Meta predictive learning model of languages in neural circuits

Chan Li,[1,2,*] Junbin Qiu,[1,*] and Haiping Huang[1,3,†]
[1]*PMI Laboratory, School of Physics, Sun Yat-sen University, Guangzhou 510275, People's Republic of China*
[2]*Department of Physics, University of California, San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA*
[3]*Guangdong Provincial Key Laboratory of Magnetoelectric Physics and Devices, Sun Yat-sen University, Guangzhou 510275, People's Republic of China*

(Received 6 November 2023; accepted 18 March 2024; published 12 April 2024)

Large language models based on self-attention mechanisms have achieved astonishing performances, not only in natural language itself, but also in a variety of tasks of different nature. However, regarding processing language, our human brain may not operate using the same principle. Then, a debate is established on the connection between brain computation and artificial self-supervision adopted in large language models. One of most influential hypotheses in brain computation is the predictive coding framework, which proposes to minimize the prediction error by local learning. However, the role of predictive coding and the associated credit assignment in language processing remains unknown. Here, we propose a mean-field learning model within the predictive coding framework, assuming that the synaptic weight of each connection follows a spike and slab distribution, and only the distribution, rather than specific weights, is trained. This meta predictive learning is successfully validated on classifying handwritten digits where pixels are input to the network in sequence, and moreover, on the toy and real language corpus. Our model reveals that most of the connections become deterministic after learning, while the output connections have a higher level of variability. The performance of the resulting network ensemble changes continuously with data load, further improving with more training data, in analogy with the emergent behavior of large language models. Therefore, our model provides a starting point to investigate the connection among brain computation, next-token prediction, and general intelligence.

DOI: 10.1103/PhysRevE.109.044309

I. INTRODUCTION

Large language models (LLMs) based on transformer structures greatly boost both industrial and academic interests in artificial general intelligence [1]. LLMs are able to achieve state-of-the-art performances in a variety of different tasks, only trained by next-token prediction. The transformer structure computes self-attention scores to capture statistical correlations among input tokens in parallel, which is in stark contrast to brainlike recurrent computation based on synaptic feedback in temporal depth (e.g., a short working memory). In addition, LLMs typically require a sizable number of corpus to trigger emergence of intelligence, compared to the fact that much less data is needed for a child to acquire linguistic ability. Therefore, it is necessary to establish a mechanistic model of language processing to understand the biological plausible mechanism and underlying physics law governing phase transitions through statistical patterns of model hyperparameters [2].

In brain science, predictive coding is one of the most influential hypotheses that can implement hierarchical information processing [3,4]. The predictive coding derives the neuroplasticity rule based on local error signal [5], whose goal is to minimize the surprise between the prediction and belief of a generative model of the outside world [6]. The framework of predictive coding has several benefits for theoretical research. First, this framework can be understood as a joint optimization of neural dynamics and synaptic connections in order to maximize the observation probability of sensory inputs to the neural circuits [7]. Second, this principle shares exactly the same spirit adopted in variational free energy frameworks [6]. Recently, there appeared intense interest in studying the biological implementation of this hypothesis [8–10], in developing algorithmic applications [11–13], and in studying the trade-off between energy minimization and information robustness in a linear model of lateral predictive coding [14].

Predictive coding postulates that the cortex carries out a predictive model where the incoming sensory signals are predicted by using prediction-error-driven learning and inference. In this sense, predictive coding is a nice framework to model language processing. However, weight uncertainty is commonly observed in neural circuits [15,16], e.g., synaptic transmission is stochastic, and spine size is subject to fluctuation. But these effects were not taken into account in previous studies of predictive coding, as remarked in a recent review [17]. In addition, the weight uncertainty was recently studied in recurrent neural networks [18], inspiring fluctuation-driven synaptic plasticity. Therefore, exploring how the weight uncertainty affects predictive coding in language processing will help to establish a mechanistic model of language processing to understand the biological plausible mechanism and underlying physics law governing phase transitions through associated statistical patterns of model hyperparameters. In this work we derive a mean-field learning rule for predictive coding in recurrent neural networks (RNNs), which is a fundamental structure for natural language processing [19–25], and

[*]These authors contributed equally to this work.
[†]huanghp7@mail.sysu.edu.cn

we assume that each direction of connection follows a weight distribution incorporating weight sparsity and variance. We thus call this rule meta predictive learning (MPL). This framework is tested first on the classification of the MNIST dataset [26], where pixels in an image are divided into groups, then a toy language dataset, where we can have a thorough exploration of algorithmic capabilities, and finally, a language corpus in the real world (Penn Treebank corpus [27]).

Our proposed MPL achieves equal or even better performance compared with traditional methods in all three tasks, showing the advantage of *ensemble* predictive coding, since examples of single networks can be readily sampled from the trained distribution [18,28]. By analyzing the distribution of hyperparameters we are able to find that most connections are deterministic in the input and recurrent layers, while the output layer has a higher level of variability. The observation that the output connections bear a higher level of variability is a universal result in all three tasks, which may particularly connect to the generative function of the language processing model. The network performance changes nonlinearly and continuously with data load $\alpha = \frac{M}{N}$, where M is the training data size and N is the number of neurons in the circuit, and we found that the critical point is given by $\alpha_c \approx 0.02$, beyond which a chance level of prediction is absent. With increasing the size of training data, the performance further improves until a perfect learning is achieved. We can then test the resulting network to generate text of arbitrary length (to create something is the first step to understanding that thing), and the generated text follows perfectly the grammatical rule set before training. In addition, our MPL is able to accomplish comparable performances in the Penn Treebank corpus with other training methods in RNN, although the framework is less accurate than the transformer structure, which thereby calls for further studies about the mechanistic difference between biological learning and nonbiological transformer learning, and how the latter can inspire discovery of new fundamental elements of computation that can realize logical and mathematical reasoning in many different tasks [29,30].

II. METHOD

Here we consider meta predictive learning in a vanilla RNN which processes a time-dependent sequence $\mathbf{x}$ with time length T. We adapt the predictive learning framework (see a recent review [6]) to the following vision and language processing (main focus of this paper). In this framework a belief neural state is introduced and updated to minimize the deviation from the actual neural state (an energy function), which is called the inference phase; when the currently best belief is achieved, the learning phase starts, i.e., the synaptic weights connecting neurons are jointly updated to minimize the same energy function further for the goal of achieving a minimal surprise between the belief and prediction (or actual and target outputs, see details below). This inference-learning loop can be repeated many times until a satisfied accuracy is achieved. When inference-learning loops are completed, a prediction phase starts, during which the generation or classification ability is tested using the belief state and the distribution parameters of weights. Our idea is thus to focus on the weight distribution for the learning phase, including the sparsity of weights and the weight variance if the connection is not absent. Because the hyperparameters of the distribution are learned, rather than the specific weight values, we call this learning rule meta predictive learning. Rare studies focus on the important role of weight fluctuations in predictive learning, especially in language processing. In the following we detail this framework.

More precisely, the input signal of $N_{\rm in}$ dimension is first mapped to the recurrent reservoir of N neurons by an input weight matrix $\mathbf{w}^{\rm in} \in \mathbb{R}^{N\times N_{\rm in}}$, whose element $w_{ij}^{\rm in}$ indicates the connection weight value from neuron j in the input to the reservoir neuron i. The neurons in the reservoir interact with each other with reciprocal connections $\mathbf{w} \in \mathbb{R}^{N\times N}$, where elements w_{ij} specify the directional coupling from neuron j to neuron i, and moreover, $w_{ij} \neq w_{ji}$. The self-connectivity w_{ii} is also included and can be learned without imposing any prior knowledge [18]. The internal neural dynamics $r_i(t)$ is read out via the output weight $\mathbf{w}^{\rm out} \in \mathbb{R}^{N_{\rm out}\times N}$. In the predictive learning setting, $\mathbf{r}$ is interpreted as a *belief* state when $\mathbf{x}$ is observed as a sensory input, which can be continuously updated to match the actual prediction whose dynamics reads as follows:

$$\begin{aligned} h_i(t) &= \sum_{j=1}^{N} w_{ij} f[r_j(t-1)] + \sum_{j=1}^{N_{\rm in}} w_{ij}^{\rm in} x_j(t), \\ y_i(t) &= \phi\left(\sum_{j=1}^{N} w_{ij}^{\rm out} f[r_j(t)]\right), \end{aligned} \tag{1}$$

where hereafter t denotes the discrete time steps to run the dynamics, $y_i(t)$ is the ith component of the network output, $f(\cdot)$ denotes the nonlinear activation function, and we use the ReLU function for all tasks. $\phi(\cdot)$ is the output nonlinear function, and we use the softmax function to specify the probability over all classes, which is defined as $\phi[z_k(t)] = \frac{e^{z_k(t)}}{\sum_j e^{z_j(t)}}$. The belief state $\mathbf{r}(t)$ is updated for all time steps up to the sequence length to minimize the prediction error between $\mathbf{r}$ and $\mathbf{h}$, which will be detailed below. Note that $\mathbf{r}(0) = 0$, and we fix the belief of the output node $\mathbf{r}_y = \hat{\mathbf{y}}$, where $\hat{\mathbf{y}}$ denotes the label of input $\mathbf{x}$. Generally speaking, all beliefs can be initialized to random values.

