DYNAMICS-INSPIRED NEUROMORPHIC REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper investigates the dynamics-inspired neuromorphic architecture for neural representation and learning following Hamilton's principle. The proposed approach converts weight-based neural structure to its dynamics-based form that consists of finite sub-models, whose mutual relations measured by computing path integrals amongst their dynamic states are equivalent to the typical neural weights. The feedback signals interpreted as stress forces amongst sub-models push them to move based on the entropy reduction process derived from the Euler-Lagrange equations. We first train a dynamics-based neural model from scratch and observe that this model outperforms its corresponding feedforward neural networks on MNIST dataset. Then we convert several pre-trained neural structures (*e.g.*, DenseNet, ResNet, Transformers, *etc.*) into dynamics-based forms, followed by fine-tuning via entropy reduction to obtain the stabilized dynamic states. We observe consistent improvements of these transformed models on the ImageNet dataset in terms of computational complexity, the number of trainable units, testing accuracy, and robustness. Moreover, we demonstrate the correlation between the performance of a neural system and its structural entropy.

1 INTRODUCTION

A biological brain learns by both the structural evolution via rewiring neural pathways (Chklovskii et al., 2004) and the numerical evolution via strengthening/weakening neural connections (Cho et al., 2015). Following the rule of biological neurons, the artificial neural networks (ANNs) mimic the biological brain with neurons organized in a fixed multi-layered structure organized as the fully connected neural network (Hinton et al., 2006), CNN (Krizhevsky et al., 2017) and Transformers (Liu et al., 2021). In real applications such as image recognition, extensive experiments (Golubeva et al., 2020) reveal that different neural structures demonstrate varying performance on widely-used datasets (*e.g.*, ImageNet (Deng et al., 2009)). Despite the great success already achieved, ANNs are believed to have several intrinsic drawbacks, such as the requirement of a massive number of parameters, the gradient vanishing and/or explosion (Pascanu et al., 2013), and the inefficiency due to the redundant computations (RoyChowdhury et al., 2018). Therefore, the fixed structure of ANNs might be a suboptimal design choice that hinders the ANNs from approximating the brain more efficiently (Han et al., 2021).

It has been proved that evolving neural topologies via methods such as NeuroEvolution (Stanley & Miikkulainen, 2002) and neural architecture search (NAS) (Elsken et al., 2019) that alter the layers and their parameters can significantly outperform their counterparts with fixed structures (Assunção et al., 2018). Nonetheless, the evolving neural mechanisms established as dynamic neural networks (Han et al., 2021) require a large number of expensive fitness evaluations that are inefficient for real-time learning (Stork et al., 2019) and appear to be unstable compared to the fixed structure paradigm. As another direction of research endeavors, some approaches attempt to reconsider the importance of neural weight and structure, for instance, weight agnostic neural network (Gaier & Ha, 2019), Spiking neural networks Basegmez (2014) and weight mirror (Akrout et al., 2019). However, these approaches either still rely on explicit neural weight computing or have yet to develop an efficient learning mechanism, *e.g.*, difficulties of training a spiking neural network via supervised learning (Huynh et al., 2022). In summary, the evolving neural mechanism still relies on the weight-based parameters, which leads to evolutionary inefficiency and huge search space (Ren et al., 2021); thus, the traditional weight-based ANNs can hardly achieve unified structural and numerical learning, which hinders the development of brain-inspired learning system.

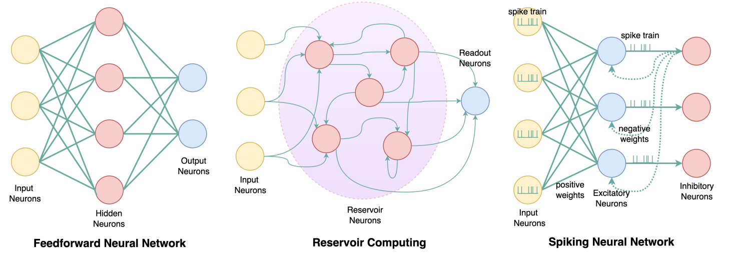
In our study, we consider the structure and numerical learning from neuromorphic dynamics aspect, inspired by Hebb's learning rule (Cooper, 2005). The rule states that neural connections between neurons with similar dynamic behaviors tend to be stronger. Rather than constructing a neural structure with a learning mechanism following Hebb's rule, it is better to make the dynamic behaviors of a neuron directly on its coordinates. In this setting, neurons with similar behavior have stronger connections because their coordinates are intrinsically closer. To formalize the whole learning mechanism, we further reinterpret the universal approximation theorem (UAT) (Scarselli & Tsoi, 1998) into a dynamical alternative, claiming that the neural weight is equivalent to the covariant of neuronal states while retaining their approximation capacity (Fig. 1b). In our proposed principle, neurons receive input signals that drive them to adjust their dynamical states and excite them to fire signals to each other. The signals between neurons decay during transmission (via path integral) and the output signals are obtained when the neuronal states reach equilibrium. In this case, we do not need to treat the neural weight between neurons as the trainable units, since these weights can be expressed as path integrals between trainable neuronal dynamics. Given a specific learning task, the total path length of all neurons that follow the dynamical UAT is theoretically conservative. It gradually approaches an upper bound when dealing with an increasingly complicated computational task (see Section F in the Appendix). Accordingly, we assert that *neural weights are mathematically trivial*. This claim is biologically well supported since neurons in the human brain have good spatial effectiveness (Gerstner et al., 2014), and contain sufficient information for the generation and evolution of neural connections (Camp & Treutlein, 2022).

