

# Binding threshold units with artificial oscillatory neurons

Vladimir Fanaskov  
AIRI, Skoltech  
fanaskov.vladimir@gmail.com

Ivan Oseledets  
AIRI, Skoltech

## Abstract

Artificial Kuramoto oscillatory neurons were recently introduced as an alternative to threshold units. Empirical evidence suggests that oscillatory units outperform threshold units in several tasks including unsupervised object discovery and certain reasoning problems. The proposed coupling mechanism for these oscillatory neurons is heterogeneous, combining a generalized Kuramoto equation with standard coupling methods used for threshold units. In this research note, we present a theoretical framework that clearly distinguishes oscillatory neurons from threshold units and establishes a coupling mechanism between them. We argue that, from a biological standpoint, oscillatory and threshold units realise distinct aspects of neural coding: roughly, threshold units model intensity of neuron firing, while oscillatory units facilitate information exchange by frequency modulation. To derive interaction between these two types of units, we constrain their dynamics by focusing on dynamical systems that admit Lyapunov functions. For threshold units, this leads to Hopfield associative memory model, and for oscillatory units it yields a specific form of generalized Kuramoto model. The resulting dynamical systems can be naturally coupled to form a Hopfield-Kuramoto associative memory model, which also admits a Lyapunov function. Various forms of coupling are possible. Notably, oscillatory neurons can be employed to implement a low-rank correction to the weight matrix of a Hopfield network. This correction can be viewed either as a form of Hebbian learning or as a popular LoRA method used for fine-tuning of large language models. We demonstrate the practical realization of this particular coupling through illustrative toy experiments.

## 1 Introduction

Artificial neural network consists of simplified abstract artificial neurons that follow principles of McCulloch–Pitts model [48]. The elementary constituents, artificial neurons or threshold units<sup>1</sup>, linearly integrate incoming information and apply simple nonlinearity before passing it to the next neuron. In approximation theory such representations are known as superpositions and demonstrate remarkable properties not found in linear approximation schemes [10], [5], [60]. The threshold unit is not a realistic model of a biological neuron but is well-suited to capture time-averaged neuron interactions in a simplified abstract way [44]. Organized into layers, these artificial neurons form hierarchical multilayer networks capable of modeling both feedforward and feedback connections, structures also crucial in biological systems [41]. Despite their simplicity, networks of these units have achieved remarkable state-of-the-art performance across a wide range of cognitive tasks [55], [38].

More recently, a novel oscillatory artificial neuron was proposed [49], aiming to incorporate the crucial role of oscillatory dynamics observed in biological brain functions [7], [13], [17], [57], [30]. Mathematically, a key property of these oscillatory neurons is their capacity for synchronization, often modeled by the Kuramoto equation [37], [12]. This equation describes how interconnected oscillators tend to synchronize their rhythms by exchanging information based on the oscillation phase differences. This synchronization mechanism allows for information propagation across the network and the formation of dynamically coherent groups. Oscillatory neurons, and related models have already shown promising results in tasks such as object discovery, segmentation, and reasoning [49], [43], [40].

<sup>1</sup>Formally, we consider “second generation” neurons with continuous activation functions [44].

Dynamics of oscillatory artificial neurons is fairly constrained by the form of generalized Kuramoto equation. To enhance the expressiveness of these units, the work in [49] introduced several simplifying assumptions. As a result, proposed architecture is a complicated mixture of Kuramoto update and standard deep learning layers which is performant, but lacking theoretical justification and coherence.

In this research note, we present a more theoretically grounded approach to integrate threshold units with oscillatory neurons. Our central argument is that threshold units and oscillatory units model distinct aspects of biological neuronal interaction, compel a clear separation in their dynamical updates. Specifically, threshold units model interactions based on averaged neuronal activity, akin to the Hopfield associative memory model [22], while oscillatory units capture interactions through frequency modulation, following the generalized Kuramoto model [42]. To bridge these two types of units, we demonstrate that under specific technical conditions, a generalized Kuramoto model possesses a global energy function, which can be naturally extended to serve as a Lyapunov function for our proposed Hopfield-Kuramoto associative memory model that contain coupling between oscillatory and threshold neurons. We offer several interpretations for the resulting interaction terms, notably they can be understood as a form of “fast weights” [3], a time-dependent low-rank correction (LoRA) [25], or a Hebbian learning mechanism for the weight matrix of threshold and oscillatory units [21], [1, Definition 2].

The concise list of contribution is as follows:

1. We introduce coupling between artificial oscillatory and threshold units.
2. To theoretically justify this coupling, a novel Hopfield-Kuramoto associative memory model with a provable global energy function is proposed.
3. Our work develops a general framework for constructing deep neural networks that integrate both oscillatory and threshold units.
4. Several interpretations of the derived coupling mechanisms are offered, including connections to fast weights, LoRA, and Hebbian learning.
5. Simple numerical experiments validate the effectiveness of a low-rank coupling mechanism.

## Notation

Vectors, vector functions, matrices are in bold font and their components are in regular font. For example,  $\mathbf{g}(\mathbf{x})$  is a vector function of vector argument  $\mathbf{x}$ ,  $g(\mathbf{x})_3$  is its third component;  $\mathbf{W}_{ij}$  is a block matrix;  $\mathbf{I}_N \in \mathbb{R}^{N \times N}$  is identity matrix, subscript is omitted when dimension is evident from the context;  $\mathbf{1}$  is identity vector, i.e.,  $(\mathbf{1})_i = 1$  for all components.

We use  $\dot{f}$  to indicate time derivatives. In all initial-value problems initial conditions are assumed to be given. We omit them when appropriate.

Matrix operations that we use include Hadamard product  $(\mathbf{A} \odot \mathbf{B})_{ij} = A_{ij}B_{ij}$ , tensor product

$$\begin{pmatrix} b_1 & b_2 \end{pmatrix} \otimes \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} b_1 a_{11} & b_1 a_{12} & b_2 a_{11} & b_2 a_{12} \\ b_1 a_{21} & b_1 a_{22} & b_2 a_{21} & b_2 a_{22} \end{pmatrix},$$

Gram matrix  $(\mathcal{G}(\mathbf{a}, \mathbf{b}))_{ij} = \mathbf{a}_i^\top \mathbf{b}_j$  for two sets of vectors  $\mathbf{a}_i, i \in 1, \dots, N$ ,  $\mathbf{b}_j, j \in 1, \dots, M$ ,  $\mathcal{G}(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^{N \times M}$ , and trace  $\|\mathbf{A}\|_F = \sqrt{\sum_{ij} (A_{ij})^2}$ .

Gradient of scalar function is  $\left(\frac{\partial L(\mathbf{x})}{\partial \mathbf{x}}\right)_i = \frac{\partial L(\mathbf{x})}{\partial x_i}$  and Hessian of scalar function reads  $\left(\frac{\partial^2 L(\mathbf{x})}{\partial \mathbf{x}^2}\right)_{ij} = \frac{\partial^2 L(\mathbf{x})}{\partial x_i \partial x_j}$ ;  $\delta_{ij}$  refers to components of identity matrix,  $\exp(\mathbf{A}) = \sum_{k=0}^{\infty} \mathbf{A}^k / k!$  is a standard matrix exponent.

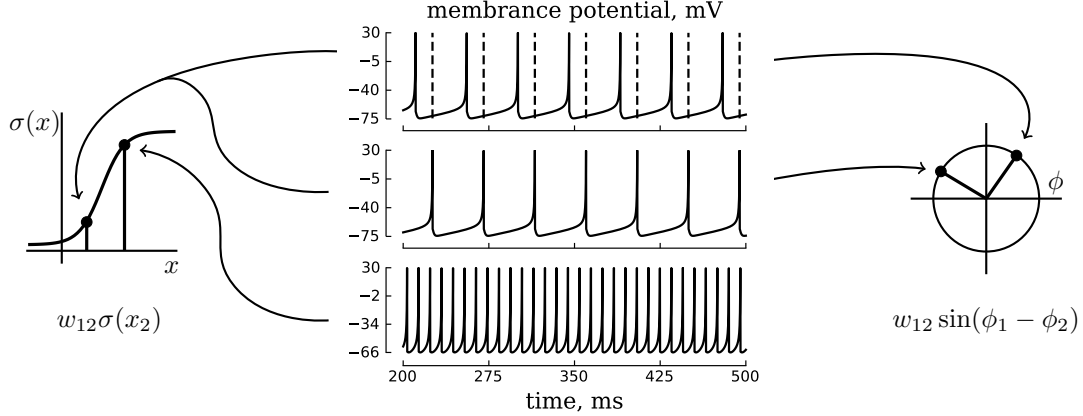


Figure 1: **In the middle:** membrane potential of three Izhikevich neurons under constant injected dc-current. Two first neurons fire at the same frequency but have different phases. The third neuron spikes at a higher frequency. **On the left:** threshold unit with smooth activation function  $\sigma(x)$ ; interaction term of additive model. Threshold unit is a simplified description of neuron’s interaction that only models time-averaged intensity of spikes  $x$ . Phase shift is ignored by the model. **On the right:** artificial oscillatory neuron with  $D = 1$ ; interaction term of Kuramoto model. Artificial oscillatory neuron is a simplified model that describes a neuron as a harmonic oscillator with fixed natural frequency. Interaction of artificial oscillatory neurons depends only on the phase difference. According to [27] oscillatory neurons with different non-resonant frequencies do not interact, and in this sense average intensity of spikes is ignored by the model. See Section 2 for discussion.

## 2 Threshold units and artificial oscillatory neurons

We start by describing models of threshold units and oscillatory neurons along with their biological interpretations.

The intricate and incompletely understood nature of biological neuron interactions has led to a diverse array of neuron models, each varying in its level of biological plausibility. One particularly convenient biologically realistic model was proposed by Izhikevich in [28]. Employing just two ordinary differential equations and an event-based reset condition, this model effectively captures a wide range of biologically relevant neuronal behaviors.. The membrane potential under constant injected DC current predicted by the Izhikevich model is available on the central panel of Figure 1. The dynamics is clearly non-smooth owing to the presence of a reset mechanism in the model. This inherent non-smoothness is a key challenge in training spiking neural networks<sup>2</sup> A common approach to circumvent this training challenge is to employ simplified models that abstract away certain biological complexities while offering more desirable numerical properties, such as smoother dynamics. The threshold unit is a prime example of such a model.

The McCulloch-Pitts artificial neuron, or threshold unit, represents the time-averaged activity of biological neurons, such as their average firing frequency, through a scalar state variable  $x$ . The interaction with other artificial neurons is

$$\dot{x}_i = \sum_j W_{ij} \sigma(x_j) - x_i, \quad (1)$$

where  $x_j$  are states of contributing neurons,  $W_{ij}$  synaptic weights,  $b_i$  is a bias term and  $\sigma$  is nonlinear activation function that models synaptic transmission of signal [24, Section 1.4.3]. Equation (1) represents a specific instance of an additive model, which has a direct connection to the Hopfield associative memory

<sup>2</sup>While specialized training methods exist [14], classical deep learning techniques often exhibit superior performance. This motivates the exploration of simplified, smoother models.

framework [15], [22]. The rationale behind the threshold unit is schematically captured in the left panel of Figure 1. In the context of deep learning, the continuous dynamics of the threshold unit are often approximated by a discrete update rule  $\sum_j W_{ij} \sigma(x_j)$ . While the temporal evolution differs, the fundamental interpretation in terms of weighted input and nonlinear activation remains largely consistent.