The core idea of the proposed MPL is assuming the distribution of the network parameters is subject to the following spike and slab (SaS) form [18,28]:

$$\begin{aligned} P\left(w_{ij}^{\rm in}\right) &= \pi_{ij}^{\rm in}\delta\left(w_{ij}^{\rm in}\right) + \left(1-\pi_{ij}^{\rm in}\right) \\ &\quad\times \mathcal{N}\left(\frac{m_{ij}^{\rm in}}{N_{\rm in}\left(1-\pi_{ij}^{\rm in}\right)}, \frac{\Xi_{ij}^{\rm in}}{N_{\rm in}\left(1-\pi_{ij}^{\rm in}\right)}\right), \\ P(w_{ij}) &= \pi_{ij}\delta(w_{ij}) + (1-\pi_{ij}) \\ &\quad\times \mathcal{N}\left(\frac{m_{ij}}{N(1-\pi_{ij})}, \frac{\Xi_{ij}}{N(1-\pi_{ij})}\right), \\ P\left(w_{ki}^{\rm out}\right) &= \pi_{ki}^{\rm out}\delta\left(w_{ki}^{\rm out}\right) + \left(1-\pi_{ki}^{\rm out}\right) \\ &\quad\times \mathcal{N}\left(\frac{m_{ki}^{\rm out}}{N\left(1-\pi_{ki}^{\rm out}\right)}, \frac{\Xi_{ki}^{\rm out}}{N\left(1-\pi_{ki}^{\rm out}\right)}\right). \end{aligned} \tag{2}$$

Note that $N(1-\pi)$ specifies the mean degree of each neuron in the reservoir, as $1-\pi$ specifies the synaptic connection probability, which is biologically plausible due to unreliable stochastic noise [31]. The first and second moments of elements w_{ij}^{ℓ} (ℓ is in, out, or recurrent, depending on the context; for the recurrent context, the item has no superscript) can be derived as $\mu_{ij}^{\ell}=\frac{m_{ij}^{\ell}}{N_\ell}$, $\varrho_{ij}^{\ell}=\frac{(m_{ij}^{\ell})^2}{N_\ell^2(1-\pi_{ij}^{\ell})}+\frac{\Xi_{ij}^{\ell}}{N_\ell}$, respectively. Note that the mean and variance of the Gaussian slab are scaled by the number of mean synaptic connections, such that the prediction of neuron is of the order 1.

Considering the statistics of synaptic weights and a large number of afferent projections for each neuron in Eq. (1), which is true in real neural circuits [32], we can reasonably assume the prediction $h_i(t)$ ($\forall i$) follows an evolving Gaussian distribution whose mean and variance are defined by $G_i=G_i^{\rm rec}+G_i^{\rm in}$ and $\Delta_i^1=(\Delta_i^{\rm in})^2+(\Delta_i^{\rm rec})^2$, respectively. This is intuitively the result of a central limit theorem. The statistics of readout neural currents can be derived in a similar way. Therefore, the mean-field dynamics of this model can be written as

$$\begin{aligned}h_i(t+1)&=G_i^{\rm rec}(t)+G_i^{\rm in}(t+1)\\&\quad+\epsilon_i^1(t+1)\sqrt{\left[\Delta_i^{\rm in}(t+1)\right]^2+\left[\Delta_i^{\rm rec}(t)\right]^2},\\ y_k(t)&=\phi\left[G_k^{\rm out}(t)+\epsilon_k^2(t)\Delta_k^{\rm out}(t)\right],\end{aligned}\tag{3}$$

where t denotes the discrete time steps to run the dynamics, and the superscript in ϵ indicates different types of standard Gaussian random variables—one for reservoir neurons (with superscript 1) and the other for readout neurons (with superscript 2). By definition, $\{\epsilon_i^{1,2}(t)\}$ are both time and neuron index dependent. Given μ_{ij}^{ℓ} and ϱ_{ij}^{ℓ}, the mean currents together with the associated fluctuations are derived below:

$$\begin{aligned}G_i^{\rm in}(t+1)&=\sum_j\mu_{ij}^{\rm in}x_j(t+1)\\ G_i^{\rm rec}(t+1)&=\sum_j\mu_{ij}f[r_j(t+1)]\\ G_k^{\rm out}(t+1)&=\sum_j\mu_{kj}^{\rm out}f[r_j(t+1)]\\ \left[\Delta_i^{\rm in}(t+1)\right]^2&=\sum_j\left(\varrho_{ij}^{\rm in}-\left(\mu_{ij}^{\rm in}\right)^2\right)[x_j(t+1)]^2\\ \left[\Delta_i^{\rm rec}(t+1)\right]^2&=\sum_j\left(\varrho_{ij}-\left(\mu_{ij}\right)^2\right)(f[r_j(t+1)])^2\\ \left[\Delta_k^{\rm out}(t+1)\right]^2&=\sum_j\left(\varrho_{kj}^{\rm out}-\left(\mu_{kj}^{\rm out}\right)^2\right)(f[r_j(t+1)])^2.\end{aligned}\tag{4}$$

The prediction dynamics [Eq. (1) or Eq. (3) in the meta learning context] can be interpreted as perceptual inference, widely used in energy-based optimization of brain dynamics [7], while the learning given below is called the neuroplasticity. Both processes minimize exactly the same energy (or variational free energy in general [5]).

Predictive learning can be derived from a temporally hierarchical Gaussian probabilistic principle [4,5], where the objective function is given by the negative log-likelihood of the joint neural-state distribution. To optimize this objective function, we apply a mean-field approximation of the joint distribution and an additional Laplace approximation that leads to Gaussian forms [17]. We give a brief interpretation in Appendix A. In essence, the predictive learning maximizing this log-likelihood aims to minimize the following energy cost [12,33]:

$$\mathcal{F}=\sum_{j=1}^{2}\sum_{t=1}^{T}\mathscr{E}_j(t).\tag{5}$$

This energy function is exactly the variational free energy in the above Gaussian probabilistic principle. The choice of $\mathscr{E}_j(t)$ depends on the problem at hand. If the network produces an output at every time step as in the language model, $\mathscr{E}_1(t)=\frac{1}{2}\|\mathbf{r}(t)-\mathbf{h}(t)\|^2$, and $\mathscr{E}_2(t)=-\sum_i\hat{y}_i(t)\ln[y_i(t)]$. However, if the network only makes the final decision in the last time step as in the classification task, i.e., $y_k(T)=\phi[G_k^{\rm out}(T)+\epsilon_k^2(T)\Delta_k^{\rm out}(T)]$, we then have the energy terms $\mathscr{E}_1(t)=\frac{1}{2}\|\mathbf{r}(t)-\mathbf{h}(t)\|^2$ for $t=1,\dots,T-1$, $\mathscr{E}_1(T)=0$, and $\mathscr{E}_2(t)=0$ for $t<T$, $\mathscr{E}_2(T)=-\sum_i\hat{y}_i\ln[y_i(T)]$. Moreover, we define the prediction error $\mathscr{E}_1'(t)=\mathbf{r}(t)-\mathbf{h}(t)$ and $\mathscr{E}_2'(t)=\mathbf{r_y}(t)-\mathbf{y}(t)$. This error can be propagated along the dendritic connections in neural circuits [34]. In a mathematical sense, the prediction errors can be interpreted as the gradients of the above energy cost.

In essence, the predictive learning consists of three phases: inference phase, learning phase, and prediction phase [see Eq. (1), and in the current meta-learning, Eq. (3) is used]. We next show the predictive learning details for the language processing, while other applications can be readily adapted. First of all, during the inference phase, the belief $\mathbf{r}(t)$ is updated to minimize the energy function $\mathcal{F}$ with the following increment:

$$\begin{aligned}\Delta r_i(t')&=-\gamma\frac{\partial\mathcal{F}}{\partial r_i(t')}\\&=-\gamma\frac{\partial\mathscr{E}_1(t')}{\partial r_i(t')}-\gamma\frac{\partial\sum_{t\neq t'}\mathscr{E}_1(t)}{\partial r_i(t')}-\gamma\frac{\partial\mathscr{E}_2(t')}{\partial r_i(t')}\\&=-\gamma\mathscr{E}_{1,i}'(t')+\gamma\sum_j\mathscr{E}_{1,j}'(t'+1)\frac{\partial h_j(t'+1)}{\partial r_i(t')}\\&\quad+\gamma\sum_j\mathscr{E}_{2,j}'(t')\frac{\partial\left[G_j^{\rm out}(t')+\epsilon_j^2(t')\Delta_j^{\rm out}(t')\right]}{\partial r_i(t')}\\&=-\gamma\mathscr{E}_{1,i}'(t')+\gamma f'[r_i(t')]\sum_j\mathscr{E}_{1,j}'(t'+1)\mu_{ji}\\&\quad+\gamma\sum_j\mathscr{E}_{2,j}'(t')\mu_{ji}^{\rm out}f'[r_i(t')]\\&\quad+\gamma\sum_j\mathscr{E}_{1,j}'(t'+1)\hat{\epsilon}_{ji}^1+\gamma\sum_j\mathscr{E}_{2,j}'(t')\hat{\epsilon}_{ji}^2,\end{aligned}\tag{6}$$

where γ indicates the learning rate for the inference phase (we choose $\gamma=0.1$ for all tasks), $\hat{\epsilon}_{ji}^1=\epsilon_j^1(t'+1)\frac{[\varrho_{ji}-(\mu_{ji})^2]f'[r_i(t')]f[r_i(t')]}{\sqrt{(\Delta_j^{\rm in}(t'+1))^2+(\Delta_j^{\rm rec}(t'))^2}}$, and $\hat{\epsilon}_{ji}^2=\epsilon_j^2(t')\frac{[\varrho_{ji}^{\rm out}-(\mu_{ji}^{\rm out})^2]f'[r_i(t')]f[r_i(t')]}{\Delta_j^{\rm out}(t')}$. It is evident that the last two terms in Eq. (6) are related to the fluctuations caused by the network statistics. The interplay between the network statistics and the prediction errors governs the belief dynamics, which was not considered in previous studies.

We emphasize this intrinsic property of neural dynamics is due to ongoing fluctuations of synaptic weights in the presence of circuit noise [31]. Equation (6) thus addresses how the neural belief is shaped under the fluctuating circuit environment.