In this regard, we propose a framework where the essential trainable parameters are the dynamic states of neurons rather than the neural weights connecting neurons, as shown in Fig. 1b. Our neuromorphic architecture DyN applies dynamics-inspired update rules on finite sub-models, *i.e.*, the highly functional neurons capable of receiving-emitting signals and changing dynamic states. Under this formulation, the typical neural weights used in most mainstream neural models can be explicitly avoided by computing the path integral between interacting neuronal dynamics. From a computational aspect, as we replace the classical weight connection with functional integrals (Weinberg, 1995), it allows one to change coordinates between distinct neurons easily. Therefore, this formulation makes the system independent of a network's structural design. It provides a more global and efficient way to implement the interaction between arbitrary neurons of distinct layers, compared to the traditional layer-by-layer update rules (Pascanu et al., 2013).

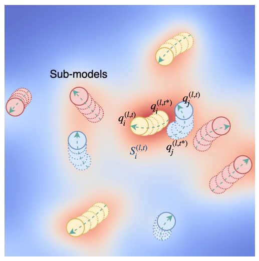
Experimentally, we first validate our DyN on MNIST to verify its capacity as a universal approximator and observe that a DyN layer trained from scratch requires only 7.9% amount of parameters to outperform its equivalent fully-connected layer in a feedforward neural network by 0.26% on accuracy. We transform the weight-based connection blocks of several mainstream pre-trained neural models to the DyN blocks, followed by fine-tuning on ImageNet. With our transformation, the amount of parameters has been reduced by a factor of 5-10, and the transformed models outperform the original ones on the large-scale ImageNet benchmark in terms of improved accuracy and reduced parameter size. These observations reveal the possibility and potentiality of building an efficient neural computing architecture focusing on neuronal dynamics rather than weight transport. Meanwhile, we also consider the practical implementation issue, assuming that existing computing devices contains noisy naturally (Braverman et al., 2015), *e.g.*, the quantization error of digitalization, which affects the model precision during the storage and calculation process. Thus, we assume a model's parameter space is reconstructed by noise (uniform on an ϵ-ball). We round all parameters to a certain precision and see how the performance is affected. The results indicate that our model is robust to different level of noise, which also indicates that the model can be further accelerated via hashing in the future.

2 Preliminaries

Interpreting ANN as a dynamics system: A typical ANN is composed of neurons that follow a specific structure, and these neurons are connected by trainable neural weights of specific values. The neural models equipped with arbitrary mechanisms (*e.g.*, convolution, transformer, *etc.*) are equivalent to the typical ones (will be discussed later). For a neural network, the neurons are inter-



(a) **Weights-based neural networks**: weights are trainable variants that are explicitly isolated from each others; they are treated as the essential parameters that directly affected by the feedback signals such as predictive error during the back-propagation process.



(b) **Dynamics-inspired neural network**: weights are path integrals between neuronal dynamics.

Figure 1: **Comparison between neural networks and dynamics-inspired neuromorphic system.** A dynamics-inspired neural system does not require weights. Weights are mathematically equivalent to the path integral between sub-models' dynamic states (denoted by $q_i^{(l,t)}$, which are the trainable units). The inference phase and the learning phase involve both feedforward and feedback signals that simultaneously push the sub-models continuously move until reaching equilibrium state. A sub-model's dynamic states determine the spatial "density" nearby, affecting the signals travelling among sub-models.

connected by the predefined weighted connection structure. The weights among different neurons are updated during the training stage, with layer-by-layer updating rules.

From the dynamics perspective, an ANN can be viewed as a dynamic system where the neurons are dynamically interacting towards an minimal objective function, *e.g.*, the cross-entropy loss. However, the learning strategy of ANN seems to be local and suboptimal from the dynamics perspective. First, the fixed connection structure constrains the learning process towards a comprehensive and universally optimal configuration. Second, the fixed connection structure imposes an unnecessary computational burden on the training and inference process.

In comparison, our method is straightforward, *i.e.*, by treating each neuron as a basic unit in the whole neuron system, we *replace the trainable neural weights between neurons with the dynamic relations between neuronal dynamics*. In contrary to a typical ANN whose neural weights are trainable units, we suggest that the neurons are trainable units, and the neural weights are just intermediate values measuring the interaction between neurons, and they can be directly accessed from the dynamic states of neurons. Under this dynamics-inspired framework, the neurons are supposed to be fully interacted during the training stage, thus comprehensively releasing the model capacity.

Sub-models and subsystems: In our method, a neuron's dynamic states (*aka.* neuronal dynamics), *e.g.*, spatial location, velocity, acceleration, activation/inhibition state, *etc.*, can be described in a specific phase space in which neurons that are located close to each other or have similar dynamic patterns are closer together. To distinguish them from node-like neurons in the computational sense, we use the concept of *sub-models*. A sub-model is a highly functional neuron capable of receiving or emitting signals and changing its dynamic states. This paper interprets both the signals and the dynamic states as time-variant d-dimensional vectors: $S_i^{(l,t)}$ and $q_i^{(l,t)}$, where t refers to the time-step, i refers to the index of a sub-model, and l refers to the index of the *subsystem* that contains the sub-model i. Note that a group of sub-models sharing identical global settings (*e.g.*, a specified hidden layer in an ANN) is collectively referred to as a *subsystem*.

Basically, any neural layer interpreted as a tensor-formed structure can be equivalently transformed to a set of subsystems. For an MLP or a feedforward DNN, the sub-models corresponding to neurons at the same layer form a subsystem. For a convolution layer (kernel's window shape $k \times k$, with N_{in} input channels and N_{out} output channels), there are k^2 subsystems containing $N_{in} + N_{out}$ sub-models. For a transformer layer, the concatenated matrix $\mathbb{R}^{3d \times d}$ of Query, Key, and Value matrices refers to a subsystem containing $3d + d$ sub-models, in which d represents the dimension of a semantic token. The corresponding learning mechanisms are equivalent to the one applied in the MLP case. The pseudocode for each neural layer is present in Appendix. H).

