ **do**
    Computer the gradient of loss on weight W
    $S \leftarrow S + \nabla_W D(W, \{(\mathbf{x}^i, \mathbf{p}^i)\})$
  end for
  $W \leftarrow W - \epsilon \cdot S$ (ϵ is the learning rate)
  for $i = 1, ..., A$ **do**
    Calculate the loss of every training sample i
    $d_i = ||\mathbf{x}^i - f_W^L(\mathbf{p}^i))||^2$
  end for
  Calculate the loss and difference: $D(r) = \sum_{i=1}^{A} d_i$, $\Delta D \leftarrow |D(r) - D(r - 100)|$
  if $\Delta D < \delta$ (δ is convergence threshold.) **then**
    exit while loop
  end if
  $r \leftarrow r + 1$
end while
return W

$$\text{MER} := \frac{1}{|\widetilde{\mathcal{P}}_{\text{test}}|} \sum_{(\mathbf{x}^m, \mathbf{p}^{m,k}) \in \widetilde{\mathcal{P}}_{\text{test}}} \mathbf{1}(\mathbf{x}^m = f_W^{\infty}(\mathbf{p}^{m,k})) \tag{12}$$

where $f_W^{\infty}(\cdot)$ is the retrived message, and $\mathbf{1}(\cdot)$ is the indicator function taking value 1 if its argument is true, and 0 otherwise. Similarly,

$$\text{BER} := \frac{1}{N|\widetilde{\mathcal{P}}_{\text{test}}|} \sum_{(\mathbf{x}^m, \mathbf{p}^{m,k}) \in \widetilde{\mathcal{P}}_{\text{test}}} \mathbf{N}(\mathbf{x}^m, f_W^{\infty}(\mathbf{p}^{m,k})) \tag{13}$$

where $\mathbf{N}(\cdot, \cdot)$ counts the number of locations at which the correct and the retrived message differ. In this section, we induce that the bound of metrics is related with the memory messages and the noise level of training set in our specific case, which supports our intuition of training the RNN to get the weight of Hopfield network.

Theorem 1. *For a given memory message set $\mathcal{M}$ which is drawn from the Bernoulli (α) sequence of length N, if the weight setting of Hopfield networks is the **TRN** method under training set with noise level β, then the bound of maximum bit error rate (BER) and message error rate (MER) are*

$$\begin{aligned} BER_i &= \phi(\mu_{h_i^1}, \sigma_{h_i}; 0) + [1 - \phi(\mu_{h_i^{-1}}, \sigma_{h_i}; 0)], \\ BER_{max} &= \max_i BER_i, \\ MER &= 1 - (1 - BER_{max})^N. \end{aligned} \tag{14}$$

$\phi(\mu_{h_i^1}, \sigma_{h_i}; 0)$ represent the integral larger than 0 with the Gaussian distribution with parameter mean value $\mu_{h_i^1}$ and variance σ_{h_i} larger than 0, where $\mu_{h_i^1}, \mu_{h_i^{-1}}, \sigma_{h_i}$ are:

$$\mu_{h_i^1} = W_{ii}(1 - 2\beta) + \sum_{k \neq i}^{N} W_{ik}(1 - 2\alpha)(2\beta - 1), \tag{15}$$

$$\mu_{h_i^{-1}} = W_{ii}(2\beta - 1) + \sum_{k \neq i}^{N} W_{ik}(1 - 2\alpha)(2\beta - 1), \tag{16}$$

$$\sigma_{h_i}^2 = \sum_{k \neq i}^{N} W_{ik}^2[\beta^2 + (1 - \beta)^2]4\alpha(1 - \alpha). \tag{17}$$

Proof. Without loss of generality, element i's state for a specific memory message vector X^m is $x_i^m = 1$. Corresponding noise message is Y^m. The i's output can be expressed as $h_i = W_{ii}y_i^m + \sum_{k \neq i}^{N} W_{ik}y_k^m$ approximately, since most noise messages retrieve to the attractor just by one step's evolution, according to the explanation of in Sect. 3.

$$\mathbb{E}(h_i | x_i^m = 1) = W_{ii}(1 - 2\beta) + \sum_{k \neq i} W_{ik}(1 - 2\alpha)(2\beta - 1). \tag{18}$$

Since $R_k = W_{ik}[\beta \bar{X}_k + (1 - \beta)X_k]$, R_k is an independent random variable, each with finite expected value μ_k and variance σ_k^2.