The goal of this inference process is to find the best configuration of belief for synaptic weight modifications (aforementioned neuroplasticity). When the decrease of the energy $\mathcal{F}$ becomes stable, e.g., $|\mathcal{F}^t - \mathcal{F}^{t-1}| < 0.1$, or when a maximal number of iterations (n in our algorithm 1) is approached, the learning phase starts, i.e., the hyperparameters $[\mathrm{m}^\ell, \boldsymbol{\pi}^\ell, \boldsymbol{\Xi}^\ell]$ are updated based on the local error signal $\mathscr{E}_j^\prime(t)$ with the following increments:

$$\Delta m_{ij}^\ell = -\eta \frac{\partial \mathcal{F}}{\partial m_{ij}^\ell} = -\eta \sum_t \mathscr{E}^\prime_{\ell^\prime,i}(t) \times \left[-\frac{1}{N_\ell}\xi_j^\ell - \epsilon_i^{\ell^\prime}(t)\frac{m_{ij}^\ell \pi_{ij}^\ell \left(\xi_j^\ell\right)^2}{(N^\ell)^2(1-\pi_{ij}^\ell)\sqrt{\Delta_i^{\ell^\prime}}}\right],$$

$$\Delta \pi_{ij}^\ell = -\eta \frac{\partial \mathcal{F}}{\partial \pi_{ij}^\ell} = -\eta \sum_t \mathscr{E}^\prime_{\ell^\prime,i}(t) \times \left[-\epsilon_i^{\ell^\prime}(t)\frac{\left(m_{ij}^\ell\right)^2 \left(\xi_j^\ell\right)^2}{2(N_\ell)^2\left(1-\pi_{ij}^\ell\right)^2\sqrt{\Delta_i^{\ell^\prime}}}\right],$$

$$\Delta \Xi_{ij}^\ell = -\eta \frac{\partial \mathcal{F}}{\partial \Xi_{ij}^\ell} = -\eta \sum_t \mathscr{E}^\prime_{\ell^\prime,i}(t)\left[-\epsilon_i^{\ell^\prime}(t)\frac{\left(\xi_j^\ell\right)^2}{2N_\ell\sqrt{\Delta_i^{\ell^\prime}}}\right], \tag{7}$$

where η denotes the learning rate for the learning phase, $\Delta_i^1 = [\Delta_i^{\text{in}}(t)]^2 + [\Delta_i^{\text{rec}}(t-1)]^2$, and $\Delta_i^2 = [\Delta_i^{\text{out}}(t)]^2$. To derive Eq. (7), the chain rule and mean-field dynamics [Eq. (3)] are used. The meaning of superscripts depends on the network structure where the computation is carried out. If $\ell = \text{in}$, $\ell^\prime = 1$, $\xi_j^\ell = x_j(t)$, $N_\ell = N_{\text{in}}$; if ℓ indicates the recurrent reservoir, $\ell^\prime = 1$, $\xi_j^\ell = f[r_j(t-1)]$, $N_\ell = N$; if $\ell = \text{out}$, $\ell^\prime = 2$, $\xi_j^\ell = f[r_j(t)]$, $N_\ell = N$. For easy comprehension, we summarize all mathematical items and associated explanations in Appendix D. The dynamics of π and Ξ are purely driven by the synaptic fluctuation, while the m dynamics is contributed by the activity (belief or sensory observation) and the synaptic fluctuation. The m yields impacts on π and Ξ as well. Note that the vanilla predictive coding does not take into account synaptic fluctuations (see also Appendix C), which is indeed ubiquitous in neural circuits [16]. One typical source is that the synaptic noise results from noisy biochemical processes underlying synaptic transmission, while the other source is the fluctuation of spine sizes in the neocortex, and the existence of silent synapses [35,36].

In practice, implementation of the meta learning rule in Eq. (7) immediately follows the inference phase, where the update of belief $\mathbf{r}$ has made $\mathcal{F}$ converge. To improve the prediction performance, the inference and learning phases are repeated a number of times. The prediction phase is carried out after a round of inference-learning loop to test the model's

ALGORITHM 1. Meta predictive learning algorithm.

```
 1: # Inference
 2: Given: input $\mathbf{x}$, label $\hat{\mathbf{y}}$, randomly initialized belief $\mathbf{r}$, $\mathbf{r}_y = \hat{\mathbf{y}}$,
    standard Gaussian variables $\boldsymbol{\epsilon}^1$ and $\boldsymbol{\epsilon}^2$
 3: for iter = 1, ..., $n$ do
 4:     for $t = 1, \ldots, T$ do
 5:         $h_i(t+1) = G_i^{\text{rec}}(t) + G_i^{\text{in}}(t+1) + \epsilon_i^1(t+1)\sqrt{[\Delta_i^{\text{in}}(t+1)]^2 + [\Delta_i^{\text{rec}}(t)]^2}$;
 6:         $y_k(t) = \phi[G_k^{\text{out}}(t) + \epsilon_k^2(t)\Delta_k^{\text{out}}(t)]$;
 7:         $\mathbf{r}(t) = \mathbf{r}(t) + \Delta\mathbf{r}(t)$.
 8:     end for
 9: end for
10: # Learning
11: for $\ell$ = in, out, recurrent do
12:     for $t = 1, \ldots, T$ do
13:         $\mathbf{m}^\ell = \mathbf{m}^\ell + \Delta\mathbf{m}^\ell$;
14:         $\boldsymbol{\pi}^\ell = \boldsymbol{\pi}^\ell + \Delta\boldsymbol{\pi}^\ell$;
15:         $\boldsymbol{\Xi}^\ell = \boldsymbol{\Xi}^\ell + \Delta\boldsymbol{\Xi}^\ell$.
16:     end for
17: end for
18: Output: $\mathbf{r}$.
19: # Prediction
20: Given: test data $\mathbf{x}$, converged belief $\mathbf{r}$, another set of standard
    Gaussian variables $\boldsymbol{\epsilon}^1$ and $\boldsymbol{\epsilon}^2$
21: for $t = 1, \ldots, T$ do
22:     $h_i(t+1) = G_i^{\text{rec}}(t) + G_i^{\text{in}}(t+1) + \epsilon_i^1(t+1)\sqrt{[\Delta_i^{\text{in}}(t+1)]^2 + [\Delta_i^{\text{rec}}(t)]^2}$;
23:     $y_k(t) = \phi[G_k^{\text{out}}(t) + \epsilon_k^2(t)\Delta_k^{\text{out}}(t)]$;
24: end for
25: Output: $\mathbf{y}$.
```

generalization performance. Three phases can be concisely represented by the pseudocode in Algorithm 1. Codes to reproduce the numerical results provided in the next section are available in our GitHub [37]. We finally remark that during inference and prediction phases, the actual weight configuration is not drawn, and instead, the reparameterized form of the Gaussian fields is used, which is determined by the first and second moments, i.e., μ_{ij}^ℓ and ϱ_{ij}^ℓ.

III. RESULTS AND DISCUSSION

In this section we first apply the MPL in the digit classification task, where an MNIST digit of 784 pixels is divided into a sequence of pixels, and subgroups of 28 pixels are input to the network at each single time step. As a proof of concept, the first example is to show our framework can be applied to any computational tasks of temporal structures. Then, we extend the application to two language processing tasks; one is at the toy level and the other is the real corpus.

A. MNIST digit classification

The recurrent neural network is trained to classify an input image after 28 time steps, seeing 28 pixels at each time step. This task requires long-term memory, because the recurrent neural network makes the final decision only after seeing all the pixels, and the information in the previous time steps (up to 28 steps before) must be stored and processed in the last

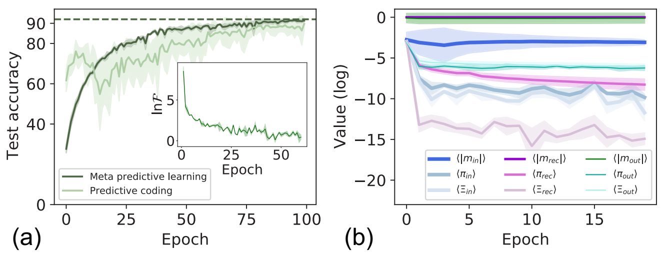

FIG. 1. The performance of meta predictive learning on the 28×28 MNIST classification task. (a) Test accuracy as a function of epoch. The network with $N = 100$ recurrent neurons, $N_{\text{in}} = 28$ input units, and $N_{\text{out}} = 10$ output nodes is trained on the full MNIST dataset with 60 k training images (handwritten digits) and validated on another unseen 10 k test handwritten digits. Predictive coding indicates the learning direct in the weight space rather than the distribution space. If the epoch is less than 40, the number of inference steps is set to $n = 100$, and $n = 200$ otherwise. The inset shows how $\ln\mathcal{F}$ changes with training in the first 60 training epochs (this log-energy becomes stable in the late training stage, and is thus not shown). Five independent runs are considered for the fluctuation of the result. (b) The logarithmic average value of $[\boldsymbol{\Xi}^{\ell}, \boldsymbol{\pi}^{\ell}, \mathbf{m}^{\ell}]$ vs epoch in all layers, the log means logarithm with the base e. Only the first 20 epochs are considered (the result remains stable in the later training stage), and the fluctuation is computed from five independent runs.

step. To carry out this task, we use a vanilla RNN with $N = 100$ recurrent neurons, $N_{\text{in}} = 28$ input units, and $N_{\text{out}} = 10$ output nodes indicating the output class in the one-hot form. The entire dataset is divided into several batches, and we use stochastic gradient descent (SGD) in the learning phase to update the hyperparameters $[\mathbf{m}^{\ell}, \boldsymbol{\pi}^{\ell}, \boldsymbol{\Xi}^{\ell}]$ and an Adam optimizer is applied [38]. Despite working on the network ensemble level and the fact that weight uncertainty must be taken into account during the inference, learning, and prediction phases, our model can achieve better and more stable performances than the predictive coding without any distribution training [Fig. 1(a)]. As expected, the overall energy $\mathcal{F}$ consistently decreases during training and reaches the point near zero in the late stage of training.

The macroscopic behavior of the network is corroborated by the statistical pattern of model hyperparameters underlying synaptic weights, as shown in Fig. 1(b). The weight uncertainty characterized by hyperparameters $[\boldsymbol{\Xi}^{\ell}, \boldsymbol{\pi}^{\ell}]$ decreases over training, showing that the weight is becoming more deterministic, and we use the average value, e.g., $\langle \Xi_{\text{in}} \rangle = \frac{1}{N \times N_{\text{in}}} \sum_{ij} \Xi_{ij}^{\text{in}}$, to compute the average uncertainty level (for the mean $\mathbf{m}$, we take its absolute value before the average is carried out). Interestingly, the uncertainty level is highest in the output layer, which is in striking contrast to the results obtained by a generalized backpropagation through time (*not* the local learning guided by prediction error considered in the current work) at the ensemble level [18] where the uncertainty is highest in the recurrent layer. From the predictive coding perspective, the readout weight has more flexibility to extract the information in the reservoir. On one hand, this higher variability may be due to the local nature of learning that is driven by minimizing the prediction error, and on the other hand, the decision making here is implemented by reading only the belief neural state, which must cooperatively interact with weight statistics to reduce the readout errors. A precise mathematical explanation would be insightful but left for future works. This indicates that a more biological plausible training may lead to different interpretations of the same computational tasks as implemented in neural circuits. Therefore, to reveal biological mechanisms, a biological plausible training is a necessary ingredient.