Universal Approximation Theorem (UAT): The UAT demonstrates the approximation capabilities of a feedforward neural network in the space of continuous functions between two Euclidean spaces. This part focuses on Cybenko's arbitrary-width one. It states that, *for every* $n \in \mathbb{N}$, $m \in \mathbb{N}$, *compact* $K \subseteq \mathbb{R}^n$, $f : K \to \mathbb{R}^n$, $\varepsilon > 0$, *there exists* $k \in \mathbb{N}$, $A \in \mathbb{R}^{k\times n}$, $C \in \mathbb{R}^{m\times k}$ *such that*

$$\|f(x) - C \cdot (\sigma(A \cdot x + b))\| < \varepsilon$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ *is not polynomial.* There are various UAT for the arbitrary-depth case and other widely used architectures like convolutional neural networks. However, they all rely on a setting that the trainable units are neural weights between neurons. Next, we will present an alternative to these typical UAT by considering the neuronal states rather than their weights as the trainable units.

Principle of dynamic subsystems: On the basis of Cybenko's UAT, we propose its *DyN* form in which the principal trainable units are the neurons' dynamic states $Q^{(t)}$ rather than the weight-based parameters $k \in \mathbb{N}$, $A \in \mathbb{R}^{k\times n}$, $C \in \mathbb{R}^{m\times k}$ in UAT. *For every* $d \in \mathbb{N}$, $k \in \mathbb{N}$, $l \in \mathbb{N}$, $N \in \mathbb{N}$, $M \in \mathbb{N}$, *given a system of sub-models with a set of time-variant coordinates* $Q^{(t)} = \{q_i^{(t)} \in \mathbb{R}^d, i \in [1, N]\}$ *that receive and emit time-variant signals* $S_{in}^{(i;t)} \in \mathbb{R}^k$ *and* $S_{out}^{(i;t)} \in \mathbb{R}^l$, *then for arbitrary sequential mapping* F*:* $S_{in}^{(i;1..M)} \in \mathbb{R}^{k\times M} \to S_{out}^{(i;1...M)} \in \mathbb{R}^{l\times M}$, *there exists* $A \in \mathbb{R}^{k\times l}$, $B \in \mathbb{R}^{d\times l}$), $C \in \mathbb{R}^{k\times l}$, $D \in \mathbb{R}^{d\times l}$, $Q^{(t)}$, *and a 2-form* $\varphi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$, *such that for any* $t \in [1, M]$*:*

$$S_{out}^{(i;t)} = A^T S_{in}^{(i;t-\delta)} + B^T \frac{d}{dt} q_i^{(t)}; \quad \frac{d}{dt} q_i^{(t)} = C^T S_{in}^{(i;t-\delta)} + D^T q_i^{(t)}$$
$$S_{in}^{(i;t)} = \sum_{j\neq i}^{N} S_{out}^{(i;t-\delta)} \cdot \varphi(q_i^{(t)}, q_j^{(t-\delta)}) \tag{1}$$

where the non-polynomial 2-form $\varphi(x, y) = \theta(y - x) \cdot \|y - x\|$ *denotes the path integral between* x *and* y*, and is applied on a polar field* $\theta : \mathbb{R}^d \to \mathbb{R}^d$ *that turns the translational variation of* y *to the angular variation of* x *such that the spatial displacements of sub-models have nonlinear effect (e.g., logistic activation function) on each other.*

The proof is presented in Appendix. A. This principle implies the equivalence between a weight-based neural structure and a neuron-based one that considers weights as the covariants of the trainable neuronal states. Consequently, the dynamic relations amongst trainable neurons are sufficient to construct an arbitrary continuous function between topological spaces based on specific metrics. This fact reveals that the trainable neural weights in an ANN might be mathematically trivial, *i.e.*, they can be interpreted as the covariants of neuronal dynamics.

Formulation of a ***DyN*** **system**: The proposed DyN system contains finite subsystems $\{P^{(l)}, l \in [1, L]\}$ interpreted as a directed graph denoted by $\mathscr{G}$. Each subsystem contains finite time-variant (via time-step t) sub-models with d-dimensional neuronal states $\{q_i^{(l,t)} \in \mathbb{R}^d, i \in [1, N]\}$ capable of emitting or receiving signals. The emitting signals are denoted by $E_i^{(l,t)} \in \mathbb{R}^d$, and the receiving signals are denoted by $R_i^{(l,t)} = \Sigma_{j\neq i} E_{ji}^{(l,t)}$, where $E_{ji}^{(l,t)}$ is the signal emitted from a sub-model $q_j^{(l,t^*)}$ at unspecified time-step t^* and then arrived at $q_i^{(l,t)}$ at time-step t. Details of notation t^* and path integral are presented in Appendix. B. The generalized dynamics is given as:

$$\frac{\partial}{\partial t} q_i^{(l,t)} = \sum_{j\neq i}^{N} f^{(l)}(E_j^{(l,t^*)}, q_j^{(l,t^*)}, q_i^{(l,t)}); \quad \frac{\partial}{\partial l} E_i^{(l,t)} = \sum_{k\neq i}^{N} g^{(l)}(R_k^{(l^*,t)}, q_k^{(l^*,t)}, q_i^{(l,t)}) \tag{2}$$

where $f^{(l)}$ and $g^{(l)}$ are specific nonlinear mappings, and l^* represents the index of an adjacent subsystem $P^{(l^*)}$ of $P^{(l)}$ in graph $\mathscr{G}$. Intuitively, a sub-model's dynamic states are related to its received signals and local states; and its initial emitting signal is related to its received signals from the adjacent subsystems. Since the relations between $E_i^{(l,t)}$ and $R_i^{(l,t)}$ are implicitly included in $f^{(l)}$ and $g^{(l)}$, thus, both $E_i^{(l,t)}$ and $R_i^{(l,t)}$ are simplified as $S_i^{(l,t)}$ in the paper.