However, threshold units inherently fail to capture crucial aspects of neuronal interaction. As illustrated in Figure 1, they are insensitive to the relative timing, or phase difference, of neuronal spikes. This phase information is critical because even if neurons exhibit high average activity (large  $\sigma(x)$ ), effective communication can be hindered if presynaptic spikes arrive during the postsynaptic neuron’s refractory period. Such phase-dependent interactions, along with other effects related to frequency modulation, compel alternative simplified models. For instance, Izhikevich demonstrated that under certain conditions, the dynamics of periodically firing pulse-coupled neurons can be effectively described by the Kuramoto equation [27, Theorem 1, Section V.D, equation (17)]. The Kuramoto neuron’s state is represented by a phase  $\phi_i \in [0, 2\pi]$ . In the absence of coupling, each neuron oscillates with its intrinsic frequency  $\omega_i$  and their interaction is governed by the Kuramoto equation:

$$\dot{\phi}_i = \omega_i + \sum_j w_{ij} \sin(\phi_j - \phi_i) \quad (2)$$

Kuramoto neuron, an example of a more general oscillatory neuron, is graphically described in the right panel of Figure 1.

The artificial Kuramoto oscillatory neuron, recently introduced in [49], represents a natural extension of the standard Kuramoto model (Equation (2)). Its dynamics is based on the generalized Kuramoto equation proposed by Lohe [42]:

$$\dot{\boldsymbol{\mu}}_i = \boldsymbol{\Omega}_i \boldsymbol{\mu}_i + (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \sum_j w_{ij} \boldsymbol{\mu}_j, \quad (3)$$

where  $\boldsymbol{\mu}_i \in \mathbb{R}^{D+1}$ ,  $\boldsymbol{\Omega}_i$  is a skew-symmetric matrix. It can be readily observed that  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_i$  remains constant, and given that  $\boldsymbol{\Omega}_i$  is skew symmetric (describe rotation),  $\boldsymbol{\mu}_i$  evolves on the surface of the sphere. Furthermore, with initial conditions satisfying  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_i = 1$  the generalized Kuramoto equation (Equation (3)) simplifies to the standard Kuramoto model (Equation (2)) when  $D = 1$ .

Thus, the generalized Kuramoto model (Equation (3)) encompasses the standard Kuramoto neuron (Equation (2)) as a specific case when  $D = 1$ . Even when  $D > 1$  the generalized Kuramoto model retains biological plausibility. Artificial neurons are often interpreted as representing local populations of biological neurons rather than individual cells [24, Section 1.4.2]. While this population averaging is inherent in additive models without altering their form, for oscillatory models, it becomes plausible to represent a neuronal population with an oscillator possessing multiple intrinsic frequencies. In the generalized Kuramoto equation, these frequencies are linked to the eigenvalues of the skew-symmetric matrix  $\boldsymbol{\Omega}_i$ .<sup>3</sup>

As we have seen, both threshold units and artificial oscillatory neurons find justification from biological principles. Our analysis reveals that they correspond to distinct aspects of neural coding: threshold units primarily model averaged neuronal activity, disregarding temporal phase, while oscillatory neurons specifically capture interactions arising from the phase difference between neural oscillations. The co-existence of both threshold units and oscillatory neurons within a single network holds the promise of capturing a richer repertoire of neural dynamics than either model can achieve in isolation. In the subsequent two sections, we will leverage the general formalism of associative memory to introduce such a coupled model, bridging these distinct modes of neural interaction.

---

<sup>3</sup>Recall, that skew-symmetric matrix is unitary equivalent to block diagonal matrix with either  $1 \times 1$  zero block or  $2 \times 2$  block  $\begin{pmatrix} 0 & -\lambda_i \\ \lambda_i & 0 \end{pmatrix}$ . The later one describes rotation with frequency  $\lambda$ .

### 3 Two memory models

A canonical way to model associative memory was formulated in [22]. One constructs autonomous dynamical system

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}) \quad (4)$$

with Lyapunov function  $E$  non-increasing on trajectories of dynamical system

$$\dot{E}(\mathbf{x}) = \left( \frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} \right)^\top \dot{\mathbf{x}} = \left( \frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} \right)^\top \mathbf{F}(\mathbf{x}) \leq 0. \quad (5)$$

For sufficiently good energy functions, condition (5) ensures that trajectories end up at particular steady state  $\mathbf{x}^* : \mathbf{F}(\mathbf{x}^*) = 0$ . Each steady state has basin of attraction, the set of initial conditions that converge to  $\mathbf{x}^*$ ,  $\mathcal{D}_{\mathbf{x}^*} = \{\mathbf{y} : \mathbf{x}(0) = \mathbf{y}, \mathbf{x}(\infty) = \mathbf{x}^*, \dot{\mathbf{x}} = \mathbf{F}(\mathbf{x})\}$ .

Dynamical system (4) with these properties has a natural interpretation: steady states  $\mathbf{x}^*$  are stored memories and their number is the memory capacity; attractor  $\mathcal{D}_{\mathbf{x}^*}$  of steady state  $\mathbf{x}^*$  is a set of all stimuli that lead to the recall of  $\mathbf{x}^*$ .

Associative memory can also be understood as a productive computation platform. It was applied to many practical machine learning tasks including denoising [53], [20], clustering [45], [29], [52], classification [59], [51], representation learning [19], [20].

#### 3.1 Hopfield associative memory

We consider the continuous Hopfield memory model introduced in [35]. More precisely, we use the dense Hopfield layer from [34]. Both constructions are based on the Lagrange function  $L(\mathbf{x})$  that is a scalar function of a state variable whose gradient equals activation function. The dynamical system of the model reads

$$\dot{\mathbf{x}} = \mathbf{W} \frac{\partial L(\mathbf{x})}{\partial \mathbf{x}} - \mathbf{x} + \mathbf{b}. \quad (6)$$

In [35], [34] authors provide the following formal characteristic of this system.

**Theorem 3.1** (Lyapunov function for Hopfield-Krotov associative memory). *Let for dynamical system (6) weight matrix be symmetric  $\mathbf{W}^\top = \mathbf{W}$ , and Hessian of Lagrange function be positive-semidefinite matrix  $\frac{\partial^2 L}{\partial \mathbf{x}^2} \geq 0$ ,  $\mathbf{g}(\mathbf{x}) \equiv \frac{\partial L(\mathbf{x})}{\partial \mathbf{x}}$ . Energy function*

$$E_H(\mathbf{x}) = (\mathbf{x} - \mathbf{b})^\top \mathbf{g}(\mathbf{x}) - L(\mathbf{x}) - \frac{1}{2} \mathbf{g}(\mathbf{x})^\top \mathbf{W} \mathbf{g}(\mathbf{x}) \quad (7)$$

*is a Lyapunov function of (6) since*

$$\dot{E}_H(\mathbf{x}) = -\dot{\mathbf{x}}^\top \frac{\partial^2 L(\mathbf{x})}{\partial \mathbf{x}^2} \dot{\mathbf{x}} \leq 0. \quad (8)$$

*Proof.* Proof is available in [35], [34] and reproduced in Appendix A for convenience.  $\square$

So we see that under specified conditions on  $L$  and  $\mathbf{W}$ , dynamical system (6) is suitable for modelling of associative memory in a sense explained in the introduction to this section<sup>4</sup>.

The construction may look rather abstract, so we provide a concrete example of Lagrange function and weight matrix  $\mathbf{W}$ . The end of this example is twofold: (i) to illustrate the approach used to build deep neural network from shallow implicit model [4], [34]; (ii) to demonstrate that proper choice of Lagrange function may lead to neural networks with familiar activation functions.

---

<sup>4</sup>The symmetry of weights is undesirable from biological perspective, but not from the approximation theory standpoint: universal approximation property for networks with symmetric weights is demonstrated in [26].

**Example 1** (deep ReLU network). *For simplicity we will consider a network with three layers. Generalisation on deeper networks is straightforward. We consider  $\mathbf{W}$ ,  $\mathbf{b}$ ,  $\mathbf{x}$  with block structure*

$$\mathbf{W} = \begin{pmatrix} 0 & \mathbf{W}_{12} & 0 \\ \mathbf{W}_{12}^\top & 0 & \mathbf{W}_{23} \\ 0 & \mathbf{W}_{23}^\top & 0 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{pmatrix}.$$

*The Lagrange function that we use has additive structure  $L(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^3 \mathbf{1}^\top (\text{ReLU}(\mathbf{x}_i))^2$ , where  $\mathbf{1}$  is a vector with all components equal one. For this choice of  $\mathbf{W}$  and Lagrange function, steady-state reads*

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{W}_{12} \text{ReLU}(\mathbf{x}_2) + \mathbf{b}_1, \mathbf{x}_3 = \mathbf{W}_{32} \text{ReLU}(\mathbf{x}_2) + \mathbf{b}_3; \\ \mathbf{x}_2 &= \mathbf{W}_{21} \text{ReLU}(\mathbf{x}_1) + \mathbf{W}_{23} \text{ReLU}(\mathbf{x}_3) + \mathbf{b}_2. \end{aligned}$$

*So we see that steady-state has a structure of neural network with three layers and symmetric feedback connections.*

Techniques demonstrated in this section were used to derive energy function for many interesting architectures including dense associative memory [35] modern Hopfield network [51], convolutional neural networks [34], [19], energy transformer [20], MLP-mixer [56], diffusion models [2], [50].

### 3.2 Kuramoto associative memory

There exist many models of associative memory based on synchronisation of oscillators, e.g., [27], [23], [46]. A novel model of associative memory presented here is based on a version of generalised Kuramoto model [42], artificial oscillatory neurons [49] and ideas from [35]. Our main motivation is to accommodate oscillatory neurons into the general framework of associative memory to later infer possible couplings between oscillatory neurons and threshold units from the general form of global energy function.

Equations of motion for  $N$  coupled oscillator reads

$$\dot{\boldsymbol{\mu}}_i = \boldsymbol{\Omega}_i \boldsymbol{\mu}_i - (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \frac{\partial E_K}{\partial \boldsymbol{\mu}_i}, \quad (9)$$

where  $E_K$  is a potential function analogous to Lagrange function in Hopfield memory (6).

Kuramoto model (9) also has Lyapunov function under certain technical conditions on  $E_K$  and  $\boldsymbol{\Omega}_i$  as explained below.

**Theorem 3.2** (Lyapunov function for Kuramoto associative memory). *Let for dynamical system (9) potential  $E_K$  depends on  $\boldsymbol{\mu}_i$  only through scalar products  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$  and natural frequencies are identical  $\boldsymbol{\Omega}_i = \boldsymbol{\Omega}$  for all oscillators. Energy function  $E_K$  is a Lyapunov function since*

$$\dot{E}_K = - \sum_i \frac{\partial E_K}{\partial \boldsymbol{\mu}_i} (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \frac{\partial E_K}{\partial \boldsymbol{\mu}_i} \leq 0. \quad (10)$$

*Proof.* Appendix B. □

As we show in Appendix B, changing to rotating frame one can reduce Kuramoto model (9) to constrained gradient flow, under conditions of Theorem 3.2. Given that Kuramoto associative memory 9 can be used the same way as Hopfield model but with a different information encoding scheme.