$$\mu_k = W_{ik}(1 - 2\alpha)(2\beta - 1), \tag{19}$$

$$\sigma_k^2 = W_{ik}^2[\beta^2 + (1 - \beta)^2]4\alpha(1 - \alpha). \tag{20}$$

According to the Lyapunov Central Limit Theorem, if $N \to \infty$, then $\sum_{k \neq i}^{N} R_k$ is a Gaussian distribution with

$$\mu = \sum_{k \neq i}^{N} W_{ik}(1 - 2\alpha)(2\beta - 1), \tag{21}$$

$$\sigma^2 = \sum_{k \neq i}^{N} W_{ik}^2[\beta^2 + (1 - \beta)^2]4\alpha(1 - \alpha). \tag{22}$$

The above formula shows h_i is a Gaussian distribution as $x_i^m = 1$ and $N \to \infty$

$$\mu_{h_i^1} = W_{ii}(1-2\beta) + \sum_{k \neq i}^{N} W_{ik}(1-2\alpha)(2\beta-1), \tag{23}$$

$$\sigma_{h_i^1}^2 = \sum_{k \neq i}^{N} W_{ik}^2[\beta^2 + (1-\beta)^2]4\alpha(1-\alpha). \tag{24}$$

When $x_i^m = -1$, and $N \to \infty$, the prove is all the same with $X_i = 1$. Since the $\sigma_{h_i^1}$ is the same with $\sigma_{h_i^{-1}}$, then it can be abbreviated as σ_{h_i}. So the bit error rate (BER) and the message error rate (MER) are as the Eq. (14).

According to this theorem, if the memory messages $\mathcal{M}$ and the noise level of the training set β are given, we can get the bound of metrics of the **TRN** protocol. The meaning is that the metrics MER and BER on the **TRN** protocol can bound by this theorem. In other words, our **TRN** protocol to writing the weight of Hopfield is feasible and controllable.

5 Experiments

We compare **TRN** with existing writing protocols by performing a number of experiments. Each experiment involves three sets of data: a message set $\mathcal{M} = \{\mathbf{x}^1, \ldots, \mathbf{x}^M\}$, a training set $\widetilde{\mathcal{P}}_{\text{train}}$, and a testing set $\widetilde{\mathcal{P}}_{\text{test}}$. The training set

$$\widetilde{\mathcal{P}}_{\text{train}} = \{(\mathbf{x}^1, \mathbf{p}^{1,k}), \ldots, (\mathbf{x}^M, \mathbf{p}^{M,k}) \mid k = 1, \ldots, K\}$$

is constructed by generating K probes for each message, where each probe is generated by flipping every bit of probe with probability β. The fraction β of corrupted probability in the message is referred to as the noise level. The testing set is generated similarly where we use γ to refer to the noise level to avoid confusion.

In the following experiments, we concentrate on the noise recovery ability and the capacity of under various protocols. In all our experiments, each message in the memory message set $\mathcal{M}$ is generated by drawing a i.i.d. Bernoulli($\frac{1}{2}$) sequence of length N.

5.1 Noise Recovery Study

In this experiment, the number of neuron N is chosen as 100, the learning rate is set to 0.001, and the number of memory messages M takes values in $\{10, 15, 20\}$. These choices of M are based on the understanding that the capacity of Hopfield network under **HOP** is about $0.14N$ [9]. So the three choices correspond respectively to "below capacity", "around capacity", and "above capacity". For each message in $\mathcal{M}$, 10000 noisy probes are generated. In this process, we heuristically allocate the number of probes for each noise level $\gamma = 0.1$, 0.2, and 0.3

Table 1. The comparison among different protocols on metrics MER and BER under the testing set with different noise levels γ and different number of memory messages M. $N = 100$.

MER							BER						
γ	**M**	**HOP**	**PSE**	**STO**	**MPF**	**TRN**	γ	**M**	**HOP**	**PSE**	**STO**	**MPF**	**TRN**
0.1	10	0.017	0	0	0	**0**	0.1	10	0.001	0	0	0	**0**
	15	0.562	0	0	0	**0**		15	0.049	0	0	0	**0**
	20	0.893	0	0.009	0	**0**		20	0.158	0	0	0	**0**
0.2	10	0.089	0.001	0.002	0.005	**0.001**	0.2	10	0.01	0	0	0.001	**0**
	15	0.705	0.01	0.017	0.008	**0.002**		15	0.102	0.001	0.002	0.002	**0**
	20	0.952	0.058	0.11	0.021	**0.008**		20	0.23	0.007	0.012	0.004	**0.002**
0.3	10	0.353	0.103	0.111	0.143	**0.063**	0.3	10	0.079	0.025	0.027	0.043	**0.021**
	15	0.883	0.279	0.3	0.207	**0.138**		15	0.212	0.059	0.065	0.06	**0.041**
	20	0.989	0.551	0.57	0.34	**0.253**		20	0.312	0.11	0.114	0.093	**0.069**

to construct our training set and testing set. Our method **TRN** adopts 3-layer RNN to get the weight of Hopfield network.