3 THE DYN MECHANISM

In this section, we first present how to transform a fully-connected layer of shape $\mathbb{R}^{a\times b}$ into a DyN subsystem that contains $a+b$ sub-models via a dynamics-inspired mechanism, then we present how to apply this mechanism to any tensor-formed neural layer. Lastly, we present the equivalence between this dynamics-inspired mechanism and the process of entropy reduction.

Training a linear layer with *DyN* mechanism: For a feed-forward neural network, the l-th fully-connected layer with M input neurons and N output neurons is a matrix $\mathbf{A}\in\mathbb{R}^{M\times N}$. Here, we present a method that converts $\mathbf{A}$ into two subsystems of sub-models: M receival sub-models $Q^{(l)}=\{q_i^{(l)}, i\in[1,M]\}$ and N disposal sub-models $Q^{(l+1)}=\{q_j^{(l+1)}, j\in[1,N]\}$, which are responsible for receiving and emitting signals to the others. The relative positional vector between sub-models of distinct subsystems is denoted by $v_{ij}=q_i^{(l)}-q_j^{(l+1)}$. We assume that $\frac{\partial}{\partial l}E_i^{(l,t)}=0$ in Eq. 2 because we deal with a single linear layer; thus, the dynamic states of multiple subsystems are synchronous, *e.g.*, $q_j^{(l+1,t^*)}=q_j^{(l,t)}$, implying that $q_i^{(l,t)}-q_j^{(l,t^*)}=v_{ij}$ and $E_j^{(l,t^*)}=E_{ij}$. Thus, $f^{(l)}$ in Eq. 2 can be defined as

$$f^{(l)}(E_j^{(l,t^*)}, q_j^{(l,t^*)}, q_i^{(l,t)})=\frac{v_{ij}}{\|v_{ij}\|}\cdot E_{ij} \tag{3}$$

In an idealized case, the dynamic relation between two sub-models is computed as the path integral on an unevenly distributed non-Euclidean space, which is not a computation-friendly formulation. Because a nonlinear relation can be split into a weighted sum of multiple linear relations, we can efficiently deal with the non-linearity by dividing each sub-model into w copies. Each copy refers to a sub-subsystem $P_k^{(l)}=\{q_{ik}^{(l)}, k\in[1,w]\}$. Note that w is not fixed; it changes during learning stage: subsystems with similar behavior are merged into one subsystem, and a subsystem with fluctuating behavior is split into multiple systems. Therefore, an element $D_{ij}^{(l;l+1)}\in\mathbb{R}$ of the distance matrix $D^{(l;l+1)}\in\mathbb{R}^{M\times N}$ between $Q^{(l)}$ and $Q^{(l+1)}$ is

$$D_{ij}^{(l;l+1)}=\sum_{k=1}^{w}h_k^{(l;l+1)}\left\|q_{ik}^{(l)}-q_{jk}^{(l+1)}\right\| \tag{4}$$

where $h_k^{(l;l+1)}\in\mathbb{R}$ represents the shared coefficient between $P_k^{(l)}$ and $P_k^{(l+1)}$. The DyN mechanism can then find proper $Q^{(l)}$ and $Q^{(l+1)}$ such that the weighted distance matrix $D^{(l;l+1)}$ approximate arbitrary matrix $\mathbf{A}\in\mathbb{R}^{M\times N}$. Suppose that $v_{ij;k}=q_{i;k}^{(l)}-q_{j;k}^{(l+1)}$, and the stress force between sub-models under the target $\mathbf{A}$ is denoted by $f_{ij;k}=A_{ij}-\|q_{i;k}^{(l)}-q_{j;k}^{(l+1)}\|$. Since the stress force can be equivalently interpreted as the signal E_{ij} in Eq. 3, thus, the update rules for Qs and h are:

$$\frac{\partial q_{\lambda;k}^{(l_\lambda)}}{\partial t}=-h_k^{(l;l+1)}\sum_{I_\lambda}\frac{v_{ij;k}}{\|v_{ij;k}\|}\cdot f_{ij;k},\quad \frac{\partial h_k^{(l;l+1)}}{\partial t}+\sum_{i,j}\|v_{ij}\|\cdot f_{ij;k}=0 \tag{5}$$

where $(\lambda, l_\lambda, I_\lambda)\in\{(i,l,j),(j,l+1,i)\}$. The last line of Eq. 5 is proposed due to the energy and momentum conservation laws in subsystems since we assume that our system strictly follows the dynamics theory. The proposed mechanism in Eq. 5 can dynamically approximate arbitrary linear transformation and is consistent with the back-propagation algorithm (Appendix. D). Since the hidden states $f_{ij;k}$ are accessible from the signals in $P^{(l)}$ and $P^{(l+1)}$, this iterative DyN update mechanism is capable of approximating arbitrary nonlinear transformation.