The starting points on the energy landscape as well as its local minima depend only on scalar products  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$ . Given that, only scalar products  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$  should be used to recover memory vectors and also to supply input stimulus. A natural question then is: if one wants to store components of vector  $v \in [-1, 1]^K$  in scalar products, what is the minimal number of oscillators needed? For  $N$  constrained oscillators one has  $N(N-1)/2$  scalar products. Unfortunately, these scalar products are not independent. It is easy to see that the worst case is realised when one needs to encode a vector  $v$  with all components equal 1, in which case all

oscillators point to the same direction. So, in total, with  $N$  oscillators on a sphere we can reliably encode  $N - 1$  scalars. This is explained in detail in Appendix C.

A special structure of energy  $E_K$  and Kuramoto model (9) makes it unnatural to consider hierarchical models, but Lagrange formalism from [35] allows one to construct various nonlinear coupling in dense networks with oscillatory units. Below we provide two examples with ReLU and attention activation functions.

**Example 2** (dense ReLU network). *To define dense ReLU network we consider the following energy and coupling term*

$$E_K = \sum_{i=1}^{N-1} \sum_{j>i} \left( \frac{1}{2} W_{ij} (\text{ReLU}(\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j))^2 + b_{ij} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j \right),$$

$$\frac{\partial E_K}{\partial \boldsymbol{\mu}_i} = \sum_{j \neq i} (W_{ij} \text{ReLU}(\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j) + b_{ij}) \boldsymbol{\mu}_j.$$

*The form of a coupling is clearly related to the Hopfield network with ReLU activation functions. The main difference is that for Hopfield network degrees of freedom  $\mathbf{x}$  can be processed by nonlinear function directly, and in the Kuramoto memory model this can be done only with scalar products.*

**Example 3** (dense attention network). *Similarly one can construct attentive interaction*

$$E_K = \log \left( \sum_{i=1}^{N-1} \sum_{j>i} \exp(W_{ij} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j) \right) + \sum_{i=1}^{N-1} \sum_{j>i} b_{ij} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$$

$$\frac{\partial E_K}{\partial \boldsymbol{\mu}_i} = \sum_{j \neq i} \left( W_{ij} \frac{\exp(W_{ij} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j)}{\sum_{i=1}^{N-1} \sum_{j>i} \exp(W_{ij} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j)} + b_{ij} \right) \boldsymbol{\mu}_j.$$

*This model is analogous to Modern Hopfield Network defined in [51].*

The result we used here relies heavily on the assumption  $\boldsymbol{\Omega}_i = \boldsymbol{\Omega}$ . This means all natural frequencies are the same so frequency synchronisation is trivial. When frequencies are different but do not deviate too much from the mean frequency, the generalised Kuramoto model still converges to synchronisation state [47]. When frequency terms differ significantly, synchronisation is not possible. This provides an interesting opportunity to drive the system away from local energy minima without remodelling of potential, countering the argument against attractor-based formalism presented in [33].

The assumption  $\boldsymbol{\Omega}_i = \boldsymbol{\Omega}$  for  $D \gg 1$  is far less restrictive than  $\omega_i = \omega$  in classical Kuramoto model (2), especially when learning dynamics is considered. When  $D \gg 1$  we effectively allow each neuron to choose from a large number of potentially different frequencies. Given that, we expect the Kuramoto associative memory model to be more expressive when  $D$  increases. This is partially confirmed in Section 6.

## 4 Hopfield-Kuramoto associative memory

To define a joint Hopfield-Kuramoto memory model we consider  $N$  neurons that have  $D$  oscillatory and 1 scalar degrees of freedom. The equations of motion are those of Hopfield (6) and Kuramoto (9) models but with additional interaction terms

$$\dot{x}_i = \sum_j W_{ij} \left( \frac{\partial L(\mathbf{x})}{\partial \mathbf{x}} \right)_j - x_i + b_i + \frac{1}{\kappa_H} \Delta x_i(\mathbf{x}, \boldsymbol{\mu}),$$

$$\dot{\boldsymbol{\mu}}_i = \boldsymbol{\Omega}_i \boldsymbol{\mu}_i + (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \left( -\frac{\partial E_K}{\partial \boldsymbol{\mu}_i} + \frac{1}{\kappa_K} \delta \boldsymbol{\mu}_i(\mathbf{x}, \boldsymbol{\mu}) \right),$$
(11)

where we rewrite Hopfield model (6) in its scalar form to align it better with Kuramoto model, and introduce coupling constants  $\kappa_H, \kappa_K > 0$  that control interaction strength.

The main difficulty is to introduce coupling terms that still allow us to define Lyapunov function. An appropriate strategy is to modify existing Hopfield and Kuramoto energy functions to introduce interaction. Kuramoto energy function should depend only on scalar products  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$  and this is the only constraint. On the other hand, the Hopfield energy function (7) has a fairly restricted form. Part where scalar products  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$  fit naturally is the term with synaptic weights  $W_{ij}$ . This suggest us to replace the last term in Hopfield energy (7) with

$$-\sum_{ij} W_{ij} (\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j) \left( \frac{\partial L(\mathbf{x})}{\partial \mathbf{x}} \right)_i \left( \frac{\partial L(\mathbf{x})}{\partial \mathbf{x}} \right)_j,$$

which corresponds to the multiplication of synaptic weights on scalar products  $(\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j)$ . Clearly, these manipulations satisfy constraints of Kuramoto energy since only scalar products are involved. Besides that they also lead to valid Hopfield energy, since weights remain symmetric and other parts of energy function (7) are not modified.

Following similar lines we derived more general interaction terms with the Lyapunov function presented below.

**Theorem 4.1** (Lyapunov function for Hopfield-Kuramoto associative memory). *Let  $\mathbf{g}(\mathbf{x}) \equiv \frac{\partial L(\mathbf{x})}{\partial \mathbf{x}}$ ,  $G_{ij} = G_{ji}$  and  $\chi_{ij} = \chi_{ji}$  be smooth scalar function and Hessian of Lagrange function be positive-semidefinite matrix  $\frac{\partial^2 L}{\partial \mathbf{x}^2} \geq 0$  and  $\boldsymbol{\Omega}_i = \boldsymbol{\Omega}$ ,  $\kappa_H$  and  $\kappa_K$  are positive coupling constants. Energy function*

$$E_{HK}(\mathbf{x}, \boldsymbol{\mu}) = \kappa_H E_H(\mathbf{x}) + \kappa_K E_K(\boldsymbol{\mu}) - \frac{1}{2} \sum_{ij} G_{ij} (g(\mathbf{x})_i g(\mathbf{x})_j) \chi_{ij} (\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j) \quad (12)$$

is Lyapunov function for joint Kuramoto-Hopfield model (11) with coupling terms

$$\begin{aligned} \Delta x_i(\mathbf{x}, \boldsymbol{\mu}) &= \sum_j G'_{ij} (g(\mathbf{x})_i g(\mathbf{x})_j) \chi_{ij} (\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j) g(\mathbf{x})_j, \\ \delta \boldsymbol{\mu}_i(\mathbf{x}, \boldsymbol{\mu}) &= \sum_{j \neq i} G_{ij} (g(\mathbf{x})_i g(\mathbf{x})_j) \chi'_{ij} (\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j) \boldsymbol{\mu}_j, \end{aligned} \quad (13)$$

where  $G'_{ij}$  and  $\chi'_{ij}$  are derivatives of  $G_{ij}$  and  $\chi_{ij}$ . More precisely, under conditions specified

$$\dot{E}_{HK}(\mathbf{x}, \boldsymbol{\mu}) = -\kappa_H \dot{\mathbf{x}}^\top \frac{\partial^2 L(\mathbf{x})}{\partial \mathbf{x}^2} \dot{\mathbf{x}} - \frac{1}{\kappa_K} \sum_i \frac{\partial E_{HK}}{\partial \boldsymbol{\mu}_i} (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \frac{\partial E_{HK}}{\partial \boldsymbol{\mu}_i} \leq 0. \quad (14)$$

*Proof.* Appendix D. □

Coupling (13) allows for a rich set of models with various control of threshold units by oscillatory units and vice versa. Using formalism of Lagrange functions from [35] it is possible to introduce Hopfield-Kuramoto architectures with convolution layers, attention mechanism, normalisation, pooling, etc. Several concrete architectures are available in Appendix E. Below we build a quantitative example that shows how oscillatory degrees of freedom can realise multiplexing.

**Example 4** (multiplexing with oscillatory units). *We will demonstrate that in the joint Hopfield-Kuramoto memory model the same set of threshold units can be reused in two different computations. Which computation is invoked is determined by oscillatory units. Let  $G_1$  and  $G_2$  be two sets of indices  $G_1 \cap G_2 = \emptyset$ . Neurons from  $G_2$  interact with neurons from  $G_1$  with synaptic weights  $W_{ij}, i \in G_1, j \in G_2$ . We select coupling constants  $\kappa_K \gg 1$  and  $\kappa_H = 1$ . With this choice dynamics for oscillatory units reduced to uncoupled Kuramoto memory (9), but threshold units remain coupled to oscillatory units with  $\Delta x_i$ .*

Suppose we select energy  $E_K$  in such a way to for input  $\boldsymbol{\mu}_i(0) = \boldsymbol{\mu}_i^{(A)}$  oscillatory neurons from  $G_1$  and  $G_2$  converges to aligned state  $\boldsymbol{\mu}_i^\top(\infty) \boldsymbol{\mu}_j(\infty) = 1$  and for input  $\boldsymbol{\mu}_i(0) = \boldsymbol{\mu}_i^{(B)}$  they converge to the orthogonal state  $\boldsymbol{\mu}_i^\top(\infty) \boldsymbol{\mu}_j(\infty) = 0$ . Recall that  $E_K$  is not constrained, so as long as inputs  $A$  and  $B$  are sufficiently distinct it is possible to construct such energy function.



Choosing  $\Delta x_i = \sum_j (\widetilde{W}_{ij} - W_{ij}) \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j g(\mathbf{x})_i$  as correction term we observe that for input  $A$  the effect of correction term is the change of weights from  $W_{ij}$  to  $\widetilde{W}_{ij}$  for interaction of neuron in  $G_1$  and  $G_2$ . This means selected threshold units perform two different functions based on the state of oscillatory neurons. In a similar way multiplexing with more than two functions can be achieved with additional interaction terms.

Multiplexing example suggests that oscillatory neurons can remodel energy landscape of Hopfield associative memory during recall. This means memories stored in the Hopfield-Kuramoto network explicitly depend on the input stimulus. Formally, Hopfield model implement mapping  $\mathbf{x}(0) \rightarrow \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ , where  $M$  is memory capacity, i.e., it selects among  $M$  stored memories. If we suppose for simplicity that initial state of oscillatory units is a function of  $\mathbf{x}(0)$ , Hopfield-Kuramoto model realises mapping  $\mathbf{x}(0) \rightarrow \{\mathbf{x}_1(\mathbf{x}(0)), \dots, \mathbf{x}_{M(\mathbf{x}(0))}(\mathbf{x}(0))\}$  where both capacity and stored memories are functions of  $\mathbf{x}(0)$ . We will see numerical example of memory edit in Section 6.

## 5 Interpretations of binding mechanism

While the general form of the coupling mechanism (Equation (13)) offers flexibility, its interpretation can be nuanced. To gain clearer insights, and building upon Example 4, we now analyze specific choices for the interaction terms and explore their connections to established mechanisms in standard deep learning architectures.