The testing results are tabulated in Table 1. Because we just keep three figures after the decimal point, some little BER is approximate to 0. We can find the following observations.

At low noise level $\gamma = 0.1$, all protocols perform well except **HOP**. The fact that **HOP** demonstrates poor noise resilience can be explained by its simple construction of weight matrix, which only takes into account the memory message set. On the other hand, the **HOP** loses efficiency even on 15 memory messages since the MER = 0.562. It is obviously verified by experiments that **HOP** cannot handle more memory messages than $0.14N$.

At medium noise level $\gamma = 0.2$, most protocols perform well except **HOP**. There is a fact that the MER and BER are increased with the number of memory messages. It is obviously judged that there is a capacity among all the protocols. The protocol **HOP** can handle 10 memory messages, but get failed on 15 and 20 memory messages. Except that **HOP**, all other protocols can get better results among all memory messages.

At high noise level $\gamma = 0.3$, there are some errors among all protocols on all memory messages. Below the capacity of Hopfield network, all protocols except **HOP** can handle the memory task. Around the capacity of Hopfield network and above capacity of Hopfield network, there are more errors on other protocols. We can also find that along with the increase of memory messages for all the protocols, the memory ability is getting weaker.

In a word, as the noise contained in the probe increases, all protocols degrade their performance, among which the degradation of **TRN** is the slowest. On 10 messages (below capacity), most protocols perform well except **HOP**. The fact that **HOP** demonstrates poor noise resilience can be explained by its simple construction of weight matrix, which only takes into account the message set. All other messages demonstrate some level of noise resilience, among which **TRN**

performs the best. Even operating above capacity, namely on 20 messages, **TRN** demonstrates an acceptable error rate.

5.2 Capacity Study

According to the outstanding results of our protocol **TRN** on the noise recovery experiments, we do more experiments on different lengths of Hopfield network: Short-Message experiments, Medium-Message experiments and Long-Message experiment, to study the capacity of our protocol **TRN** with other exiting protocols.

In the following experiments, the training $\widetilde{\mathcal{P}}_{\text{train}}$ is a mixture of probes generated using noise level β. Because the number of patterns increases with the noise level, we adopt the following equation to get the different ratio of every noise level to alleviate sampling difficulty.

$$N_\beta = \frac{1}{2} + 0.01 \cdot (1+\beta)^{12} \tag{25}$$

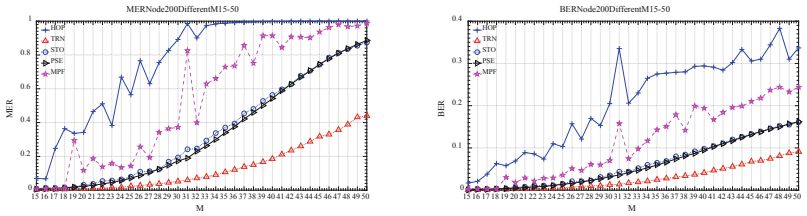
$$P_{N_\beta} = \frac{N_\beta}{\sum_{\alpha \in \{\beta\}} N_\alpha} \tag{26}$$

On Short-Message experiments, the training set $\widetilde{\mathcal{P}}_{\text{train}}$ is generated by 20000 noisy probe for each message in $\mathcal{M}$, and the testing set $\widetilde{\mathcal{P}}_{\text{test}}$ is generated by 10000 noisy probe for each message. On Medium-Message and Long-Message experiments, both the training set $\widetilde{\mathcal{P}}_{\text{train}}$ and the testing set $\widetilde{\mathcal{P}}_{\text{test}}$ are generated by 10000 noisy probe for each message due to limited computing power. In this process, we heuristically allocate the number of probes for each noise level so that the number of probes grows with the noise level according to the above equation. The maximize training noise level β and testing noise level γ are 0.3 on Short-Message experiments, and for the other experiments, we allocate the maximize training noise level β and testing noise level γ as 0.1.