B. Toy language model

A real language corpus is commonly complicated and not simple for theoretical studies. To build a metaphor for the complicated natural language, we set up a generative process where a text (a stream of tokens) is generated through a fixed rule (similar to grammar). Following this setting, the artificial corpus consists of M texts of length T each, and each text is composed of letters from $a, b, c, \dots, z$. A periodic boundary is applied. For example, a single sample $x = \{a, c, g, i, ...\}$ is generated according to the grammatical rule that starting from letter $'a'$, only the letter $'c'$ or $'e'$ which is located two letters or four letters next to $'a'$ (with equal probabilities) can follow $'a'$, and the case of two consecutive $'c'$ is not allowed. This rule for generating toy language is just a simple model of real corpus but nontrivial enough for a neural network to learn the embedded rule. The generated examples (letter sequences) are shown to the neural network, which is required to discover the rule by our meta predictive learning working on next-letter prediction. After training, the network is tested by generating a sequence of arbitrary length following the same rule. A hierarchical compositional structure can also be incorporated into the generation process, but we leave this more interesting case to future studies based on this toy setting.

A RNN with $N = 100$, $N_{\text{in}} = 26$, $N_{\text{out}} = 26$ is trained on the full dataset following the above rule, with a total of 26 624 (calculated as $26 \times 2^{T-1}$) sequences of length $T = 11$ (other values of T can also be similarly studied), and an SGD with Adam optimizer is applied [38]. To detect a possible phase transition with increasing data size, we can use an increasing portion of the entire dataset (i.e., $M < 26\,624$). Each letter can be encoded into one-hot form before being input to the network, while the readout implements a decoding of the neural activity into one-hot form as well. Because of the simplicity in our letter space, we do not need word embedding as commonly used in language processing [39]. In Fig. 2(a) we can easily generate a sequence of arbitrary length by supplying a network with the letter generated in the previous time step, and the trained network (in the ensemble sense) successfully generates sequences following the ground truth rule. Interestingly, the well-trained network also generates sequences with length $T > 11$ following the same rule, suggesting the possibility that the network output could be creative to generate new grammatically correct sequences.

To study the emergence behavior of this simplified language model, we define the correct letter ratio to characterize the language generating ability of our model. After training, the network instance (sampled from the ensemble) is required to generate 26 sequences of length $T = 11$ whose first letters are one of all 26 letters of the alphabet, and the correct letter ratio is defined as the average ratio of correctly predicted letters. For example, the

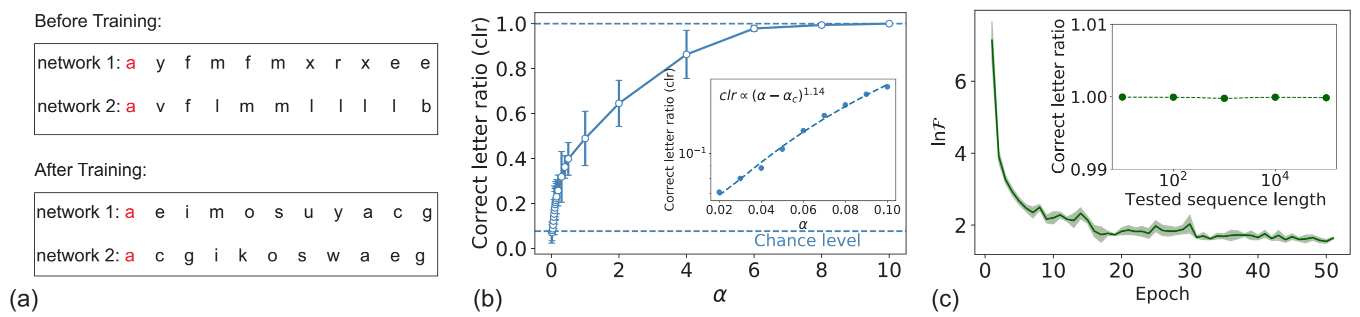

FIG. 2. The properties of meta predictive learning on the simplified language prediction task. The grammatical rule is designed as follows: starting from a random letter ($'a'$ here), only the candidates located two letters or four letters after $'a'$ can follow the starting letter with equal probability, and each letter only repeats once in this next-word generation. All letters in the alphabet form a cyclic structure. $T = 11$ is considered, and the full size of dataset is 26 624. RNN with $N = 100$, $N_{\rm in} = 26$, $N_{\rm out} = 26$ is trained, and two instances of networks are randomly sampled from the (trained or untrained) network ensemble. (a) Starting from the letter a, the network generates the next letter, which serves as the input at the next time step, until a sequence with desired length is generated. (b) The correct letter ratio as a function of data load $\alpha = \frac{M}{N}$, and five independent runs are considered. M examples of sequences are used for training. A chance level of $\frac{1}{13}$ is marked. The inset shows the correct letter ratio in the range of $\alpha \in [0.02, 0.1]$. (c) The log-energy $\ln \mathcal{F}$ changes with training epochs and decreases to near zero. The inset shows how the correct letter ratio changes with the length of generated sequence after a full dataset is used for training. The error bar is computed with five independent networks.

sequence $['a','c','e','g','k','m','o','s','w','a','z']$ has nine correct predictions, with ratio 0.9 (in total the network has to predict ten letters) for this single sequence. Therefore, the correct letter ratio indicates the language generating ability of the network ensemble, with a maximal value of 1 (100%). In Fig. 2(b) we can easily see that the correct letter ratio first remains at a very low level (close to chance level) if the data load $\alpha = \frac{M}{N}$ is small, i.e., the generated sequences are random when $\alpha < 0.02$. Beyond this threshold, the performance continuously improves, exhibiting the phenomenon of a second-order phase transition, which coincides qualitatively with empirical results of emergence discovered in large language models [40,41]. The scaling exponent of the correct letter ratio (order parameter in statistical mechanics [42]) around the transition point is about 1.14. A rigorous derivation of this critical exponent is left for future analytic works. Training RNNs with different network sizes yields qualitatively the same behavior, but a larger network size makes the transition sharper. After the transition, the network assigns the correctly predicted letter with a larger probability than other letter candidates, while the possibilities for other letters are significantly suppressed (see Fig. 3). Another important characteristic is that the learning with increasing data occurs first rapidly, followed by a slow period, and finally the performance is saturated to the perfect generalization of the language rule. This may be interpreted as a hierarchical decoding of the information embedded in the noisy (stochasticity in the generation process) sequences. We further remark that, after a full dataset of sequences with fixed length (e.g., $T = 11$) is trained, the network is able to generate the grammatically correct letter sequences of *arbitrary* length [see the inset of Fig. 2(c)]. The energy of the language model is also decreasing with training until getting stationary, which emphasizes the important role of the energy-based model in understanding recurrent language processing. A further extension of meta predictive learning to transformer structure is possible, as the Gaussian assumption used in the standard predictive coding has been shown to be generalized to arbitrary probability distributions in a recent work [43].

To study the properties of this simplified language model, we plot the distribution of hyperparameters $[\pi, m, \Xi]$ for the input layer, output layer, and recurrent layer (see Fig. 4), respectively. The distribution of $[\pi, \Xi]$ has the L-shape in all layers, while the output layer allows for more variability in both sparsity and variance of the Gaussian slab, which is characterized by a slightly broader distribution of $[\pi, \Xi]$. Extremes $\pi = 0$, $\pi = 1$, and $\Xi = 0$ have particular physics significance. $\pi = 0$ indicates the connection has no sparsity and thus carries important information for the task. The spike mass at $\pi = 1$ implies that the connection is always zero and thus is not important for the task, but none of the connections of our model belong to this case. $\Xi = 0$ shows the corresponding connection is deterministic, because the corresponding Gaussian distribution reduces to a Dirac δ peak. This result is also observed in the 28×28 MNIST classification task. The distribution of hyperparameter m is broadest in the output layer, ranging from -200 to 200, showing the higher-level variability in the connection weight of the output layer. This phenomenon demonstrates that the embedded rule can only be retrieved by using a highly heterogeneous weighting of each neuron's activity in the reservoir, which is particularly interesting from the perspective of neural decoding of language information and probabilistic computation in a biological plausible setting [10,15,30], since our embedded rule is actually a probabilistic generative rule mixed with a predefined grammatical structure. As in the digit classification example, this high variability may be the direct result of local learning and cooperative updates of belief and distribution parameters. We leave a precise mathematical analysis for future work.

C. Experiments on natural language

In this section we apply our MPL algorithm to a more complex language corpus, i.e., the Penn Treebank (PTB) corpus