Let us first consider a specific choice of interaction functions:  $G_{ij}(a) = \widetilde{W}_{ij}a$ ,  $\chi_{ij}(a) = a$ ,  $W_{ij} = 0$ ,  $E_K = 0$ ,  $\kappa_H = \kappa_K = 1$ . This particular selection leads to the following Hopfield-Kuramoto model:

$$\begin{aligned} \dot{x}_i &= \sum_j \left( \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j \widetilde{W}_{ij} \right) g(\mathbf{x})_j - x_i + b_i, \\ \dot{\boldsymbol{\mu}}_i &= \boldsymbol{\Omega}_i \boldsymbol{\mu}_i + (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \sum_j \left( g(\mathbf{x})_i g(\mathbf{x})_j \widetilde{W}_{ij} \right) \boldsymbol{\mu}_j. \end{aligned} \quad (15)$$

Observing the equation for the threshold units, we find that the term  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$  acts as a neuron-specific gating mechanism on the influence of the threshold units  $i$  and  $j$ . This bears resemblance to the gating mechanisms employed in LSTM and GRU networks [18], [9]. Specifically, when the inner product  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$  is zero, the interaction between threshold units  $i$  and  $j$  is completely suppressed, even if their individual activities  $g(\mathbf{x})_i$ ,  $g(\mathbf{x})_j$  are not zero. Biologically, this suppression could be analogous to a scenario where an action potential fails to evoke a response due to the postsynaptic neuron being in its refractory phase. Even if the average activity of connected neurons is high ( $g(\mathbf{x}) \neq 0$ ), effective interaction might be blocked if a presynaptic action potential arrives during the postsynaptic neuron's refractory period, a state where the neuron is temporarily less responsive. Conversely, the synchronization of oscillations between two neurons is contingent on their corresponding threshold unit activities being non-zero ( $g(\mathbf{x})_i \neq 0$ ,  $g(\mathbf{x})_j \neq 0$ ), suggesting that the threshold units modulate the oscillatory interactions.

Now, let us consider another choice for the interaction terms:  $G_{ij}(a) = a$ ,  $\chi_{ij}(a) = a$ ,  $E_K = -\sum_{i>j} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j R_{ij}$ ,  $\kappa_H = \kappa_K = 1$ . This selection yields the following coupled dynamics:

$$\begin{aligned} \dot{x}_i &= \sum_j (W_{ij} + \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j) g(\mathbf{x})_j - x_i + b_i, \\ \dot{\boldsymbol{\mu}}_i &= \boldsymbol{\Omega}_i \boldsymbol{\mu}_i + (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \sum_j (R_{ij} + g(\mathbf{x})_i g(\mathbf{x})_j) \boldsymbol{\mu}_j. \end{aligned} \quad (16)$$

Here, we observe that the weight matrix  $\mathbf{R}$  governing the interaction between oscillatory neurons undergoes a rank-1 update based on the outer product of the threshold unit activities:  $\mathbf{R} \leftarrow \mathbf{R} + g(\mathbf{x})g(\mathbf{x})^\top$ . Similarly, the weight matrix of threshold units subjects to a rank- $m$  correction, where  $m \leq D+1$ , determined by the dimensionality of the oscillatory neuron state  $\boldsymbol{\mu}_i \in \mathbb{R}^{D+1}$ . The rank- $m$  correction arises from the term  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$ . Defining a matrix  $\mathbf{G}$  with entries  $G_{ij} \equiv \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$  we recognise  $\mathbf{G}$  as a Gram matrix of vectors  $\boldsymbol{\mu}_i$ . Since  $\mathbf{G}$  is a

Gram matrix formed from  $N$  vectors in  $\mathbb{R}^{D+1}$ , it is positive semi-definite with rank at most  $D + 1$  (assuming  $N > D$ ). This property implies the existence of a matrix  $\mathbf{A}(\boldsymbol{\mu}) \in \mathbb{R}^{N \times k}$  such that  $\mathbf{G}$  can be factorised as  $\mathbf{G} = \mathbf{A}(\boldsymbol{\mu})\mathbf{A}(\boldsymbol{\mu})^\top$ .<sup>5</sup> Thus, this specific form of coupling leads to two low-rank update rules for the weight matrices:

$$\mathbf{R} \leftarrow \mathbf{R} + \mathbf{g}(\mathbf{x})\mathbf{g}(\mathbf{x})^\top, \mathbf{W} \leftarrow \mathbf{W} + \mathbf{A}(\boldsymbol{\mu})\mathbf{A}(\boldsymbol{\mu})^\top. \quad (17)$$

Interestingly, similar low-rank weight updates are employed in deep learning, notably in the Low-Rank Adaptation (LoRA) method [25], a computationally efficient technique for fine-tuning large language models. A key distinction from standard LoRA is that the factors in our low-rank corrections (Equation (17)) are functions of the network’s state  $(\boldsymbol{\mu}_i, x_i)$ , i.e., time-dependent. This bears similarity to the concept of “fast weights,” which were also proposed as low-rank, time-dependent modifications to weight matrices [3], [54].

From a biological perspective, the weight updates in Equation (17) can be interpreted as a form of Hebbian learning, a principle widely used for pattern storage in Hopfield networks [21], [1, Definition 2]. In this context, the oscillatory degrees of freedom can potentially contribute up to  $D + 1$  new patterns to those encoded in the threshold unit weight matrix  $\mathbf{W}$ , while the threshold unit activities can, in turn, influence at most one pattern stored in the oscillatory unit weight matrix  $\mathbf{R}$ . Given the time-dependent nature of the correction terms in Equation (17), we anticipate a significant influence on the basins of attraction of the network’s dynamics. However, as  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j$  and  $\mathbf{x}$  are expected to converge to steady-state values, the effective rank of these corrections at equilibrium is expected to be at most  $D + 1$  and 1 respectively. Consistent with the multiplexing phenomenon illustrated in Example 4, these weight corrections exhibit an explicit dependence on the initial conditions of the network.

To empirically validate the implications of this low-rank coupling mechanism (Equation (17)) and the interpretations discussed, we will present results from simple learning problems in Section 6.

## 6 Numerical illustration of coupling mechanism

To illustrate the effect of coupling numerically we select the following model

$$\begin{aligned} \dot{x}_i &= \sum_j (W_{ij} + \kappa \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j) \text{ReLU}(\mathbf{x})_j - x_i + b_i, \\ \dot{\boldsymbol{\mu}}_i &= (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \sum_j (R_{ij} + \kappa \text{ReLU}(\mathbf{x})_i \text{ReLU}(\mathbf{x})_j) \boldsymbol{\mu}_j \\ &\quad + (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \sum_j S_{ij} \text{ReLU}(\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j) \boldsymbol{\mu}_j + \boldsymbol{\Omega} \boldsymbol{\mu}_i, \end{aligned} \quad (18)$$

where both  $\mathbf{S}$  and  $\mathbf{R}$  are sparse matrices with sparsity pattern taken from adjacency matrix of Watts-Strogatz small world network with constant probability of edge rewriting  $p = 0.1$  and variable number of neighbours  $k$  [58]. Sparse matrices were chosen to reduce memory consumption and a small world network was selected because of its biological plausibility. Model (18) is a particular version of low-rank coupling (15) with slightly altered  $E_K$  intended to increase expressivity of subnetwork with oscillatory neurons.

To study coupling between networks we design a special experiment protocol, illustrated in Figure 2. First, we take  $\kappa = 0$  and train Hopfield subnetwork to recognise MNIST digits [39]. After this stage we have trained weights  $\mathbf{W}$  and bias term  $\mathbf{b}$  such that starting from MNIST digit  $\mathbf{x}(0)$ , last 10 elements of  $\mathbf{x}(1)$  converge to class labels encoded as +1 for correct class and -1 for incorrect class. Training is done with backpropagation through time and overall the approach closely follows established techniques for learning with associative memory models [36], [19]. The accuracy of Hopfield network is reported in Table 1.

After the pre-training stage we modify MNIST labels and train a combined network with learnable  $\kappa$  and frozen parameters of Hopfield subnetwork. In this fine-tuning stage the only trainable parameters are  $\kappa$ , and weights of subnetwork with oscillatory units  $\mathbf{R}, \mathbf{S}, \boldsymbol{\Omega}$ . The fine-tuning problem is challenging because information exchange between subnetworks comes from a very restrictive coupling term that has a single

<sup>5</sup>An explicit way to find  $\mathbf{A}$  is to use Cholesky decomposition.

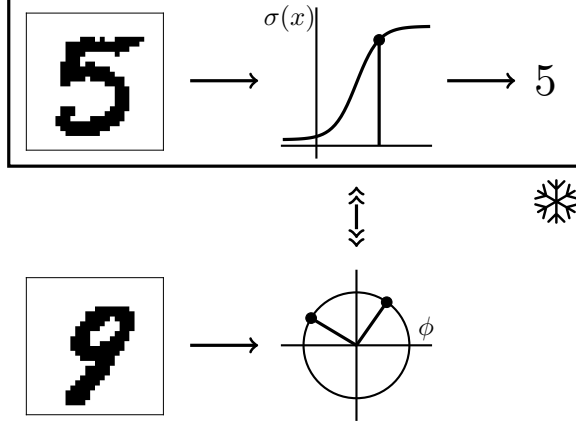


Figure 2: Empirical validation of interaction between threshold and oscillatory units. **On the first stage** Hopfield network with threshold units is pre-trained to recognise MNIST digits and all its parameters are frozen. **On the second stage** oscillatory units are added to the network and the combined network is trained on the modified task. For example, if 1 is present to oscillatory neurons, Hopfield network should swap labels for 0 and 1; if additional input is 0, the output of Hopfield network is not modified. Learnable parameters are interaction strength (scalar) and weights of oscillatory units.

learnable parameter  $\kappa$ . By design of model (18) fine-tuning can be successful only when oscillatory neurons learn to converge to an appropriate low-rank correction (17) that can adapt weights  $\mathbf{W}$  of Hopfield network to a changed task.

We consider four modification of MNIST classification problem:

- **Swap.** Labels 0 and 1 are exchanged. Initial values  $\mu_i(0)$  are learnable parameters.
- **Conflation.** Label 1 is changed to 0. Initial values  $\mu_i(0)$  are learnable parameters.
- **Associative swap.** For each MNIST sample digits we draw random 0 and 1 MNIST digits with equal probability. These additional samples are used to initialise oscillatory neurons  $\mu_i(0)$ . For a given sample, if a randomly selected digit (input to oscillatory units) is 0 MNIST label is not changed, if it is 1, labels 0 and 1 are exchanged.
- **Associative conflation.** Similar to associative swap, but label 1 is changed to label 0 in case additional input is digit 1.

In other words, we exchange labels 0 and 1 (swap) or change label 1 to 0 (conflation) either for all relevant MNIST samples or only for a fraction, conditioned on the additionally selected digit  $\{0, 1\}$  presented as an input (association) for oscillatory units. The accuracy of the pre-trained Hopfield network on four modified

Table 1: Performance of pre-trained Hopfield model in original MNIST dataset and its altered version: swap refers to change of label 0 to 1 and vice versa; conflation refers to change of label 1 to 0; associative version do the same but for fraction of randomly selected samples.

dataset	train accuracy	test accuracy
MNIST	98.2%	97.2%
conflation	87.1%	86.0%
swap	77.4%	76.3%
associative conflation	92.8%	91.7%
associative swap	88.0%	86.8%

Table 2: Results of fine-tuning for non-associative MNIST modifications.