Training an MLP with *DyN* mechanism: Given a multilayer perceptron that contains two tensor-formed weights $W^{(ih)}\in\mathbb{R}^{N_{in}\times N_h}$ and $W^{(ho)}\in\mathbb{R}^{N_h\times N_{ou}}$. The transmitting signals along with each layer are defined using a sequence: $[S^{(in)}, \sigma(S^{(in)}), S^{(h)}, \sigma(S^{(h)}), S^{(ou)}]$. Inspired by Eq. 5, we can train this MLP via a *DyN* system containing three subsystems: input neurons $Q^{(in)}=\{q_i^{(in)}, i\in[1,N_{in}]\}$, hidden neurons $Q^{(h)}=\{q_k^{(h)}, k\in[1,N_h]\}$, and output neurons $Q^{(ou)}=\{q_j^{(ou)}, j\in[1,N_{ou}]\}$. The path integral between arbitrary $q_i^{(x)}$ and $q_j^{(y)}$ is denoted by $I_{ij}^{(xy)}$. The resultant signal received by sub-model $q_j^{(y)}$ from all the sub-models in subsystem $Q^{(x)}$ is defined by $\Phi_j^{(xy)}=\Sigma_i\sigma(S_i^{(x)})\cdot v_{ij}^{(xy)}$. Recall that $v_{ij}^{(xy)}=q_i^{(x)}-q_j^{(y)}$. Our goal is to find proper Qs such that

$I_{ij}^{(in;h)} = W_{ij}^{(ih)}$ and $I_{ij}^{(h;ou)} = W_{ij}^{(ho)}$ for each i, j. Using back-propagation to replace $f_{ij;k}$ in Eq. 5 with the gradients *w.r.t.* neural weights as hidden states, we obtain the update rule for Qs as follows

$$[\frac{\partial q_i^{(in,t)}}{\partial t}; \frac{\partial q_k^{(ou,t)}}{\partial t}] \approx [\Phi_i^{(in)} \cdot \Phi_i^{(h;in)}; \Phi_k^{(ou)} \cdot \Phi_k^{(h;ou)}], \quad \frac{\partial q_j^{(h)}}{\partial t} \approx \Phi_j^{(in;h)} + \Phi_j^{(ou;h)} \tag{6}$$

where $\Phi_i^{(in)} = \sigma(S_i^{(in)})$, and $\Phi_k^{(ou)} = T_k^{(out)} - S_k^{(ou)}$ denotes the stress force between the desired output $T_k^{(out)}$ and the actual output $S_k^{(ou)}$. Mathematically, Eq. 6 is equivalent to the process of implicitly updating neural weights via back-propagation while simultaneously minimizing the stress force between neuronal states via Eq. 5.

There are some practical tips to save computational complexity and memory. For instance, the adjacent sub-models that share similar dynamic behaviors can be clustered as a single one, and the resultant spatial coordinates of sub-models under specified precision (see Eq. 8) can be compressed as a sparse matrix via methods like vector quantization. Besides, a sub-model's dynamic states of $\mathbb{R}^d$ with an exact resolution $1/\delta$ can be encoded via a Cantor space $M : \mathbb{R}^d \to \mathbb{N} \in [1, \delta^{-d}]$, where δ^{-d} should not exceed the maximal allowable integer allowed in the current computing system, *e.g.*, 1.8×10^{108} for any floating-point number represented as a 64-bit double-precision value.

Interpreting *DyN* mechanism as entropy-reduction: The dynamics presented in Eq. 6 is also equivalent to the process of entropy reduction derived from the Hamilton's principle and the Euler-Lagrange equation (Appendix. E):

$$\frac{\partial q_i^{(L,t)}}{\partial t} = \eta_i \cdot exp\left(- \frac{\Sigma_{k \neq i} \frac{\partial}{\partial t} L_{ik}^{(L,t)}}{\Sigma_{k \neq i} L_{ik}^{(L,t)}} \cdot t \right) \tag{7}$$

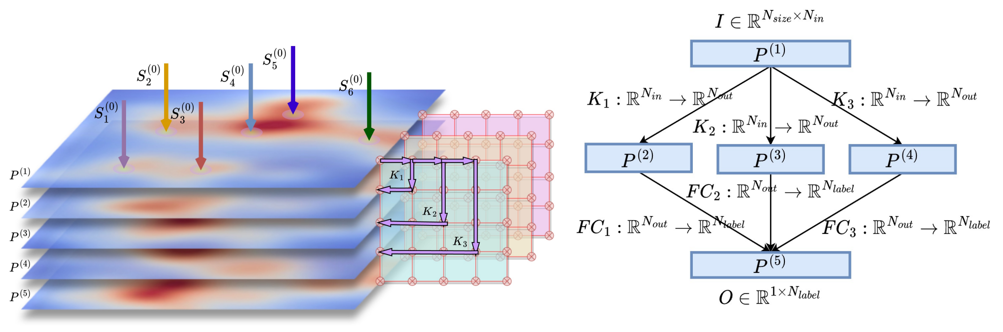
where $q_i^{(L,t)}$ is the dynamic states of a sub-model, and η_i is a constant related to $q_i^{(L,t)}$. The Lagrangian $L_{ik}^{(L,t)} = S_{ik}^{(L,t)} \Phi_i^{(L,t)}$, where $S_{ik}^{(L,t)}$ is emitted from $q_i^{(L,t^*)}$ and received by $q_k^{(L,t)}$, being influenced by the resultant potential field $\Phi_i^{(L,t)}$ around $q_i^{(L,t)}$. The signals $S_{ik}^{(L,t)}$ refers to the feedback control correlated with an objective function, and the trainable potential field $\Phi_i^{(L,t)}$ is related to specified sub-models; we simplify $\Phi_i^{(L,t)}$ as a constant field by adding shared coefficients applied in Eq. 4. Intuitively, a sub-model tends to move toward the region with a lower structural entropy of energy distribution as the traveling signals (*i.e.*, signals just emitted but not yet received) can be regarded as packets of energy (visualized in Eq. 5 of Appendix. F). This result presents a practical approach for physical implementation that a well-formed *DyN* system can be updated in a global dynamics-based way. Furthermore, MLP blocks, convolutional blocks (Fig. 2b), and attention layers can be transformed to the DyN forms; the details are discussed in section 2 and Appendix. H.