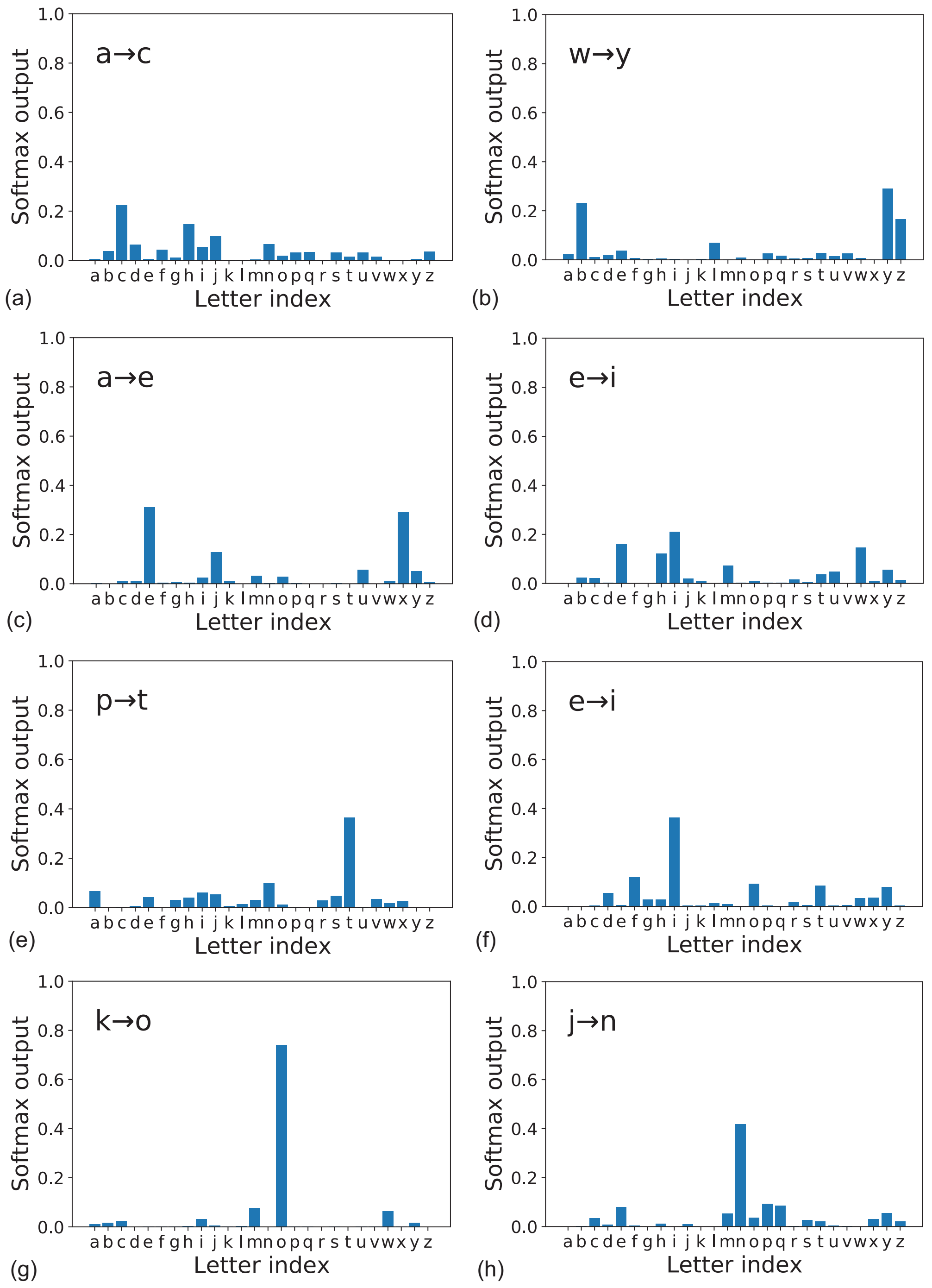

FIG. 3. Softmax values of the output units for different data loads α. Panels (a,b), (c,d), (e,f), and (g,h) show two typical patterns for each data load $\alpha = 0$, $\alpha = 0.01$, $\alpha = 0.03$, and $\alpha = 0.05$, respectively. Only predictions following the designed language rule are displayed, and the text shown in the panel $''a \to c''$ means inputting the letter $'a'$ and the network predicts the immediate following letter $'c'$ (corresponding to the largest softmax output). The training conditions are the same as in Fig. 2.

[27], which contains nearly 50 000 sentences collected from the *Wall Street Journal*. The PTB is one of the most known and used corpus for word-level language modeling.

Due to the large space of the corresponding vocabulary, the corpus needs to be pre-processed using word embedding techniques before sending the sentences into the network [39]. Here we describe the main steps. The first step is to use a tokenizer tool to split the sentences into tokens and replace the useless words or characters with a special token named <unk>, indicating an unknown token. In addition, the tokens that appeared less than five times in the whole corpus will be replaced with <unk> to help the network concentrate on the major tokens of high frequency. Next, we collect all tokens to generate a vocabulary to store all the different tokens. However, directly inputting the tokens (treated as one-hot vectors) into the network is inconvenient when the size of

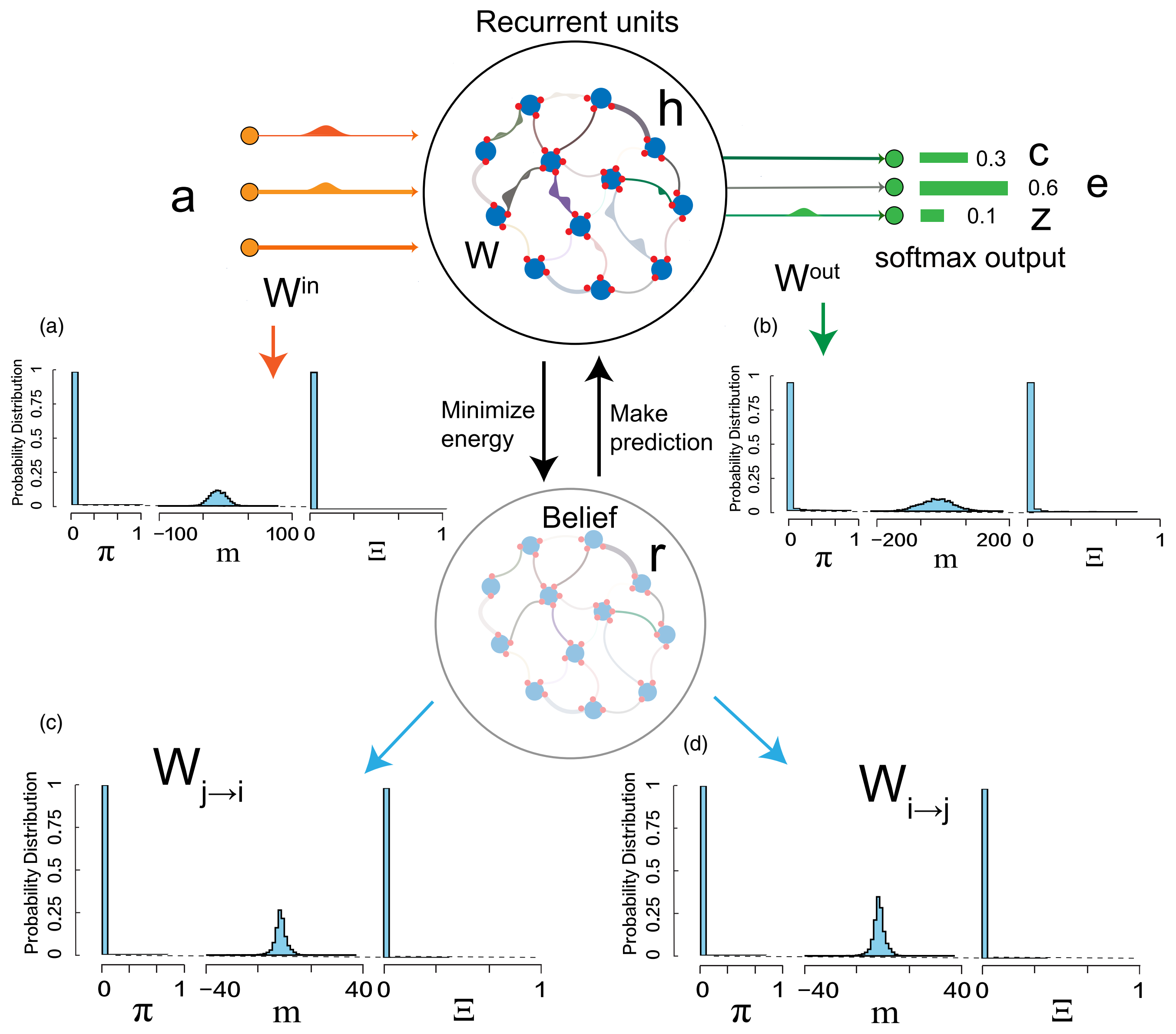

FIG. 4. Illustration of hyperparameters [π, m, Ξ] in meta predictive learning on the simplified language task. The training conditions are the same as in Fig. 2. In (c-d) we show statistical properties of bidirectional connections, and $i < j$ is considered.

vocabulary is large. Hence, we set up a look-up table called an embedding layer to transform every token into vectors in a low-dimensional feature space via neural networks. The training goal is to learn word vector representations that are able to predict the nearby words. Rows of the trained encoding matrix give the distributed representation of words. The embedding layer is trained by a traditional backpropagation algorithm [39], while both the recurrent reservoir and the readout layer are trained by our MPL as described above (or other alternatives if comparison among algorithms is made).

The overall energy for the predictive learning is given below:

$$\mathcal{F} = \frac{1}{2}\sum_t ||\mathscr{E}'_{\text{rec}}(t)||^2 + \mathcal{L}, \tag{8}$$

where $\mathscr{E}'_{\text{rec}}$ denotes the prediction error for the recurrent reservoir, $\mathcal{L}(\mathbf{y}, \mathbf{r}_y) = -\sum_t \sum_i (\mathbf{r}_y)_i(t) \ln y_i(t)$ is related to the readout error. In order to measure the accuracy of the language model, we use a perplexity metric, which measures how well the model predicts the next token, i.e., the uncertainty about the prediction, precisely given by [20]

$$\text{ppl} = \left[\prod_{i=1}^{T} p(w_i|w_{i-1}, \dots, w_0)\right]^{-\frac{1}{T}}. \tag{9}$$

It is intuitive that minimizing the perplexity is equivalent to maximizing the probability of a corpus which is composed of T words indicated by $\{w_0, w_1, \dots, w_T\}$. Because the output of our model $y_i(t)$ is actually the prediction probability in the nonlinear softmax form, we can recast the perplexity as

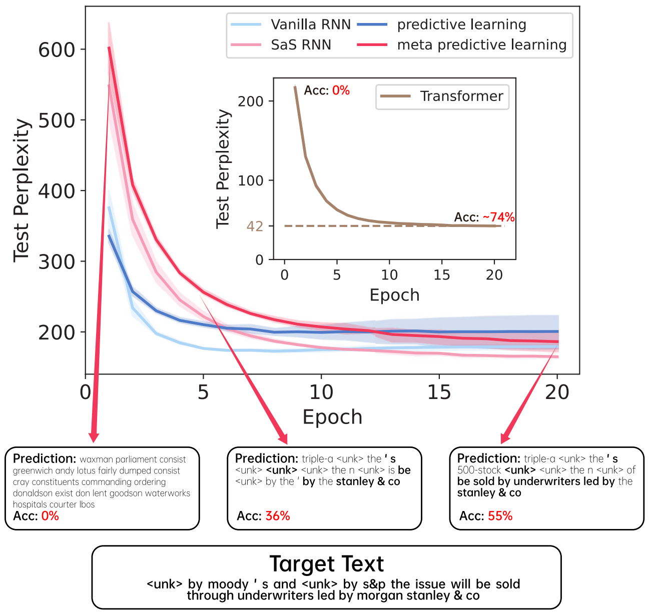

FIG. 5. Training performance of networks with different architectures in the Penn Treebank dataset. In the upper part of the figure, we choose the vanilla RNN [42], SaS RNN (ensemble learning) [18], RNN with standard predictive coding [12], and RNN with meta predictive learning to show how test perplexity decreases with the training epoch. The first two algorithms belong to the backpropagation through time category [42]. In the inset we provide the performance of transformer model (see details in Appendix B) with single encoder block for comparison. We also mark the mean test accuracy of the transformer model at the beginning of training and at the end of training. In the bottom part of the figure, we select untrained, trained-for-five-epoch, and full-trained RNNs with meta predictive learning to show the performances at different training stages in generating one of the sentences in the test dataset. The correctly predicted tokens from the test sentence are highlighted, while the wrongly predicted tokens are gray colored. The indicated accuracy is the ratio of the number of correctly predicted tokens from the test sentence to the total number of tokens in the sentence. The mean accuracy evaluated from 100 sentences is about 0%, $21.3\% \pm 10.5\%$, $23.5\% \pm 11.3\%$ at the three shown stages, respectively. Note that all the models share the same training hyperparameters, such as batch size, learning rate, and training optimizers (see Appendix B for details).