$k$	$D$	conflation		swap	
		train acc.	test acc.	train acc.	test acc.
150	4	96.9%	96.0%	88.8%	88.2%
250	4	97.9%	97.0%	96.3%	95.3%
350	4	98.0%	97.1%	97.9%	96.8%
150	6	97.6%	96.6%	89.5%	88.8%
250	6	98.0%	97.1%	97.4%	96.5%
350	6	98.1%	97.2%	98.0%	97.0%

tasks is reported in Table 1. In general swap is more challenging than conflation and associative setting is harder than non-associative one. Ideally, we want fine-tuning to recover initial test accuracy of  $\simeq 97\%$ .

We train several coupled models with frozen parameters of Hopfield network and  $D = 2, D = 3$  and  $k = 150, 250, 350, 450$ . Recall that  $D$  is the number of oscillatory degrees of freedom for each neuron, and the larger  $k$  the more trainable parameters are available in sparse matrices  $\mathbf{R}, \mathbf{S}$ . In general models with larger  $D$  and  $k$  should be more capable. The results of fine-tuning are available in Table 2 and Table 3.

We can clearly see that accuracy is improved after fine-tuning. Moreover in the best cases initial 97% accuracy is recovered for all tasks but associative swap, where the highest obtained accuracy is 93.6%. For non-associative problems we find that  $D = 6$  uniformly leads to better results than  $D = 4$ . For associative problems we observe no difference between  $D = 4$  and  $D = 6$ . This is likely because the bottleneck in performance comes from the expressivity of oscillatory subnetwork (controlled by  $k$ ) because it faces more challenging tasks for this class of problems. In general we see that the results confirm that oscillatory neurons can be used to alter the content of Hopfield associative memory.

More details on numerical experiments discussed in this section are available in Appendix F.

## 7 Conclusion

In this research note, we have introduced a novel theoretically grounded coupling mechanism between artificial oscillatory neurons and traditional threshold units. This interaction term is justified based on derived joint Hopfield-Kuramoto associative memory model. Our analysis reveals that this coupling term admits several intriguing interpretations, including a form of low-rank weight correction, a multiplexing of information channels, and a dynamic gating mechanism influencing neuronal interactions. Furthermore, we have experimentally validated the capacity of oscillatory neurons to dynamically modify the stored memories within a Hopfield network through a set of carefully designed simple experiments.

These findings suggest a promising avenue for developing more sophisticated neural network architectures that leveraging the distinct computational strengths of models capturing averaged activity and oscillatory dynamics. Future work could explore the application of this coupling mechanism in deeper network architectures and investigate its potential benefits in addressing complex cognitive tasks where temporal dynamics

Table 3: Results of fine-tuning for associative MNIST modifications.

$k$	$D$	conflation		swap	
		train acc.	test acc.	train acc.	test acc.
150	4, 6	92.7%	91.5%	87.8%	86.6%
250	4, 6	94.0%	92.8%	88.8%	87.9%
350	4, 6	97.5%	96.4%	94.7%	93.5%
450	4, 6	97.9%	97.0%	94.8%	93.6%

and synchronization play a crucial role. Further theoretical analysis into the stability and capacity of the proposed Hopfield-Kuramoto model could also yield valuable insights.

## References

- [1] Linda Albanese, Adriano Barra, Pierluigi Bianco, Fabrizio Durante, and Diego Pallara. Hebbian learning from first principles. *Journal of Mathematical Physics*, 65(11), 2024.
- [2] Luca Ambrogioni. In search of dispersed memories: Generative diffusion models are associative memory networks. *arXiv preprint arXiv:2309.17290*, 2023.
- [3] Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 29, 2016.
- [4] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.
- [5] Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.
- [6] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.
- [7] Gyorgy Buzsaki and Andreas Draguhn. Neuronal oscillations in cortical networks. *science*, 304(5679):1926–1929, 2004.
- [8] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.
- [9] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
- [10] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- [11] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.
- [12] Florian Dörfler and Francesco Bullo. Synchronization in complex networks of phase oscillators: A survey. *Automatica*, 50(6):1539–1564, 2014.
- [13] Emrah Düzel, Will D Penny, and Neil Burgess. Brain oscillations and memory. *Current opinion in neurobiology*, 20(2):143–149, 2010.
- [14] Jason K Eshraghian, Max Ward, Emre O Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennisamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

- [15] Stephen Grossberg. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural networks*, 1(1):17–61, 1988.
- [16] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.
- [17] Drew B Headley and Denis Paré. Common oscillatory mechanisms across multiple memory systems. *npj Science of Learning*, 2(1):1, 2017.
- [18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [19] Benjamin Hoover, Duen Horng Chau, Hendrik Strobelt, and Dmitry Krotov. A universal abstraction for hierarchical hopfield networks. In *The Symbiosis of Deep Learning and Differential Equations II*, 2022.
- [20] Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed Zaki, and Dmitry Krotov. Energy transformer. *Advances in Neural Information Processing Systems*, 36, 2024.
- [21] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.
- [22] John J Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.
- [23] Frank C Hoppensteadt and Eugene M Izhikevich. Oscillatory neurocomputers with dynamic connectivity. *Physical Review Letters*, 82(14):2983, 1999.
- [24] Frank C Hoppensteadt and Eugene M Izhikevich. *Weakly connected neural networks*, volume 126. Springer Science & Business Media, 2012.
- [25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- [26] Shell Xu Hu, Sergey Zagoruyko, and Nikos Komodakis. Exploring weight symmetry in deep neural networks. *Computer Vision and Image Understanding*, 187:102786, 2019.
- [27] Eugene M Izhikevich. Weakly pulse-coupled oscillators, fm interactions, synchronization, and oscillatory associative memory. *IEEE Transactions on Neural Networks*, 10(3):508–526, 1999.
- [28] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.
- [29] Khalid Kandali, Lamyae Bennis, and Hamid Bennis. A new hybrid routing protocol using a modified k-means clustering algorithm and continuous hopfield network for vanet. *IEEE Access*, 9:47169–47183, 2021.
- [30] T Anderson Keller, Lyle Muller, Terrence J Sejnowski, and Max Welling. A spacetime perspective on dynamical computation in neural information processing systems. *arXiv preprint arXiv:2409.13669*, 2024.
- [31] Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.
- [32] Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

- [33] Daniel Koch, Akhilesh Nandan, Gayathri Ramesan, and Aneta Koseska. Biological computations: limitations of attractor-based formalisms and the need for transients. *Biochemical and Biophysical Research Communications*, 720:150069, 2024.
- [34] Dmitry Krotov. Hierarchical associative memory. *arXiv preprint arXiv:2107.06446*, 2021.
- [35] Dmitry Krotov and John Hopfield. Large associative memory problem in neurobiology and machine learning. *arXiv preprint arXiv:2008.06996*, 2020.
- [36] Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.
- [37] Yoshiki Kuramoto. *Chemical Oscillations, Waves, and Turbulence*. Springer, 1984.
- [38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [39] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2, 2010.
- [40] Luisa HB Liboni, Roberto C Budzinski, Alexandra N Busch, Sindy Löwe, Thomas A Keller, Max Welling, and Lyle E Muller. Image segmentation with traveling waves in an exactly solvable recurrent neural network. *Proceedings of the National Academy of Sciences*, 122(1):e2321319121, 2025.
- [41] Grace W Lindsay. Convolutional neural networks as a model of the visual system: Past, present, and future. *Journal of cognitive neuroscience*, 33(10):2017–2031, 2021.
- [42] MA2539317 Lohe. Non-Abelian Kuramoto models and synchronization. *Journal of Physics A: Mathematical and Theoretical*, 42(39):395101, 2009.
- [43] Sindy Löwe, Phillip Lippe, Francesco Locatello, and Max Welling. Rotating features for object discovery. *Advances in Neural Information Processing Systems*, 36, 2024.
- [44] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.
- [45] Stefan R Maetschke and Mark A Ragan. Characterizing cancer subtypes as attractors of hopfield networks. *Bioinformatics*, 30(9):1273–1279, 2014.
- [46] Paolo Maffezzoni, Bichoy Bahr, Zheng Zhang, and Luca Daniel. Oscillator array models for associative memory and pattern recognition. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(6):1591–1598, 2015.
- [47] Johan Markdahl, Daniele Proverbio, La Mi, and Jorge Goncalves. Almost global convergence to practical synchronization in the generalized kuramoto model on networks over the n-sphere. *Communications Physics*, 4(1):187, 2021.
- [48] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.
- [49] Takeru Miyato, Sindy Löwe, Andreas Geiger, and Max Welling. Artificial Kuramoto Oscillatory Neurons. *arXiv preprint arXiv:2410.13821*, 2024.
- [50] Bao Pham, Gabriel Raya, Matteo Negri, Mohammed J Zaki, Luca Ambrogioni, and Dmitry Krotov. Memorization to generalization: The emergence of diffusion models from associative memory. 2024.
- [51] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

- [52] Bishwajit Saha, Dmitry Krotov, Mohammed J Zaki, and Parikshit Ram. End-to-end differentiable clustering with associative memories. In *International Conference on Machine Learning*, pages 29649–29670. PMLR, 2023.
- [53] Tommaso Salvatori, Yuhang Song, Yujian Hong, Lei Sha, Simon Frieder, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, 34:3874–3886, 2021.
- [54] Jürgen Schmidhuber. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In *ICANN’93: Proceedings of the International Conference on Artificial Neural Networks Amsterdam, The Netherlands 13–16 September 1993 3*, pages 460–463. Springer, 1993.
- [55] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [56] Fei Tang and Michael Kopp. A remark on a paper of krotov and hopfield [arxiv: 2008.06996]. *arXiv preprint arXiv:2105.15034*, 2021.
- [57] Sander van Bree, Daniel Levenstein, Matthew R Krause, Bradley Voytek, and Richard Gao. Processes and measurements: a framework for understanding neural oscillations in field potentials. *Trends in Cognitive Sciences*, 2025.
- [58] Duncan J Watts and Steven H Strogatz. Collective dynamics of ‘small-world’ networks. *nature*, 393(6684):440–442, 1998.
- [59] Michael Widrich, Bernhard Schäfl, Milena Pavlović, Hubert Ramsauer, Lukas Gruber, Markus Holzleitner, Johannes Brandstetter, Geir Kjetil Sandve, Victor Greiff, Sepp Hochreiter, et al. Modern hopfield networks and attention for immune repertoire classification. *Advances in neural information processing systems*, 33:18832–18845, 2020.
- [60] Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural networks*, 94:103–114, 2017.



## A Lyapunov function for Hopfield associative memory

As we mentioned in the main text the proof of this statement is available in [35], [34]. Nonetheless we find it appropriate to reproduce this proof here because in Appendix D we will need it for a more general result on the joint Hopfield-Kuramoto memory model.