4 Experiments

4.1 Evaluation metrics and datasets

Datasets and compared approaches. We evaluate our approaches on two visual classification datasets, MNIST (Deng, 2012) and ImageNet (Deng et al., 2009)). We first conduct experiments on MNIST to compare a simple feedforward neural network trained via back-propagation algorithm with its DyN-formed alternative, which converts each Linear layer of shape $\mathbb{R}^{a \times b}$ into a subsystem containing $a + b$ sub-models. Then we further validate our approaches on ImageNet, by converting mainstream pre-trained neural models from torch.hub, including Inception-v3 (Szegedy et al., 2016), DenseNet-161 (Huang et al., 2017), ResNet-152 (He et al., 2016), ViTs (Dosovitskiy et al., 2020), Swin-Transformer (Liu et al., 2021), to their DyN forms. Note that on both datasets, for fair comparison, the model configuration of the original ANNs and their DyN counterparts (*e.g.*, the number of hidden units, validation criterion, *etc.*) are set to be the same.

Evaluation metrics. On both datasets, we report the top-1 accuracy (the training procedure for each configuration is repeated 20 times to calculate its variance), the number of parameters, and the computational complexity that measures how many operations are implemented for each model during the inference phase. In addition to the typical evaluation (*e.g.*, *ideal* column in Table. 2), we also conduct experiments under varying resolution $1/\delta(\delta > 0)$ that measures a model's robustness


$S_1^{(0)}$
$S_2^{(0)}$
$S_3^{(0)}$
$S_4^{(0)}$
$S_5^{(0)}$
$S_6^{(0)}$
$P^{(1)}$
$P^{(2)}$
$P^{(3)}$
$P^{(4)}$
$P^{(5)}$
K_1
K_2
K_3
$I \in \mathbb{R}^{N_{size} \times N_{in}}$
$K_1 : \mathbb{R}^{N_{in}} \to \mathbb{R}^{N_{out}}$
$K_2 : \mathbb{R}^{N_{in}} \to \mathbb{R}^{N_{out}}$
$K_3 : \mathbb{R}^{N_{in}} \to \mathbb{R}^{N_{out}}$
$FC_1 : \mathbb{R}^{N_{out}} \to \mathbb{R}^{N_{label}}$
$FC_2 : \mathbb{R}^{N_{out}} \to \mathbb{R}^{N_{label}}$
$FC_3 : \mathbb{R}^{N_{out}} \to \mathbb{R}^{N_{label}}$
$O \in \mathbb{R}^{1 \times N_{label}}$

(a) A DyN system in physical view (b) The corresponding CNN in DyN form

Figure 2: **a.** A DyN system consists of layered subsystems, whose hierarchical structures are controlled by specifying the relations amongst them (the right side). The external signals hit the uppermost subsystem $P^{(1)}$ and interact with the sub-models inside, the processed signals are transmitted based on the specified relations between subsystems. **b.** The subsystem $P^{(1)}$ receives an input image I with N_{size} pixels as N_{in} channels signals, and the three subsystems $P^{(2)}$, $P^{(3)}$ and $P^{(4)}$ responsible for processing signals in distinct kernel-units as convolutional layers K_1, K_2 and K_3. The processed signals are passed through the fully-connected layers FC_1, FC_2 and FC_3 then concatenated as an output vector O presented in the subsystem $P^{(5)}$.

during inference stage. This hyper-parameter truncates the trainable parameters $W \in \mathbb{R}^{m \times n}$ into its physical form $W_{py} \in \mathbb{R}^{m \times n}$ via:

$$W_{py} = \|p_\delta(W)\| = \delta \cdot \lfloor \frac{\|W - W_{min}\|}{\delta \cdot \|W_{max} - W_{min}\|} \rfloor \tag{8}$$

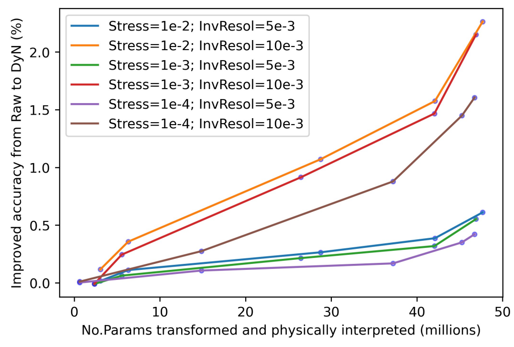
For the ideal case where $\delta = 0$, $W_{py} = \frac{\|W - W_{min}\|}{\|W_{max} - W_{min}\|}$. Note that the trainable units in a weight-based neural model are the weights $\{w_i, i \in [0, mn]\}$, while the ones in a DyN model are the dynamic states $Q = [q_i^j \in \mathbb{R}^d, i \in [1, m+n], j \in [1, d]]$ of sub-models. This fact implies that $p_\delta(w_i) = p_{\delta/\sqrt{d}}(q_i^j)$. The computational complexities of a weight-based model and its DyN alternative are measured using Multiply-Accumulate units (MACs), measured by the number of FLOPs (Patil & Kulkarni, 2018). As for a DyN model, we also compute its computational complexity in physical way (denoted in brackets next to the MACs), which assumes that the computational process of path integral amongst sub-models is reached instantaneously without any computational complexity (a phenomenon that exists in the spatiotemporal liquid crystal structures (Zhang et al., 2021)). The dimension d of dynamic states is often set to 9 (each of position, momentum, and acceleration accounts for 3 states) unless otherwise noted, since we are concerning with the three-dimensional cases that are consistent with the hardware implementation in physical world. The models are trained on a cloud server with eight NVIDIA RTX3090 24GB GPUs.