$\text{ppl} = e^{\mathcal{L}}$, where $\mathcal{L}$ represents the cross-entropy objective used to train the network.

We apply our MPL to model this real corpus with comparison among other competitive algorithms (see Fig. 5). The test perplexity obtained by backpropagation through time with/without meta learning reaches a similar level to those obtained by standard predictive coding and our MPL method, after tens of epochs. A salient feature is that those trainings without the SaS premise get easily overfitted at later stages of training. For comparison, the transformer network with self-attention blocks (see Appendix B for details) achieves the lowest perplexity among all considered methods, which demonstrates that the current biological recurrent computation has still a large gap to the artificial transformer computation, where the input tokens are not shown to the network in sequence but in the form of a single block, such that the self-attention is able to integrate information from different parts of the input. This also implies that new elements, e.g., attention mechanisms, possibly in to-be-revealed biological forms, might be added to our current framework to minimize the gap on one hand, and on the other hand to develop a biological computational model of intelligent systems that can handle natural language, particularly without relying on a long working memory and an astonishingly large corpus.

To study the network behavior, we plot the distribution of hyperparameters m, π, Ξ when the RNN network is trained with the MPL method, as shown in Fig. 6. We find that the mean weight m for all layers is symmetrically distributed around zero, with a relatively narrow distribution. The distribution of π for all layers is of an L shape and peaks at $\pi = 0$, indicating a dense network is favored and formed after learning. The distribution of Ξ is of the U shape and has two peaks. One peak is at $\Xi = 0$, indicating that these weights are deterministic and could only take a single value of m, and the other peak is at $\Xi \simeq 0.01$, indicating that the corresponding connection can carry a range of candidate values. Currently, it remains unknown how to relate these microscopic details of the network structure to the decoding of the semantic information in the corpus. It is thus important in future works to design an analytically tractable model of language processing bridging neurophysiological plausibility and superperformance observed in the state-of-the-art architectures, which would help to uncover key neuron, synapse, and circuit motif types in the human brain.

IV. CONCLUSION

Predictive coding is a prediction-error-driven learning with local updates, performing a joint process of both inference and learning, thereby being a potential candidate for how the brain builds an internal generative model of the complex, evolving outside world [2]. We take the predictive coding within the language processing context, which is currently attracting intense research interest due to ChatGPT [1]. We propose a meta predictive learning method in recurrent neural networks, which encode and predict tokens in text sequences, in the presence of uncertainty. A continuous phase transition is revealed in our model, and perfect generation can be achieved after a sufficient number of training sequences are provided. Therefore, our toy model provides a good starting point to dissect the mechanism of learning in language processing [2].

Our MPL framework is relatively not prone to overfitting in training real corpus. In addition, there emerge intriguing statistical patterns of hyperparameters in our networks. However, it remains unclear how these statistical properties explain the performance (e.g., accuracy in next-token predictions) of the recurrent computation, which highly resembles what occurs in human brains. In contrast, the self-attention leveraged in transformer networks is not biological (e.g., not recurrent and nonlocal learning). Nevertheless, the transformer structure leads to emergence of intelligence to some extent, and in particular the phenomenon of in-context learning, where the trained network can perform novel tasks by a prompting

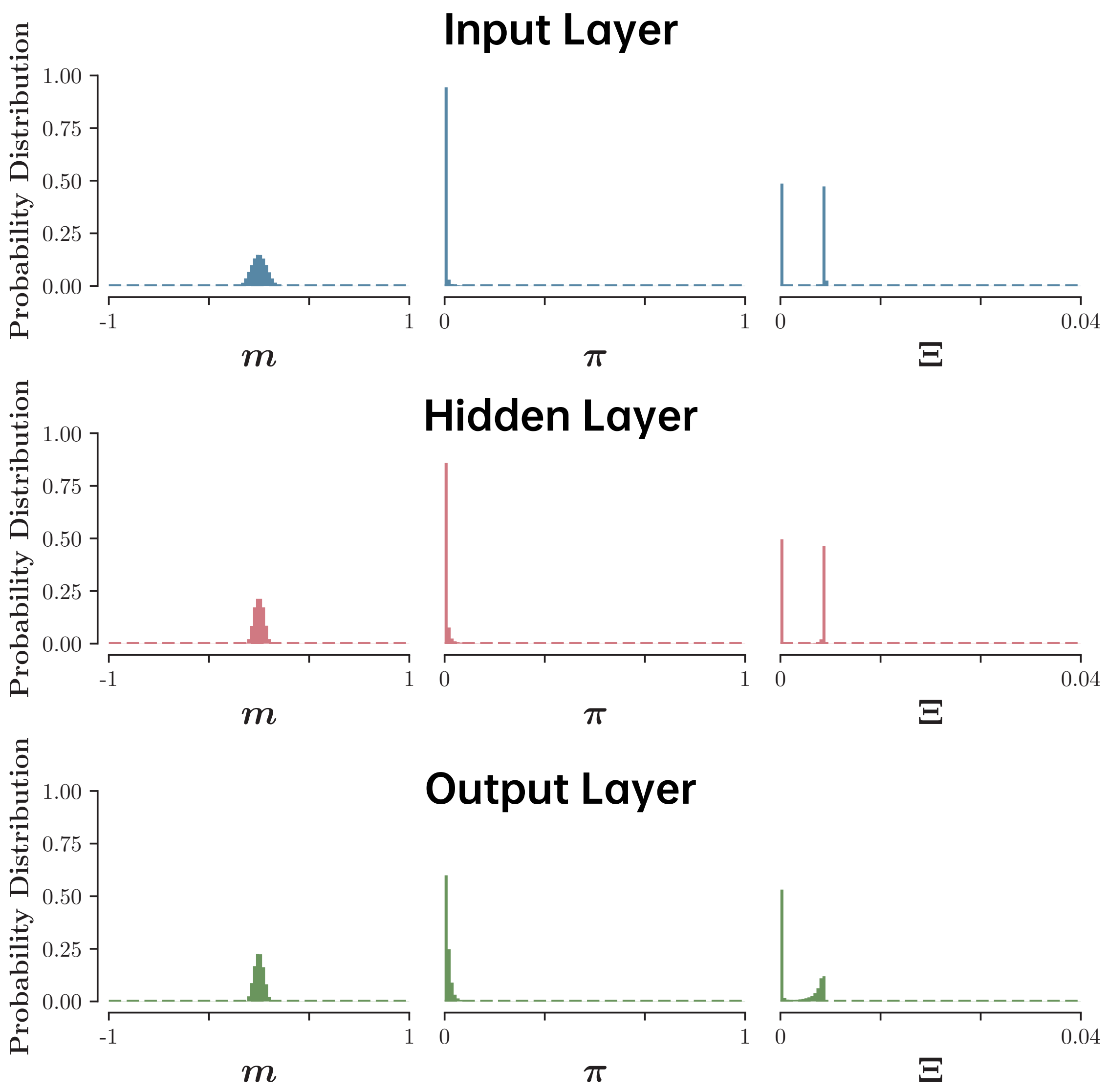

FIG. 6. Probability distribution of hyperparameters m, π, Ξ in the RNN networks trained with meta predictive learning. Distributions of hyperparameters m, π, Ξ at the input layer are shown at the top of the figure (blue histogram), and those at the hidden layer and output layer are shown at the middle (salmon histogram) and at the bottom (green histogram) of the figure, respectively.

of example demonstrations without any further learning. The ability of in-context learning emerges by only scaling models and computation costs [41]. The deviation from known brain computation for language processing triggers a hot debate on what the nature of intelligence is [44] and whether the intelligence can be achieved by next-token prediction [30]. More precisely, how a compressed representation of hierarchical compositional structure in linguistic data can be achieved by biological learning (or resulting in multitask performances beyond transformer) remains largely mysterious. Our current study shows that meta predictive learning for language processing may be a fruitful route toward this goal.

A recent work demonstrated that the weight uncertainty with the form of SaS structure can be also incorporated into the transformer [45]. In addition, gated recurrent neural networks with multiplicative mechanisms were recently shown to be able to learn to implement linear self-attention [46]. Furthermore, the relationship between linear transformers allowing for faster autoregressive learning and RNNs was established in a recent work [47]. Taken together, our current work would be a starting point to establish the bridge between the biological learning (towards the science of specialized brain circuits) and transformer learning within the seminal predictive coding hypothesis, which can be put in the theoretically solid variational free energy minimization conceptual framework.

ACKNOWLEDGMENTS

This research was supported by the National Natural Science Foundation of China for Grant No. 12122515 (H.H.), Guangdong Provincial Key Laboratory of Magnetoelectric Physics and Devices (Grant No. 2022B1212010008), and Guangdong Basic and Applied Basic Research Foundation (Grant No. 2023B1515040023).