We define  $\mathbf{g}(\mathbf{x}) = \frac{\partial L(\mathbf{x})}{\partial \mathbf{x}}$ ,  $\mathbf{\Lambda} = \frac{\partial^2 L}{\partial \mathbf{x}^2}$  and rewrite dynamical system (6) and energy function (7) in more explicit form

$$\begin{aligned}\dot{x}_i &= \sum_j W_{ij} g(\mathbf{x})_j - x_j - b_j, \\ E_H(\mathbf{x}) &= \sum_i (x_i - b_i) g(\mathbf{x})_i - L(\mathbf{x}) - \frac{1}{2} \sum_{ij} g(\mathbf{x})_i W_{ij} g(\mathbf{x})_j.\end{aligned}\tag{19}$$

Derivative of energy function reads

$$\dot{E}_H(\mathbf{x}) = \sum_i \frac{\partial E_H(\mathbf{x})}{\partial x_i} \dot{x}_i,\tag{20}$$

so we only need to find partial derivatives with respect to  $x_i$

$$\frac{\partial E_H(\mathbf{x})}{\partial x_i} = g(\mathbf{x})_i + \sum_j (x_j - b_j) \Lambda_{ij} - g(\mathbf{x})_i - \sum_{jk} \Lambda_{ij} W_{jk} g(\mathbf{x})_k = - \sum_j \Lambda_{ij} \left( \sum_k W_{jk} g(\mathbf{x})_k - x_j + b_j \right) = - \sum_j \Lambda_{ij} \dot{x}_j.\tag{21}$$

After that we find

$$\dot{E}_H(\mathbf{x}) = - \sum_{ij} \Lambda_{ij} \dot{x}_i \dot{x}_j = - \dot{\mathbf{x}}^\top \mathbf{\Lambda} \dot{\mathbf{x}} = - \dot{\mathbf{x}}^\top \frac{\partial^2 L}{\partial \mathbf{x}^2} \dot{\mathbf{x}}.\tag{22}$$

By assumption, the Hessian of the Lagrange function is a positive semidefinite matrix, so  $\dot{E}_H(\mathbf{x}) \leq 0$ .

## B Lyapunov function for Kuramoto associative memory

We provide two distinct proofs.

The first one is based on the change of variables that remove rotation from the equation. It is somewhat longer but illustrates why it is important to have an energy function that depends on scalar products only. Besides, there we show how the Kuramoto model can be reduced to the gradient flow on the sphere.

In the second proof we proceed directly. Here one can see that the special form of energy ensure that  $\mathbf{\Omega}_i$  vanish if  $\mathbf{\Omega}_i = \mathbf{\Omega}$

### B.1 Proof by change of variables

Matrix  $\exp(\mathbf{\Omega}_i t)$  is orthogonal

$$(\exp(\mathbf{\Omega}_i t))^\top = \exp(\mathbf{\Omega}_i^\top t) = \exp(-\mathbf{\Omega}_i t) = (\exp(\mathbf{\Omega}_i t))^{-1}.\tag{23}$$

We use this orthogonal transformation to switch to the rotating frame

$$\boldsymbol{\mu}_i = \exp(\mathbf{\Omega}_i t) \boldsymbol{\nu}_i, \quad \boldsymbol{\nu}_i = \exp(-\mathbf{\Omega}_i t) \boldsymbol{\mu}_i.\tag{24}$$

Observe that when  $\mathbf{\Omega}_i = \mathbf{\Omega}$ , scalar products remain the same

$$\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j = \boldsymbol{\nu}_i^\top \exp(-\mathbf{\Omega} t) \exp(\mathbf{\Omega} t) \boldsymbol{\nu}_j = \boldsymbol{\nu}_i^\top \boldsymbol{\nu}_j,\tag{25}$$

which implies  $E_K(\boldsymbol{\mu}_i) = E_K(\boldsymbol{\nu}_i)$  when  $\mathbf{\Omega}_i = \mathbf{\Omega}$  since by assumption it depends only on the scalar products.

To rewrite equations of motion in new variables we observe that

$$\frac{\partial E_K}{\partial (\boldsymbol{\mu}_i)_\alpha} = \sum_\beta \frac{\partial E_K}{\partial (\boldsymbol{\nu}_i)_\beta} \frac{\partial (\boldsymbol{\nu}_i)_\beta}{\partial (\boldsymbol{\mu}_i)_\alpha} = \sum_\beta \frac{\partial E_K}{\partial (\boldsymbol{\nu}_i)_\beta} (\exp(-\boldsymbol{\Omega}_i t))_{\beta\alpha} = \sum_\beta \frac{\partial E_K}{\partial (\boldsymbol{\nu}_i)_\beta} (\exp(\boldsymbol{\Omega}_i t))_{\alpha\beta}, \quad (26)$$

or in matrix form

$$\frac{\partial E_K}{\partial \boldsymbol{\mu}_i} = \exp(\boldsymbol{\Omega}_i t) \frac{\partial E_K}{\partial \boldsymbol{\nu}_i}. \quad (27)$$

For time derivative we find

$$\dot{\boldsymbol{\mu}}_i = \boldsymbol{\Omega}_i \exp(\boldsymbol{\Omega}_i t) \boldsymbol{\nu}_i + \exp(\boldsymbol{\Omega}_i t) \dot{\boldsymbol{\nu}}_i = \boldsymbol{\Omega}_i \boldsymbol{\mu}_i + \exp(\boldsymbol{\Omega}_i t) \dot{\boldsymbol{\nu}}_i. \quad (28)$$

With that we can rewrite dynamical system (9) in new variables

$$\exp(\boldsymbol{\Omega}_i t) \dot{\boldsymbol{\nu}}_i = -(\mathbf{I} - \exp(\boldsymbol{\Omega}_i t) \boldsymbol{\nu}_i \boldsymbol{\nu}_i^\top \exp(-\boldsymbol{\Omega}_i t)) \exp(\boldsymbol{\Omega}_i t) \frac{\partial E_K}{\partial \boldsymbol{\nu}_i} \Rightarrow \dot{\boldsymbol{\nu}}_i = -(\mathbf{I} - \boldsymbol{\nu}_i \boldsymbol{\nu}_i^\top) \frac{\partial E_K}{\partial \boldsymbol{\nu}_i}. \quad (29)$$

So we see that assumption  $\boldsymbol{\Omega}_i = \boldsymbol{\Omega}$  reduces Kuramoto memory model to the constrained gradient flow.

Since energy function has the same form in new variables we find

$$\dot{E}_K(\boldsymbol{\nu}) = \sum_i \left( \frac{\partial E_K(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}_i} \right)^\top \dot{\boldsymbol{\nu}}_i = - \sum_i \left( \frac{\partial E_K(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}_i} \right)^\top (\mathbf{I} - \boldsymbol{\nu}_i \boldsymbol{\nu}_i^\top) \frac{\partial E_K(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}_i}. \quad (30)$$

Using identity for derivatives and relation between  $\boldsymbol{\nu}_i$  and  $\boldsymbol{\mu}_i$  we change variables to the original ones

$$\dot{E}_K(\boldsymbol{\mu}) = - \sum_i \left( \frac{\partial E_K(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^\top (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \frac{\partial E_K(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i}. \quad (31)$$

Since  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_i = 1$ , matrix  $\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$  is orthogonal projector which is positive semidefinite matrix. From that we conclude

$$\dot{E}_K(\boldsymbol{\mu}) = - \sum_i \left( \frac{\partial E_K(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^\top (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \frac{\partial E_K(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \leq 0. \quad (32)$$

## B.2 Direct proof

We define  $s_{\alpha\beta} = \boldsymbol{\mu}_\alpha^\top \boldsymbol{\mu}_\beta$ . Using that energy is function of  $s_{\alpha\beta}$  only we find

$$\frac{\partial E_K(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} = \frac{1}{2} \sum_{\alpha\beta} \frac{\partial E_K}{\partial s_{\alpha\beta}} \frac{\partial s_{\alpha\beta}}{\partial \boldsymbol{\mu}_i} = \frac{1}{2} \sum_{\alpha\beta} \frac{\partial E_K}{\partial s_{\alpha\beta}} (\boldsymbol{\mu}_\beta \delta_{i\alpha} + \boldsymbol{\mu}_\alpha \delta_{i\beta}) = \sum_\beta \frac{\partial E_K}{\partial s_{i\beta}} \boldsymbol{\mu}_\beta. \quad (33)$$

This derivative appears in the equations of motion and also in the time derivative of the energy function. With this identity we find

$$\dot{E}_K(\boldsymbol{\mu}) = \sum_i \left( \frac{\partial E_K(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^\top \dot{\boldsymbol{\mu}}_i = \sum_{ij} \frac{\partial E_K}{\partial s_{ij}} \boldsymbol{\mu}_j^\top \dot{\boldsymbol{\mu}}_i = \sum_{ij} \frac{\partial E_K}{\partial s_{ij}} \boldsymbol{\mu}_j^\top \left( \boldsymbol{\Omega}_i \boldsymbol{\mu}_i - (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \sum_\beta \frac{\partial E_K}{\partial s_{i\beta}} \boldsymbol{\mu}_\beta \right). \quad (34)$$

The first term with  $\boldsymbol{\Omega}_i$  is zero when  $\boldsymbol{\Omega}_i = \boldsymbol{\Omega}$  since  $s_{ij} = s_{ji}$  but  $\boldsymbol{\mu}_j^\top \boldsymbol{\Omega} \boldsymbol{\mu}_i = -\boldsymbol{\mu}_i^\top \boldsymbol{\Omega} \boldsymbol{\mu}_j$ . The remaining term gives

$$\dot{E}_K(\boldsymbol{\mu}) = - \sum_i \left( \sum_j \frac{\partial E_K}{\partial s_{ij}} \boldsymbol{\mu}_j \right)^\top (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \left( \sum_\beta \frac{\partial E_K}{\partial s_{i\beta}} \boldsymbol{\mu}_\beta \right) = - \sum_i \left( \frac{\partial E_K(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^\top (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top) \frac{\partial E_K(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \leq 0. \quad (35)$$

## C Encoding information in scalar products

As explained in the main text, under conditions of Theorem 3.2, Kuramoto model reduces to gradient flow with energy functions that depend only on scalar products. Initial conditions for oscillatory neurons select a particular point on the energy landscape and starting from this point the dynamical system evolves toward the state corresponding to the local energy minimum. It is natural then to encode all information in the scalar products.

Suppose we need to encode  $N - 1$  real numbers  $v_i$  from the interval  $[-1, 1]$ . Evidently, we can select

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \boldsymbol{\mu}_2 = \begin{pmatrix} v_1 \\ 1 - v_1^2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \dots, \boldsymbol{\mu}_N = \begin{pmatrix} v_{N-1} \\ 1 - v_{N-1}^2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad (36)$$

so vectors are normalised  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_i = 1$  and scalar products encode required information  $\boldsymbol{\mu}_1^\top \boldsymbol{\mu}_{i+1} = v_i$ .

This encoding scheme is valid, but one may suspect it is not very efficient since it does not make use of other components of vectors  $v_i$  besides the first two. We will argue here that in the worst case this intuition is wrong and  $N$  oscillatory units can encode only  $N - 1$  real numbers from  $[-1, 1]$  regardless of the chosen scheme and dimension of vectors  $v_i$ .

To see that consider  $\mathbf{v} = \mathbf{1}$ .

First observe that if scalar product of two vectors on the sphere  $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_i = \boldsymbol{\mu}_k^\top \boldsymbol{\mu}_k = 1$  equals one  $\boldsymbol{\mu}_k^\top \boldsymbol{\mu}_i = 1$ , these vectors coincide  $\boldsymbol{\mu}_k = \boldsymbol{\mu}_i$ . This follows from Cauchy–Schwarz inequality.