4.2 Visual Classification

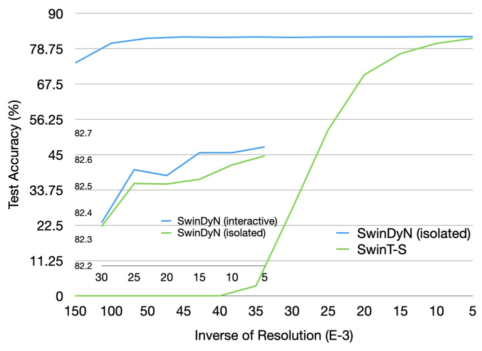
The main results on two datasets are presented in Tab. 1 and Tab. 2. Compared against feedforward neural networks and a LeNet-5, our randomly initialized DyN models from scratch trained via Eq. 6 achieve consistent improvements in accuracy and computational complexity with fewer parameters. Then we use several pre-trained models as backbone networks by converting their MLP-layers, convolution-layers, and attention-layers into DyN forms. The final dynamic states of the sub-models are determined via fine-tuning the transformed neural models based on ImageNet and the original weights; this process continues until the stress force amongst sub-models is lower than a certain threshold (*e.g.*, 10^{-3} of the normalized distances between sub-models). Specifically, we transform each $\mathbb{R}^{M \times N}$ module to a finite amount H of subsystems consisting of M receival sub-models and N disposal sub-models. The upper-bound and selection of H-value is discussed in Appendix. A. We observe that each neural model transformed via DyN mechanism achieves significant improvement in accuracy especially with a lower resolution. Furthermore, a neural model with more neural blocks transformed via DyN mechanism performs better than the one with less DyN blocks (*e.g.*, swinDyN in Tab. 2 and Fig. 3a). These results show that DyN mechanism can preserve more information captured from the ground-truth in an efficient way with a denser structure.

Table 1: Evaluating Neural Models and Their DyN Forms on MNIST

Model Configs		No.Params	C.Ratio	MFLOPs	Top1-TestAcc (50-epochs)
Structure	Layer Type				
2-layer MLP	FC Linear	0.52M	1	0.52	97.180±0.010
	DyN subsys	0.041M	12.68	0.32 (0.07)	97.345±0.005
3-layer MLP	FC Linear	2.29M	1	2.30	97.898±0.010
	DyN subsys	0.2M	11.45	0.61 (0.14)	98.077±0.011
4-layer MLP	FC Linear	4.55M	1	4.56	97.955±0.032
	DyN subsys	0.36M	12.65	1.12 (0.18)	98.215±0.013
LeNet-5	FC Linear, Conv	61.7K	1	0.60	98.681±0.003
	DyN subsys	8.3K	7.43	0.29 (0.18)	98.821±0.001



(a)



(b)

Figure 3: a) As the number of parameters transformed and instantiated via physical view increases, the DyN models outperform their original forms more significantly as the improved accuracy increases. The presented models are distinguished in terms of the inverse of resolution δ and the stress-threshold that determines when DyN process ends. b) As δ increases, though the original model performance greatly degrades, the DyN alternatives remain almost unchanged. The interactive one (Eq. 50) outperforms the isolated one (Eq. 5). This phenomenon is attributed to the global dynamics of sub-models that are moving and interacting with others in a fully stabilized manner.

4.3 Correlation with structural entropy

We examine our systems under different settings of the resolution parameter δ (*e.g.*, the coordinates of sub-models) via Eq. 8. Though larger δ leads to lower model accuracy, we observe an existence of peak that corresponds to the optimal setting of δ such that a model achieves its best testing accuracy (Fig. 6b in Appendix. F). We then postulate that the peak corresponds to some regularization effect that prevents over-fitting. This regularization term is also related to the cross entropy loss and the system's structural configuration. To validate our postulate, we first evaluate the new coordinates q_i of sub-models due to varying δ by $q_i(\delta) = \delta \times \lfloor q_i/\delta \rfloor$, then we count the spatial distribution of each newly resulted sub-model in terms of coordinates $Pr(v_x, \delta) = |\{q_i | q_i(\delta) = v_x\}| / |\{q_i\}|$, and next we obtain the function of structural entropy $\psi(\delta)$:

$$\psi(\delta) = -\sum_{v_x} Pr(v_x, \delta) \cdot \log Pr(v_x, \delta) \tag{9}$$

The structural entropy $\psi(\delta)$ defined in Eq. 9 measures the structural disorder correlated to the energy distribution of the system. To link the structural entropy with its physical meaning, we evaluate the

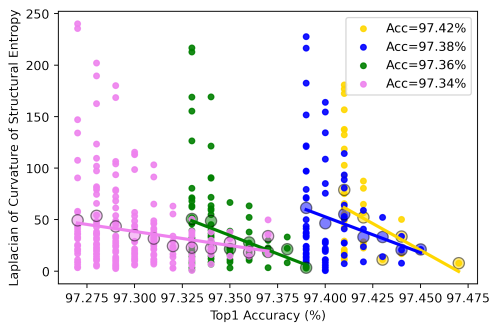
Table 2: Evaluating Weight-based Neural Models and Their DyN Forms on ImageNet