APPENDIX A: INTERPRETATION OF PREDICTIVE CODING AS VARIATIONAL FREE ENERGY MINIMIZATION

For a recurrent dynamics, we write the neural activity at each step as a latent variable $\mathbf{r}(t)$, and then it is reasonable to assume the joint probability of a trajectory can be written into the following Markovian form:

$$P[\mathbf{r}(0),\ldots,\mathbf{r}(T)] = P[\mathbf{r}(0)]\prod_{t=1}^{T} P[\mathbf{r}(t)|\mathbf{r}(t-1)]. \quad \text{(A1)}$$

We further assume a Gaussian form for the transition probability $P[\mathbf{r}(t)|\mathbf{r}(t-1)] = \mathcal{N}[\mathbf{r}(t); \mathbf{h}(t), \sigma_t^2\mathbf{I}]$, where $\mathbf{h}(t) = \mathbf{w}f[\mathbf{r}(t-1)]$, and σ_t^2 is a time-dependent variance, and for simplicity, we can set the variance to 1 without loss of generality, as the mere effect is leading to a rescaled cost function below. This Gaussian form can be obtained as an approximation by using the Laplace approximation even if the transition probability is of other forms. The goal is to optimize the negative log-likelihood of the joint distribution, defined by

$$\begin{aligned}\mathcal{F} &= -\ln P[\mathbf{r}(0),\ldots,\mathbf{r}(T)] \\ &= \frac{1}{2}\sum_t \frac{\|\mathbf{r}(t)-\mathbf{h}(t)\|^2}{\sigma_t^2} + \text{const},\end{aligned} \quad \text{(A2)}$$

which corresponds exactly to the cost function of predictive coding if we treat $\sigma_t^2 = 1$ and neglect the constant term.

APPENDIX B: TRANSFORMER MODEL

A transformer network consists of an embedding layer, encoder blocks, and decoder blocks [48]. In analogy to the RNN model, all tokens (one-hot vectors) are transformed into representations $\mathbf{X} \in \mathbb{R}^{d\times T}$ by an embedding layer, where d denotes the dimension of embedding space and T denotes the sequence length. As a clear difference from the RNN training, the input to the transformer is an entire $\mathbf{X}$ matrix rather than one column for each step during training RNNs. Note that we have not considered the position encoding scheme (e.g., adding a vector of sinusoids of different frequencies and phases to encode position of a word in a sentence) in our model.

An encoder block includes two parts. The first part is the self-attention mechanism, aiming to evaluate the correlations among words in the input block $\mathbf{X}$. To this end, we introduce three trainable matrices, namely, query Q, key K, and value V. Then a linear transformation of the input is applied:

$$\begin{aligned} Q &= W_Q\cdot\mathbf{X}, \\ K &= W_K\cdot\mathbf{X}, \\ V &= W_V\cdot\mathbf{X}, \end{aligned} \quad \text{(B1)}$$

where $W_Q, W_K \in \mathbb{R}^{d_h\times d}$, and $W_V \in \mathbb{R}^{d\times d}$ are transformation matrices, and d_h is the internal size of the attention operation. Therefore, we define X_t as the tth column of $\mathbf{X}$, and then we can define three vectors, namely, $k_t = W_K X_t$, $v_t = W_V X_t$, and $q_t = W_Q X_t$. Then the tth column of the self-attention matrix SA($\mathbf{X}$) is given by

$$\begin{aligned} \text{attn}(t) &= \sum_{i=1}^{T}\alpha_i(t)v_i, \\ \alpha_i(t) &= \frac{e^{k_i^\top q_t/\sqrt{d_h}}}{\sum_{j=1}^{T} e^{k_j^\top q_t/\sqrt{d_h}}}, \end{aligned} \quad \text{(B2)}$$

where $\alpha_i(t)$ is a softmax operation containing information about the pairwise interactions between tokens. The normalization factor $\sqrt{d_h}$ is required to retain relevant quantities in the exponential function being of the order 1.

The second part is two feed-forward layers with skip connection, i.e.,

$$\begin{aligned} \mathbf{z}_1 &= \text{SA}(\mathbf{X}) + \mathbf{X} && \text{(residual layer 1)} \\ \mathbf{z}_2 &= \text{ReLU}(W_1\cdot\mathbf{z}_1 + b_1) && \text{(feed-forward layer 1)} \\ \mathbf{z}_3 &= W_2\cdot\mathbf{z}_2 + b_2 && \text{(feed-forward layer 2)} \\ \mathbf{z}^{\text{out}} &= \mathbf{z}_1 + \mathbf{z}_3 && \text{(residual layer 2)}, \end{aligned} \quad \text{(B3)}$$

where W_1, W_2 and b_1, b_2 are weights and biases of the two feed-forward layers. The output representations $\mathbf{z}^{\text{out}}$ can be considered to be the input of the next encoder block. Here, we use the single headed attention transformer and do not use the layer normalization, which scales each element of a vector by the mean and variance of all elements in that vector.

Our used transformer model has only one encoder block and one decoder layer. The decoder layer is a linear layer (the readout layer), where the output representations $\mathbf{z}^{\text{out}}$ can be translated into the probability of the next token, which has the same function as the readout layer of the RNN model. The dimension of representations $d = 300$ for all models in Fig. 5. For four RNN models, the number of neurons in the recurrent reservoir is $N = 512$. For the transformer model, it is convenient to set the hidden dimension $d_h = d = 300$. The training parameters for all models are set to be the same. The batch size is 128, and the learning rate is 0.001. We have chosen the Adam algorithm as our training optimizer [38].

APPENDIX C: THE VANILLA PREDICTIVE LEARNING ALGORITHM

The vanilla predictive learning algorithm is a simplified version of our meta predictive learning algorithm, without considering weight uncertainty. Hence, setting $\boldsymbol{\pi} = 0$ and $\Xi = 0$ in Eq. (6) and Eq. (7) in the main text leads to the following update equations for belief and weights:

$$\begin{aligned} \Delta r_i(t') = &-\gamma\mathcal{E}'_{1,i}(t') + \gamma f'[r_i(t')] \\ &\times\sum_j \mathcal{E}'_{1,j}(t'+1)w_{ji} + \gamma f'[r_i(t')] \\ &\times\sum_j \mathcal{E}'_{2,j}(t')w_{ji}^{\text{out}}, \end{aligned} \quad \text{(C1)}$$

and

$$\Delta w_{ij}^{\ell} = \frac{\eta}{N_\ell}\sum_t \mathcal{E}'_{\ell',i}(t)\xi_j^{\ell}, \quad \text{(C2)}$$

TABLE I. Mathematical items used in the main text and associated explanations.

Item	Explanation
$r_i(t)$	*belief* of neuron i at time step t
$h_i(t)$	prediction of neuron i at time step t
$y_i(t)$	activity of output unit i at time step t
$\hat{\mathbf{y}}$	label of input
$f(\cdot)$	nonlinear activation function for recurrent dynamics
$\phi(\cdot)$	nonlinear activation function for network output
ℓ	in, out or recurrent (no superscript or subscript)
N_ℓ	the number of neurons for ℓ
w_{ij}^ℓ	directed coupling from neuron j to neuron i
μ_{ij}^ℓ	first moment of w_{ij}^ℓ: $\mu_{ij}^\ell = \frac{m_{ij}^\ell}{N_\ell}$
ϱ_{ij}^ℓ	second moment of w_{ij}^ℓ: $\varrho_{ij}^\ell = \frac{(m_{ij}^\ell)^2}{N_\ell^2(1-\pi_{ij}^\ell)} + \frac{\Xi_{ij}^\ell}{N_\ell}$
$G_i^{\text{in}}(t+1)$	input mean current: $G_i^{\text{in}}(t+1) = \sum_j \mu_{ij}^{\text{in}} x_j(t+1)$
$G_i^{\text{rec}}(t+1)$	recurrent mean current: $G_i^{\text{rec}}(t+1) = \sum_j \mu_{ij} f[r_j(t+1)]$
$G_k^{\text{out}}(t+1)$	output mean current: $G_k^{\text{out}}(t+1) = \sum_j \mu_{kj}^{\text{out}} f[r_j(t+1)]$
$\Delta_i^{\text{in}}(t+1)$	input fluctuation: $\sqrt{\sum_j (\varrho_{ij}^{\text{in}} - (\mu_{ij}^{\text{in}})^2)[x_j(t+1)]^2}$
$\Delta_i^{\text{rec}}(t+1)$	recurrent fluctuation: $\sqrt{\sum_j [\varrho_{ij} - (\mu_{ij})^2](f[r_j(t+1)])^2}$
$\Delta_k^{\text{out}}(t+1)$	output fluctuation: $\sqrt{\sum_j (\varrho_{kj}^{\text{out}} - (\mu_{kj}^{\text{out}})^2)(f[r_j(t+1)])^2}$
Δ_i^1	$\Delta_i^1 = [\Delta_i^{\text{in}}(t)]^2 + [\Delta_i^{\text{rec}}(t-1)]^2$
Δ_i^2	$\Delta_i^2 = [\Delta_i^{\text{out}}(t)]^2$
$\boldsymbol{\epsilon}^1, \boldsymbol{\epsilon}^2$	standard Gaussian variables
$\mathscr{E}_1(t)$	$\mathscr{E}_1(t) = \frac{1}{2}\lVert\mathbf{r}(t) - \mathbf{h}(t)\rVert^2$
$\mathscr{E}_2(t)$	$\mathscr{E}_2(t) = -\sum_i \hat{y}_i(t)\ln[y_i(t)]$
$\mathcal{F}$	energy cost: $\mathcal{F} = \sum_{j=1}^2 \sum_{t=1}^T \mathscr{E}_j(t)$
$\mathscr{E}_1'(t)$	prediction error vector 1: $\mathscr{E}_1'(t) = \mathbf{r}(t) - \mathbf{h}(t)$
$\mathscr{E}_2'(t)$	prediction error vector 2: $\mathscr{E}_2'(t) = \hat{\mathbf{y}}(t) - \mathbf{y}(t)$
$\hat{\epsilon}_{ji}^1$	$\hat{\epsilon}_{ji}^1 = \epsilon_j^1(t'+1)\frac{(\varrho_{ji} - (\mu_{ji})^2)f'[r_i(t')]f[r_i(t')]}{\sqrt{\Delta_j^1}}$
$\hat{\epsilon}_{ji}^2$	$\hat{\epsilon}_{ji}^2 = \epsilon_j^2(t')\frac{[\varrho_{ji}^{\text{out}} - (\mu_{ji}^{\text{out}})^2]f'[r_i(t')]f[r_i(t')]}{\sqrt{\Delta_j^2}}$
γ	learning rate for the inference phase
η	learning rate for the learning phase
ξ_j^ℓ	$\xi_j^{\text{in}} = x_j(t)$, $\xi_j = f[r_j(t-1)]$, $\xi_j^{\text{out}} = f[r_j(t)]$
M	the number of training sequence examples
α	data load $\alpha = \frac{M}{N}$
T	sequence length