This implies if we select two vectors  $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$  and encode 1 in their scalar product, we lose  $D$  degrees of freedom.

Any coding strategy that encode components of  $\mathbf{v}$  into scalar products is defined by the order in which pair of vectors is selected  $\boldsymbol{\mu}_{i_1}^\top \boldsymbol{\mu}_{j_1}^\top = v_1, \boldsymbol{\mu}_{i_2}^\top \boldsymbol{\mu}_{j_2}^\top = v_2, \dots$ . The total number of degrees of freedom that we have equals  $(N - 1) \times D$  not  $N \times D$ , because scalar products and dynamical systems are invariant under global rotation. Since each time we pick a novel pair  $i_k \neq j_k$  we lose  $D$  degrees of freedom, we can encode at most  $N - 1$  components.

The worst case result is somewhat frustrating, but we do not expect it to happen too often especially if natural data is considered. Given that, it is an interesting question what happens with “encoding capacity” in the typical case.

## D Lyapunov function for Hopfield-Kuramoto associative memory

First, we split energy function (12) of joint Hopfield-Kuramoto model on three terms

$$\begin{aligned} E_{HK}(\mathbf{x}, \boldsymbol{\mu}) &= \kappa_H E_H(\mathbf{x}) + \kappa_K E_K(\boldsymbol{\mu}) + \Delta E_{HK}(\mathbf{x}, \boldsymbol{\mu}), \\ \Delta E_{HK}(\mathbf{x}, \boldsymbol{\mu}) &= -\frac{1}{2} \sum_{ij} G_{ij} (g(\mathbf{x})_i g(\mathbf{x})_j) \chi_{ij}(\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j). \end{aligned} \quad (37)$$

Time derivative of energy function reads

$$\begin{aligned} \dot{E}_{HK}(\mathbf{x}, \boldsymbol{\mu}) &= \sum_i \frac{\partial E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial x_i} \dot{x}_i + \sum_i \left( \frac{\partial E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^\top \dot{\boldsymbol{\mu}}_i \\ &= \underbrace{\kappa_H \sum_i \frac{\partial E_H(\mathbf{x})}{\partial x_i} \dot{x}_i}_{\text{Appendix A}} + \underbrace{\kappa_K \sum_i \left( \frac{\partial E_K(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^\top \dot{\boldsymbol{\mu}}_i}_{\text{Appendix B}} + \sum_i \frac{\partial \Delta E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial x_i} \dot{x}_i + \sum_i \left( \frac{\partial \Delta E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^\top \dot{\boldsymbol{\mu}}_i. \end{aligned} \quad (38)$$

Since joint model significantly reuses equations of motion of uncoupled models, we can borrow most of the results from Appendix A and Appendix B. The last two terms in the equation above are new, so we focus on them. Hopfield and Kuramoto parts can be considered separately.

For the Hopfield model we obtain

$$\begin{aligned}\frac{\partial \Delta E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial x_i} &= -\frac{1}{2} \sum_{\alpha\beta} G'_{\alpha\beta} (g(\mathbf{x})_{\alpha} g(\mathbf{x})_{\beta}) \chi_{\alpha\beta}(\boldsymbol{\mu}_{\alpha}^{\top} \boldsymbol{\mu}_{\beta}) \frac{\partial (g(\mathbf{x})_{\alpha} g(\mathbf{x})_{\beta})}{\partial x_i} \\ &= -\frac{1}{2} \sum_{\alpha\beta} G'_{\alpha\beta} (g(\mathbf{x})_{\alpha} g(\mathbf{x})_{\beta}) \chi_{\alpha\beta}(\boldsymbol{\mu}_{\alpha}^{\top} \boldsymbol{\mu}_{\beta}) (\Lambda_{i\alpha} g(\mathbf{x})_{\beta} + \Lambda_{i\beta} g(\mathbf{x})_{\alpha}) \\ &= -\sum_{\alpha\beta} \Lambda_{i\alpha} G'_{\alpha\beta} (g(\mathbf{x})_{\alpha} g(\mathbf{x})_{\beta}) \chi_{\alpha\beta}(\boldsymbol{\mu}_{\alpha}^{\top} \boldsymbol{\mu}_{\beta}) g(\mathbf{x})_{\beta} = -\sum_{\alpha} \Lambda_{i\alpha} \Delta x_{\alpha}(\mathbf{x}, \boldsymbol{\mu}).\end{aligned}\quad (39)$$

Reusing results from Appendix A we find

$$\kappa_H \sum_i \frac{\partial E_H(\mathbf{x})}{\partial x_i} \dot{x}_i + \sum_i \frac{\partial \Delta E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial x_i} \dot{x}_i = -\kappa_H \sum_{ij} \Lambda_{ij} \left( \sum_k W_{jk} g(\mathbf{x})_k - x_j + b_j + \frac{1}{\kappa_H} \Delta x_j(\mathbf{x}, \boldsymbol{\mu}) \right) \dot{x}_i. \quad (40)$$

It is easy to see from equations of motion (11) that term in brackets is  $\dot{x}_i$  so we reproduce the first term in derivative of energy function (14).

For the Kuramoto model we find

$$\begin{aligned}\frac{\partial \Delta E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial \mu_i} &= -\frac{1}{2} \sum_{\alpha\beta} G_{\alpha\beta} (g(\mathbf{x})_{\alpha} g(\mathbf{x})_{\beta}) \chi'_{\alpha\beta}(\boldsymbol{\mu}_{\alpha}^{\top} \boldsymbol{\mu}_{\beta}) \frac{\partial (\boldsymbol{\mu}_{\alpha}^{\top} \boldsymbol{\mu}_{\beta})}{\partial \mu_i} \\ &= -\frac{1}{2} \sum_{\alpha\beta} G_{\alpha\beta} (g(\mathbf{x})_{\alpha} g(\mathbf{x})_{\beta}) \chi'_{\alpha\beta}(\boldsymbol{\mu}_{\alpha}^{\top} \boldsymbol{\mu}_{\beta}) (\delta_{\alpha i} \boldsymbol{\mu}_{\beta} + \delta_{\beta i} \boldsymbol{\mu}_{\alpha}) \\ &= -\sum_{\beta} G_{i\beta} (g(\mathbf{x})_i g(\mathbf{x})_{\beta}) \chi'_{i\beta}(\boldsymbol{\mu}_i^{\top} \boldsymbol{\mu}_{\beta}) \boldsymbol{\mu}_{\beta} = -\delta \boldsymbol{\mu}_i(\mathbf{x}, \boldsymbol{\mu})\end{aligned}\quad (41)$$

This result allows us to rewrite equations of motion (11) for oscillatory neurons in the following form

$$\begin{aligned}\dot{\boldsymbol{\mu}}_i &= \boldsymbol{\Omega}_i \boldsymbol{\mu}_i + (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^{\top}) \left( -\frac{\partial E_K}{\partial \boldsymbol{\mu}_i} + \frac{1}{\kappa_K} \delta \boldsymbol{\mu}_i(\mathbf{x}, \boldsymbol{\mu}) \right) \\ &= \boldsymbol{\Omega}_i \boldsymbol{\mu}_i - \frac{1}{\kappa_K} (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^{\top}) \left( \kappa_K \frac{\partial E_K}{\partial \boldsymbol{\mu}_i} + \frac{\partial \Delta E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right) = \boldsymbol{\Omega}_i \boldsymbol{\mu}_i - \frac{1}{\kappa_K} (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^{\top}) \frac{\partial E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i}.\end{aligned}\quad (42)$$

So we can simplify expression for time derivative of energy

$$\sum_i \left( \frac{\partial E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^{\top} \dot{\boldsymbol{\mu}}_i = -\frac{1}{\kappa_K} \sum_i \left( \frac{\partial E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^{\top} (\mathbf{I} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^{\top}) \frac{\partial E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} + \sum_i \left( \frac{\partial E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^{\top} \boldsymbol{\Omega}_i \boldsymbol{\mu}_i. \quad (43)$$

The first term in the equation above reproduces the second term in the derivative of energy function (14). Now, we need to prove that the second term is zero. For that we use result from Appendix B that gives us

$$\sum_i \left( \frac{\partial E_{HK}(\mathbf{x}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} \right)^{\top} \boldsymbol{\Omega}_i \boldsymbol{\mu}_i = \sum_{i\beta} \left( \kappa_K \frac{\partial E_K}{\partial s_{i\beta}} - G_{i\beta} (g(\mathbf{x})_i g(\mathbf{x})_{\beta}) \chi'_{i\beta}(\boldsymbol{\mu}_i^{\top} \boldsymbol{\mu}_{\beta}) \right) \boldsymbol{\mu}_{\beta}^{\top} \boldsymbol{\Omega}_i \boldsymbol{\mu}_i. \quad (44)$$

Expression in brackets is symmetric with respect to  $i$  and  $\beta$  and if  $\boldsymbol{\Omega}_i = \boldsymbol{\Omega}$  scalar product is skew-symmetric  $\boldsymbol{\mu}_{\beta}^{\top} \boldsymbol{\Omega} \boldsymbol{\mu}_i = -\boldsymbol{\mu}_i^{\top} \boldsymbol{\Omega} \boldsymbol{\mu}_{\beta}$ , so contraction is zero.

So we proved that the derivative of energy is given by the expression (14) as claimed in the theorem. Derivative is non-increasing on trajectories since individual terms are quadratic forms with negative semidefinite matrices

## E Hierarchical Hopfield-Kuramoto models

In Section 4 we showed a general construction of the joint Hopfield-Kuramoto associative memory model. Here we will provide several deep architectures with interacting oscillatory neurons and threshold units. All models from this section have global energy function non-increasing on trajectories of joint Hopfield-Kuramoto model (11). We do not provide separate proofs for each case since all of them can be reduced to Theorem 4.1 with particular choice of weights, activations and Kuramoto energy function  $E_K$ .

In all examples below lower index indicate layer. To present results in a more compact way we denote  $(\mathbf{P}_\mu)_{ij} = (\mathbf{I} - \mu_i \mu_i^\top) I_{ij}$  and stack  $i = 1, \dots, N_k$  oscillatory neurons  $\mu_i \in \mathbb{R}^{D+1}$  in a single vector  $\boldsymbol{\mu}^\top = (\mu_1^\top \dots \mu_{N_K}^\top) \in \mathbb{R}^{N \times (D+1)}$ . So with slight abuse of notation we now use  $\mu_i$  to indicate the state of *all* oscillatory neurons rather than the state of  $i$ -th oscillatory neuron as in Kuramoto associative memory (9). In this notation equation (9) with  $\Omega_i = \Omega$  becomes  $\dot{\boldsymbol{\mu}} = \mathbf{I} \otimes \Omega \boldsymbol{\mu} - \mathbf{P}_\mu \frac{\partial E_K}{\partial \boldsymbol{\mu}}$ .

In all cases we consider a network with  $K$  combined threshold-oscillatory neurons. The state of the layer is a pair  $(\mathbf{x}_i \in \mathbb{R}^{N_i}, \mu_i \in \mathbb{R}^{N_i \times (D+1)})$ .