Short-Message Experiments. In these short-message experiments, the message length N is chosen as 200, and the number of messages M takes values between 15 and 50 with gap 1. Again noting that these choices of M covers $0.14N$. The training $\widetilde{\mathcal{P}}_{\text{train}}$ is generated using the maximize noise level $\beta = 0.3$. The training $\widetilde{\mathcal{P}}_{\text{train}}$ is a mixture of probes generated using noise level β from 0.05 to 0.3 (with spacing 0.05). Then take the above equation to calculate the exactly number of training set. For each message in $\mathcal{M}$, 20000 noisy probes are generated. For the testing $\widetilde{\mathcal{P}}_{\text{test}}$, the only difference is that we generate 10000 testing probes for each message in $\mathcal{M}$. The size of training batch is 2000 and the learning rate is 0.001. Our method **TRN** adopts 3-layer RNN to get the weight of Hopfield network.

The results of these experiments are shown in Fig. 2. First of all, as the number of the memory messages increase, all protocols degrade their performance on metric MER and BER, among which the degradation of **TRN** is the slowest.

Besides, the effective of protocol **STO** and **PSE** is almost the same. The protocol **HOP** and **MPF** have some vibration with the number of memory. So our protocol has the perfect efficiency and stability. With 40 memory messages in 200-neuron Hopfield network, the MER is below 0.2, this is rather low compared with the other previous protocols. Moreover, the tendency of BER is the same with MER from the right figure in Fig. 2. This is in accord with our intuition.



(a) MER (b) BER

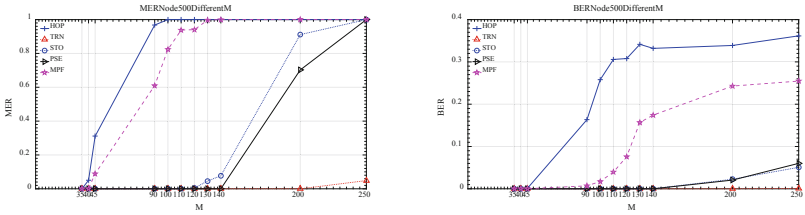
Fig. 2. The comparison among different protocols on metrics MER and BER under the testing sets with different number of memory messages M in $\{15, 16, \ldots, 50\}$ $N = 200$.

Medium-Message Experiments. In these medium-message experiments, the message length N is chosen as 500, and the number of messages M takes on different values. The training $\widetilde{\mathcal{P}}_{\text{train}}$ is a mixture of probes generated using noise level β from 0.02 to 0.1 (with spacing 0.02). Then take the above equation to calculate the exactly number of training set. For each message in $\mathcal{M}$, 10000 noisy probes are generated. Testing probes with noise level $\gamma = 0.1$. For each message, 10000 testing probes are used.The size of training batch is 2000 and the learning rate is 0.01. Our method **TRN** adopts 2-layer RNN to get the weight of Hopfield network.

The results of these experiments are shown in Fig. 3. Similarly, as the number of the memory messages increases, all protocols degrade their performance on metric MER and BER, among which the degradation of **TRN** is the slowest. With 40 messages (around the McLiece's capacity), most protocols perform well except **HOP**, this result verifies that the correctness of capacity description in McLiece paper. The fact that **STO** doesn't work on the size of 120 memory messages, and **PSE** is invalid on the size of 140 memory messages. When the size of memory messages increase to 150, all protocols collapse except our protocol **TRN**. Even on the size 200 of the memory messages, our protocol **TRN** also performs well. But our protocol gets faild on the size 250 with a low MER 0.0486.

Long-Message Experiments. In the long-message experiments, the parameter settings are the same with medium-message experiments.

All the results are similar with the Medium-Message experiments. Basically, the protocol **HOP** and **MPF** can just handle the low number of memory messages. The other three protocols **STO**, **PSE** and **TRN** can have a good efficient performance on metric MER and BER at a low number of memory messages.



(a) MER (b) BER

Fig. 3. The comparison among different protocols on metrics MER and BER under the testing sets with different number of memory messages M in $\{35, 40, 45, 90, 100, 110, 120, 130, 140, 150, 200\}$ $N = 500$.

Table 2. The comparison among different protocols on metrics MER and BER under the testing sets with different number of memory messages M. $N = 1000$.

M	Metric	**HOP**	**PSE**	**STO**	**MPF**	**TRN**
45	MER	0	0	0	0	**0**
	BER	0	0	0	0	**0**
70	MER	0.0714	0	0	0	**0**
	BER	0.0001	0	0	0	**0**
80	MER	0.1625	0	0	0	**0**
	BER	0.0002	0	0	0	**0**
190	MER	1	0	0	0.9725	**0**
	BER	0.3137	0	0	0.0159	**0**
200	MER	1	0	0.0015	0.9899	**0**
	BER	0.3332	0	0	0.0358	**0**
300	MER	1	0	0.2848	1	**0**
	BER	0.3511	0	0.0007	0.2563	**0**
350	MER	1	0.0042	0.6891	1	**0**
	BER	0.3424	0.0001	0.0061	0.3716	**0**
450	MER	1	1	0.9981	1	**0.0003**
	BER	0.3593	0.0442	0.0377	0.2761	**0**

The fact that **STO** doesn't work on 200 memory messages, and **PSE** gets failed on 350 memory messages. When the size of memory messages increase to 350, all protocols collapse except our protocol **TRN** (Table 2).