ALGORITHM 2. Vanilla predictive coding algorithm.

```
1: # Inference
2: Given: input $\mathbf{x}$, label $\hat{\mathbf{y}}$, randomly initialized belief $\mathbf{r}$, $\mathbf{r}_y = \hat{\mathbf{y}}$
3: for $i = 1, \ldots, n$ do
4:     for $t = 1, \ldots, T$ do
5:         $h_i(t) = \sum_{j=1}^N w_{ij} f[r_j(t-1)] + \sum_{j=1}^{N_{\text{in}}} w_{ij}^{\text{in}} x_j(t)$;
6:         $y_i(t) = \phi(\sum_{j=1}^N w_{ij}^{\text{out}} f[r_j(t)])$;
7:         $\mathbf{r}(t) = \mathbf{r}(t) + \Delta\mathbf{r}(t)$.
8:     end for
9: end for
10: # Learning
11: for $\ell$ = in, out, recurrent do
12:     for $t = 1, \ldots, T$ do
13:         $\mathbf{w}^\ell = \mathbf{w}^\ell + \Delta\mathbf{w}^\ell$.
14:     end for
15: end for
16: Output: $\mathbf{r}$
17: # Prediction
18: Given: test data $\mathbf{x}$, converged belief $\mathbf{r}$
19: for $t = 1, \ldots, T$ do
20:     $h_i(t) = \sum_{j=1}^N w_{ij} f[r_j(t-1)] + \sum_{j=1}^{N_{\text{in}}} w_{ij}^{\text{in}} x_j(t)$
21:     $y_i(t) = \phi(\sum_{j=1}^N w_{ij}^{\text{out}} f[r_j(t)])$
22: end for
23: Output: $\mathbf{y}$
```

where the definition of ℓ, ℓ', ξ_j^ℓ, and N_ℓ bear the same meaning as in the main text (see Table I). We present the pseudocode of the vanilla predictive learning algorithm in Algorithm 2.

APPENDIX D: MATHEMATICAL ITEMS USED IN THE MAIN TEXT AND ASSOCIATED EXPLANATIONS

We list mathematical items used in the main text and associated explanations to help readers go through the paper smoothly, as shown in the Table I.

[1] S. Bubeck, V. Chandrasekaran, R. Eldan, J. A. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y.-F. Li, S. M. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang, Sparks of artificial general intelligence: Early experiments with GPT-4, arXiv:2303.12712.

[2] H. Huang, Eight challenges in developing theory of intelligence, arXiv:2306.11232.

[3] R. P. N. Rao and D. H. Ballard, Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects, Nat. Neurosci. **2**, 79 (1999).

[4] Y. Huang and R. P. N. Rao, Predictive coding, WIREs Cognit. Sci. **2**, 580 (2011).

[5] J. C. R. Whittington and R. Bogacz, An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity, Neural Comput. **29**, 1229 (2017).

[6] B. Millidge, A. Seth, and C. L. Buckley, Predictive coding: A theoretical and experimental review, arXiv:2107.12979.

[7] K. Friston, Does predictive coding have a future? Nat. Neurosci. **21**, 1019 (2018).

[8] A. M. Bastos, W. Martin Usrey, R. A. Adams, G. R. Mangun, P. Fries, and K. J. Friston, Canonical microcircuits for predictive coding, Neuron **76**, 695 (2012).

[9] Y. Chen, H. Zhang, and T. J. Sejnowski, Hippocampus as a generative circuit for predictive coding of future sequences, bioRxiv doi: https://doi.org/10.1101/2022.05.19.492731 (2022).

[10] L. Wang, L. Schoot, T. Brothers, E. Alexander, L. Warnke, M. Kim, S. Khan, M. Hämäläinen, and G. R. Kuperberg, Predictive coding across the left fronto-temporal hierarchy during language comprehension, Cereb. Cortex **33**, 4478 (2023).

[11] S. Golkar, T. Tesileanu, Y. Bahroun, A. Sengupta, and D. Chklovskii, Constrained predictive coding as a biologically

plausible model of the cortical hierarchy, in *Advances in Neural Information Processing Systems*, edited by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Curran Associates, Inc., Red Hook, NY, 2022), Vol. 35, pp. 14155–14169.

[12] T. Salvatori, Y. Song, T. Lukasiewicz, R. Bogacz, and Z. Xu, Predictive coding can do exact backpropagation on convolutional and recurrent neural networks, arXiv:2103.03725.

[13] B. Millidge, T. Salvatori, Y. Song, R. Bogacz, and T. Lukasiewicz, Predictive coding: Towards a future of deep learning beyond backpropagation? *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, edited by Luc De Raedt (IJCAI/AAAI Press, 2022), pp. 5538–5545.

[14] Z.-Y. Huang, R. Zhou, M. Huang, and H.-J. Zhou, Energy–information trade-off induces continuous and discontinuous phase transitions in lateral predictive coding, arXiv:2302.11681.

[15] A. Pouget, J. M. Beck, W. J. Ma, and P. E. Latham, Probabilistic brains: Knowns and unknowns, Nat. Neurosci. **16**, 1170 (2013).

[16] H. Kasai, Noam E. Ziv, H. Okazaki, S. Yagishita, and T. Toyoizumi, Spine dynamics in the brain, mental disorders and artificial neural networks, Nat. Rev. Neurosci. **22**, 407 (2021).

[17] T. Salvatori, A. Mali, C. L. Buckley, T. Lukasiewicz, R. P. N. Rao, K. Friston, and A. Ororbia, Brain-inspired computational intelligence via predictive coding, arXiv:2308.07870.

[18] W. Zou, C. Li, and H. Huang, Ensemble perspective for understanding temporal credit assignment, Phys. Rev. E **107**, 024307 (2023).

[19] Sepp Hochreiter and Jurgen Schmidhuber, Long short-term memory, Neural Comput. **9**, 1735 (1997).

[20] Y. Bengio, Réjean Ducharme, P. Vincent, and C. Janvin, A neural probabilistic language model, J. Mach. Learn. Res. **3**, 1137 (2003).

[21] I. Sutskever, O. Vinyals, and Q. V. Le, Sequence to sequence learning with neural networks, *Advances in Neural Information Processing Systems 27* (Curran Associates, Inc., 2014), pp. 3104–3112.

[22] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Association for Computational Linguistics, 2014).

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv:1412.3555.

[24] D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, in *ICLR 2015: International Conference on Learning Representations* (2015).

[25] Y. Lakretz, G. Kruszewski, T. Desbordes, D. Hupkes, S. Dehaene, and M. Baroni, The emergence of number and syntax units in LSTM language models, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Association for Computational Linguistics, 2019), pp. 11–20.

[26] Y. LeCun, The MNIST database of handwritten digits, retrieved from http://yann.lecun.com/exdb/mnist.

[27] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, Building a large annotated corpus of English: The Penn Treebank, Comput. Linguist. **19**, 313 (1993).

[28] C. Li and H. Huang, Learning credit assignment, Phys. Rev. Lett. **125**, 178301 (2020).

[29] C. Caucheteux and J.-R. King, Brains and algorithms partially converge in natural language processing, Commun. Biol. **5**, 134 (2022).

[30] K. Mahowald, A. A. Ivanova, I. A. Blank, N. Kanwisher, J. B. Tenenbaum, and E. Fedorenko, Dissociating language and thought in large language models: A cognitive perspective, Trends Cognitive Sci. (2024).

[31] A. A. Faisal, L. P. J. Selen, and D. M. Wolpert, Noise in the nervous system, Nat. Rev. Neurosci. **9**, 292 (2008).

[32] L. Luo, Architectures of neuronal circuits, Science **373**, 1103 (2021).

[33] R. Rosenbaum, On the relationship between predictive coding and backpropagation, PLoS ONE **17**, e0266102 (2022).

[34] J. Sacramento, R. Ponte Costa, Y. Bengio, and W. Senn, Dendritic cortical microcircuits approximate the backpropagation algorithm, *Advances in Neural Information Processing Systems*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., Red Hook, NY, 2018), Vol. 31, pp. 8721–8732.

[35] B. Barbour, N. Brunel, V. Hakim, and J.-P. Nadal, What can we learn from synaptic weight distributions? Trends Neurosci. **30**, 622 (2007).

[36] H. Huang, Role of zero synapses in unsupervised feature learning, J. Phys. A: Math. Theor. **51**, 08LT01 (2018).

[37] https://github.com/Qjbtiger/Meta-predictive-coding.

[38] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[39] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, Distributed representations of words and phrases and their compositionality, in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pp. 3111–3119 (Curran Associates Inc., Red Hook, NY, 2013).

[40] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, Scaling laws for neural language models, arXiv:2001.08361.

[41] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, Ed. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, Emergent abilities of large language models, Trans. Mach. Learn. Res. (2022), https://openreview.net/forum?id=yzkSU5zdwD.

[42] H. Huang, *Statistical Mechanics of Neural Networks* (Springer, Singapore, 2022).

[43] L. Pinchetti, T. Salvatori, Y. Yordanov, B. Millidge, Y. Song, and T. Lukasiewicz, Predictive coding beyond Gaussian distributions, *Advances in Neural Information Processing Systems*, edited by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Curran Associates, Inc., 2022), Vol. 35, pp. 1280–1293.

[44] T. J. Sejnowski, Large language models and the reverse Turing test, Neural Comput. **35**, 309 (2023).

[45] J. Zhao, B. Xie, and X. Li, Weight uncertainty in transformer network for the traveling salesman problem, in *2023 International Symposium of Electronics Design Automation (ISEDA)* (2023), pp. 219–224.

[46] N. Zucchet, S. Kobayashi, Y. Akram, J. van Oswald, M. Larcher, A. Steger, and J. Sacramento, Gated recurrent neural networks discover attention, arXiv:2309.01775.

[47] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, Transformers are RNNs: Fast autoregressive transformers with linear attention, *Proceedings of the 37th International Conference on Machine Learning*, edited by H. Daumé III and A. Singh, Proceedings of Machine Learning Research Vol. 119 (2020), pp. 5156–5165.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention is all you need, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17 (Curran Associates, Red Hook, NY, 2017), pp. 6000–6010.