### E.1 Networks with oscillatory lateral connections

The most natural way to build hierarchical networks is to use threshold units for deep connections and oscillatory neurons for lateral connections. We will provide three examples that differ by activation function and structure of the linear layer. In all examples we use minimal coupling between oscillatory and threshold units.

#### E.1.1 Fully connected, ReLU activation

Equations of motion for layer  $i = 2, \dots, K-1$  are

$$\begin{aligned} \dot{\mathbf{x}}_i &= \sum_{p=\pm 1} \mathbf{W}_{ii+p} \text{ReLU}(\mathbf{x}_{i+p}) - \mathbf{x}_i + \mathbf{b}_i + \frac{1}{\kappa_H} \mathbf{S}_i \odot \mathcal{G}(\mu_i, \mu_i) \text{ReLU}(\mathbf{x}_i), \\ \dot{\mu}_i &= \mathbf{I} \otimes \Omega \mu_i - \mathbf{P}_{\mu_i} \frac{\partial E_K}{\partial \mu} + \mathbf{P}_{\mu_i} \frac{1}{\kappa_K} (\text{ReLU}(\mathbf{x}_i) \text{ReLU}(\mathbf{x}_i)^\top \odot \mathbf{S}_i) \otimes \mathbf{I} \mu_i. \end{aligned} \quad (45)$$

where  $\mathbf{W}_{ij}^\top = \mathbf{W}_{ij}$  are weights for feedforward and feedback connections,  $\mathbf{S}_i = \mathbf{S}_i^\top$  are weights of lateral connections. For layer 1 and  $K$  one needs to omit terms with  $\mathbf{x}_{-1}$  and  $\mathbf{x}_{K+1}$  from the equation for threshold units.

Energy function reads

$$\begin{aligned} E_{HK} &= \sum_{i=1}^K (\mathbf{x}_i - \mathbf{b}_i)^\top \text{ReLU}(\mathbf{x}_i) - \frac{1}{2} \sum_{i=1}^K \mathbf{1}^\top \text{ReLU}^2(\mathbf{x}_i) - \sum_{i=1}^{K-1} \text{ReLU}(\mathbf{x}_i)^\top \mathbf{W}_{ii+1} \text{ReLU}(\mathbf{x}_{i+1}) + E_K \\ &\quad + \frac{1}{2} \sum_{i=1}^K \|\text{ReLU}(\mathbf{x}_i) \text{ReLU}(\mathbf{x}_i)^\top \odot \mathbf{S}_i \odot \mathcal{G}(\mu_i, \mu_i)\|_F^2. \end{aligned} \quad (46)$$

From equations of motion we see that layers  $i$  and  $i+1$  are coupled through interaction of threshold units. Threshold units interact with oscillatory units only within each layer, i.e., they form lateral connections. There is no direct interaction between oscillatory units of distinct layers as long as  $E_K$  does not contain them.

#### E.1.2 Convolutional, ReLU activation

It is possible to replace dense layers with convolutional layers for both deep and lateral connections. Since the convolution layer is merely a structured linear layer, one only needs to slightly change interpretation and notation. For example, if network process images, within each layer in (45) index of neuron is triple  $(h, w, c)$

– heights, width and channel, so the state of threshold-oscillatory unit is a pair  $(\mathbf{x}_i \in \mathbb{R}^{H_i \times W_i \times C_i}, \boldsymbol{\mu}_i \in \mathbb{R}^{H_i \times W_i \times C_i \times (D+1)})$ . Next, convolution operation for both threshold and oscillatory unit has the same form

$$x_{h,w,c} = \sum_{i=-k}^k \sum_{j=-k}^k \sum_{o=1}^{N_c} w_{i,j,o} y_{h+i,w+j,o}, \quad (47)$$

but for oscillatory units  $x_{h,w,c}$  and  $y_{h,w,c}$  are replaced by vectors  $\boldsymbol{\mu}_{h,w,c}$ ,  $\boldsymbol{\nu}_{h,w,c}$  since each oscillatory unit has  $D$  rotational degrees of freedom. Other operations are also straightforward. For example, Gram matrix will become

$$(\mathcal{G}(\boldsymbol{\mu}, \boldsymbol{\mu}))_{(h_1, w_1, c_1)(h_2, w_2, c_2)} = \boldsymbol{\mu}_{h_1, w_1, c_1}^\top \boldsymbol{\mu}_{h_2, w_2, c_2}, \quad (48)$$

and the gating mechanism will make parameters of the lateral convolution kernel given by matrix  $\mathbf{S}$  explicitly space-dependent.

### E.1.3 Fully connected, softmax

Dynamical system (45) will have Lyapunov function if we replace  $\text{ReLU}(\mathbf{x})$  by softmax function  $\text{sm}(\mathbf{x}) = \exp(\mathbf{x}) / (\mathbf{1}^\top \exp(\mathbf{x}))$ . More specifically, Lyapunov function becomes

$$E_{HK} = \sum_{i=1}^K (\mathbf{x}_i - \mathbf{b}_i)^\top \text{sm}(\mathbf{x}_i) - \sum_{i=1}^K \log(\mathbf{1}^\top \exp(\mathbf{x}_i)) - \sum_{i=1}^{K-1} \text{sm}(\mathbf{x}_i)^\top \mathbf{W}_{ii+1} \text{sm}(\mathbf{x}_{i+1}) + E_K \\ + \frac{1}{2} \sum_i \|\text{sm}(\mathbf{x}_i) \text{sm}(\mathbf{x}_i)^\top \odot \mathbf{S}_i \odot \mathcal{G}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_i)\|_F^2. \quad (49)$$

It is also possible to implement attention mechanisms from [51] as explained in [35]. It is also possible to use many other activation functions [34].

## E.2 Networks with threshold lateral connections

Oscillatory units can also be used to form deep connections. Here we provide an example with ReLU activations and fully-connected layers, but other activations can be used as well. Equations of motion for layer  $i = 2, \dots, K-1$  are

$$\dot{\mathbf{x}}_i = \mathbf{W}_i \text{ReLU}(\mathbf{x}_i) - \mathbf{x}_i + \mathbf{b}_i + \frac{1}{\kappa_H} \mathbf{S}_i \odot \mathcal{G}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_i) \text{ReLU}(\mathbf{x}_i), \\ \dot{\boldsymbol{\mu}}_i = \mathbf{I} \otimes \boldsymbol{\Omega} \boldsymbol{\mu}_i - \sum_{p=\pm 1} \mathbf{P}_{\mu_i} (\mathbf{W}_{ii+p} \odot \text{ReLU}(\mathcal{G}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_{i+p}))) \otimes \mathbf{I} \boldsymbol{\mu}_{i+p} + \mathbf{P}_{\mu_i} \frac{1}{\kappa_K} (\text{ReLU}(\mathbf{x}_i) \text{ReLU}(\mathbf{x}_i)^\top \odot \mathbf{S}_i) \otimes \mathbf{I} \boldsymbol{\mu}_i. \quad (50)$$

where  $\mathbf{W}_{ij}^\top = \mathbf{W}_{ij}$  are weights for feedforward and feedback connections,  $\mathbf{S}_i = \mathbf{S}_i^\top$  and  $\mathbf{W}_i = \mathbf{W}_i^\top$  are weights of lateral connections. For layer 1 and  $K$  one needs to omit terms with  $\boldsymbol{\mu}_{-1}$  and  $\boldsymbol{\mu}_{K+1}$  from the equation for oscillatory units.

Energy function reads

$$E_{HK} = \sum_{i=1}^K (\mathbf{x}_i - \mathbf{b}_i)^\top \text{ReLU}(\mathbf{x}_i) - \frac{1}{2} \sum_{i=1}^K \mathbf{1}^\top \text{ReLU}^2(\mathbf{x}_i) - \frac{1}{2} \sum_{i=1}^K \text{ReLU}(\mathbf{x}_i)^\top \mathbf{W}_i \text{ReLU}(\mathbf{x}_i) \\ + \frac{1}{2} \sum_{i=1}^K \|\text{ReLU}(\mathbf{x}_i) \text{ReLU}(\mathbf{x}_i)^\top \odot \mathbf{S}_i \odot \mathcal{G}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_i)\|_F^2 + \frac{1}{4} \sum_{i=1}^{K-1} \|\mathbf{W}_{ii+1} \odot \text{ReLU}^2(\mathcal{G}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_{i+1}))\|_F^2 \quad (51)$$

We see that this time oscillatory neurons of layers  $i$  and  $i+1$  interact directly, whereas threshold units interact only within individual layers. Interaction of oscillatory units contain self-gating, so deep connection looks less natural than the ones formed by threshold units.

Dynamical system (50) can be extended on other activation functions and it is possible to use convolutional layers the same way as explained in the previous section.

Table 4: Number of parameters for Kuramoto subnetworks.

$k$	model size
150	$\simeq 17 \times 10^4$
250	$\simeq 28 \times 10^4$
350	$\simeq 39 \times 10^4$
450	$\simeq 50 \times 10^4$

## F Description of fine-tuning experiments

The code that reproduces our numerical results is available in <https://github.com/vlsf/HKmemory>. Trained models are available in <https://disk.yandex.ru/d/n5eQAP0Su4D4pw>.

To implement training of memory models we used JAX [6], Optax [11], Equinox [32], Diffrax [31]. NetworkX was used to generate Watts-Strogatz small world NetworkX [16]. For convenience we split our description into several parts.

### F.1 ODE solver

ODE is solved with Tsitouras’ 5/4 method, step size is adjusted with PID controller, integration interval is  $[0, 1]$ , initial time step  $10^{-2}$ , maximal number of steps allowed is 5000, library used is [31].

### F.2 Optimisation

In all cases we use Lion optimizer [8], with learning rate  $10^{-4}$ , exponential learning rate decay with rate  $\gamma = 0.5$  and 1000 transition steps, number of weight updates is 9000, batch size is 100, number of train examples is 60000, number of test examples is 10000, library used in Optax [11].

### F.3 Hopfield subnetwork

Model is described in the equation (18), input is augmented with 300 additional elements, MNIST images were normalised to  $[0, 1]$  range, weight matrix is dense, the total number of parameters of the model is  $\simeq 12 \times 10^5$ , classes were encoded as  $2\mathbf{e}_i - 1$  where  $\mathbf{e}_i \in \mathbb{R}^{10}$  is  $i$ -th column of identity matrix, training loss is  $L_2$ , and predicted classes are last 10 components of  $\mathbf{x}(1)$  vector, library used in Equinox [32].

### F.4 Kuramoto subnetwork

Model is described in the equation (18), input is augmented with 300 additional elements, MNIST images were normalised to  $[0, 1]$  range, weight matrices are sparse, sparsity patten in generated with Watts-Strogatz small world as implemented in NetworkX with  $p = 0.1$  and  $k = 150, 250, 350, 450$ , rounded number of parameters for Kuramoto subnetworks are given in Table 4, to encode MNIST digit into initial conditions we set  $\boldsymbol{\mu}_1 = \mathbf{e}_1$  and use two first component of the rest of oscillatory neurons to represent normalised pixel values as scalar products  $\boldsymbol{\mu}_1^\top \boldsymbol{\mu}_j$  (see Appendix C), for non-associative training we use  $\boldsymbol{\Omega} = 0$  and rotate vectors for each batch of data randomly during training, for associative training we use trainable  $\boldsymbol{\Omega}$ , on the fine-tuning stage only output of the Hopfield network directly contributed to the loss function.