6 Conclusion

Our proposed protocol **TRN** possesses two advantages compared to the existing writing protocols for Hopfield network. On one hand, our writing protocol has a better ability of noise recovery compared to the other existing writing protocols,

since our **TRN** training protocol is based on training set considering the noise recovery problem. On the other hand, the capacity of our writing protocol is bigger than other existing writing protocols from all experiments.

It should be noted that the superior performance of **TRN** does not come free of cost. The complexity of this protocol is much higher than the other protocols, particularly at long message length N. This is the number of potential noisy problems grows exponentially with N. At large N, generating the training set by sampling from the noisy probes with a decent sampling rate would result in a large training set, increasing the complexity significantly. In our experiments with $N = 1000$, it appears that the time required in **TRN** is about 5 times more than **MPF** and nearly 100 times more than **PSE**, **STO** and **HOP**.

This however does not close the possibility of using the **TRN** approach as a practical writing protocol. We believe that if one is to cleverly introduce additional structure in the connection matrix W of **TRN**, the required training examples can be significantly reduced. Moreover, the experiments are conducted on the artificial data, so we can do more experiments on some real data. Last but not least, the size of capacity of our writing protocol **TRN** should be studied in theory. We are planning to further investigate along these directions.

References

1. Anderson, J.R., Bower, G.H.: Human Associative Memory. Psychology Press, London (2014)
2. Bottou, L.: Largescale machine learning with stochastic gradient descent. In: Lechevallier, Y., Saporta, G. (eds.) Proceedings of COMPSTAT 2010, pp. 177–186. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-7908-2604-3_16
3. Cheng, K.S., Lin, J.S., Mao, C.W.: The application of competitive hopfield neural network to medical image segmentation. IEEE Trans. Med. Imaging **15**(4), 560–567 (1996)
4. Ding, J., Sun, Y.Z., Tan, P., Ning, Y.: Detecting communities in networks using competitive hopfield neural network. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE (2018)
5. Duan, S., Dong, Z., Hu, X., Wang, L., Li, H.: Small-world hopfield neural networks with weight salience priority and memristor synapses for digit recognition. Neural Comput. Appl. **27**(4), 837–844 (2016)
6. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**, 179–211 (1990)
7. Folli, V., Leonetti, M., Ruocco, G.: On the maximum storage capacity of the hopfield model. Front. Comput. Neurosci. **10**, 144 (2017)
8. Hillar, C., SohlDickstein, J., Koepsell, K.: Efficient and optimal binary hopfield associative memory storage using minimum probability flow. arXiv preprint arXiv:1204.2916 (2012)
9. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. **79**(8), 2554–2558 (1982)
10. Kanter, I., Sompolinsky, H.: Associative recall of memory without errors. Phys. Rev. A **35**(1), 380 (1987)
11. Kobayashi, M.: Multistate vector product hopfield neural networks. Neurocomputing **272**, 425–431 (2018)

12. Krotov, D., Hopfield, J.J.: Dense associative memory for pattern recognition. In: Advances in Neural Information Processing Systems, pp. 1172–1180 (2016)
13. Lee, K.Y., Sode-Yome, A., Park, J.H.: Adaptive hopfield neural networks for economic load dispatch. IEEE Trans. Power Syst. **13**(2), 519–526 (1998)
14. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**(4), 115–133 (1943)
15. McEliece, R., Posner, E., Rodemich, E., Venkatesh, S.: The capacity of the hopfield associative memory. IEEE Trans. Inform. Theory **33**(4), 461–482 (1987)
16. Rebentrost, P., Bromley, T.R., Weedbrook, C., Lloyd, S.: Quantum hopfield neural network. Phys. Rev. A **98**(4), 042308 (2018)
17. Storkey, A.: Increasing the capacity of a hopfield network without sacrificing functionality. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 451–456. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0020196
18. Wang, S.: Classification with incomplete survey data: a hopfield neural network approach. Comput. Oper. Res. **32**(10), 2583–2594 (2005)
19. Zhen, H., Wang, S.N., Zhou, H.J.: Unsupervised prototype learning in an associative-memory network. arXiv preprint arXiv:1704.02848 (2017)