Model Configs		No.Params	C.Ratio	GFLOPs	Top1-TestAcc		
Structure	Layer Type				Ideal	δ=1e-3	δ=5e-3
Inception-V3	FC, Conv	27.20M	1	2.85	67.278	65.400	58.176
	DyN subsys	3.66M	7.64	1.43 (2.2e-3)	67.382	67.092	66.764
DenseNet-161	FC, Conv	28.68M	1	7.82	75.254	71.336	48.594
	DyN subsys	6.05M	4.78	3.28 (0.089)	75.314	75.246	75.294
ResNet-152	FC, Conv	60.40M	1	11.58	77.014	75.776	71.692
	DyN subsys	6.51M	9.29	5.25 (3.5e-3)	77.203	76.604	76.544
ViT-S-224	FC, Conv, Attn	36.38M	1	1.11	80.108	80.038	79.970
	DyN subsys	3.71M	9.83	0.45 (0.75e-3)	80.150	80.122	80.116
SwinT-S-224	FC, Conv, Attn	49.94M	1	8.52	82.634	82.07	80.452
	DyN subsys	10.38M	4.81	3.35 (0.024)	82.646	82.604	82.594
		6.65M	7.51	2.37 (0.018)	82.688	82.66	82.604

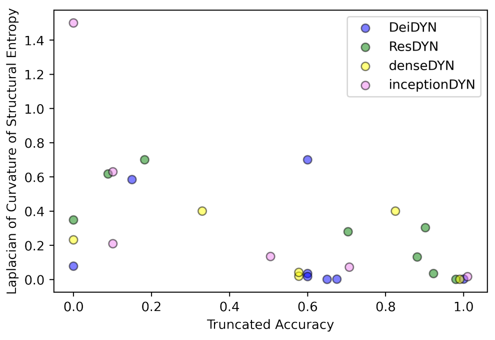
Laplacian of curvature κ of $\psi(\delta)$ (denoted by $LapCurSE$) which accounts for the energy of surface diffusion flow (Sethian & Chopp, 1999)

$$LapCurSE(\delta) = \left\| \frac{\partial^2}{\partial \delta^2} \kappa(\psi(\delta)) \right\| \tag{10}$$

and we observe that an optimal structural setting always refers to a lower $LapCurSE$, whose expected value is negatively correlated with model performance(Fig. 4a). This observation implies that an optimal performance requires a stable structure instantiated as a minimal surface of energy distribution: $\delta_{optimal} = \arg\min_{\delta} LapCurSE(\delta)$, which ensures that all sub-models find the dynamic states that make them the most stable as a whole with the lowest energy. We further evaluate the $LapCurSE$s of the *DyN* models of several mainstream models on ImageNet, and observe that an optimal performance always refers to a lower $LapCurSE$ (Fig. 4b).



(a) 2-layered DyN models on MNIST



(b) Deep DyN models on ImageNet

Figure 4: Scattered points and their expectations that represent model performances for simple 2-layer DyN model with distinct resolution on MNIST (a), and for several mainstream neural models transformed via DyN approach on ImageNet (b).

5 Conclusion

We propose a dynamics-inspired neuromorphic architecture that interprets neural representation and learning from dynamics theory. It emphasizes the state representation of the neurons rather than the neural weights. The experimental results show that our architecture conducts full exploitation of each neuronal parameter, demonstrating significant advantages over other neural models in visual classification in terms of testing accuracy, size of trainable units, and computational complexity. Future work will be devoted to the application of Dyn on multimodal data, new tasks (*e.g.*, retrieval and QA) and a more elegant physical interpretation from the dynamics theory.

REFERENCES

Henry DI Abarbanel and A Rouhi. Phase space density representation of inviscid fluid dynamics. *The Physics of fluids*, 30(10):2952–2964, 1987.

Mohamed Akrout, Collin Wilson, Peter Humphreys, Timothy Lillicrap, and Douglas B Tweed. Deep learning without weight transport. *Advances in neural information processing systems*, 32, 2019.

Filipe Assunção, Nuno Lourenço, Penousal Machado, and Bernardete Ribeiro. Evolving the topology of large scale deep neural networks. In *European Conference on Genetic Programming*, pp. 19–34. Springer, 2018.

Erdem Basegmez. The next generation neural networks: Deep learning and spiking neural networks. In *Advanced Seminar in Technical University of Munich*, pp. 1–40. Citeseer, 2014.

Mark Braverman, Jonathan Schneider, and Cristóbal Rojas. Space-bounded church-turing thesis and computational tractability of closed systems. *Physical review letters*, 115(9):098701, 2015.

J Gray Camp and Barbara Treutlein. Human brain organoids influence rat behaviour, 2022.

Dmitri B Chklovskii, BW Mel, and K Svoboda. Cortical rewiring and information storage. *Nature*, 431(7010):782–788, 2004.

Richard W Cho, Lauren K Buhl, Dina Volfson, Adrienne Tran, Feng Li, Yulia Akbergenova, and J Troy Littleton. Phosphorylation of complexin by pka regulates activity-dependent spontaneous neurotransmitter release and structural synaptic plasticity. *Neuron*, 88(4):749–761, 2015.

Steven J Cooper. Donald o. hebb's synapse and learning rule: a history and commentary. *Neuroscience & Biobehavioral Reviews*, 28(8):851–874, 2005.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Adam Gaier and David Ha. Weight agnostic neural networks. *Advances in neural information processing systems*, 32, 2019.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Anna Golubeva, Behnam Neyshabur, and Guy Gur-Ari. Are wider nets better given the same number of parameters? *arXiv preprint arXiv:2010.14495*, 2020.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Phu Khanh Huynh, M Lakshmi Varshika, Ankita Paul, Murat Isik, Adarsha Balaji, and Anup Das. Implementing spiking neural networks on neuromorphic architectures: A review. *arXiv preprint arXiv:2202.08897*, 2022.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017. ISSN 0001-0782.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. PMLR, 2013.

Priyanka A Patil and Charudatta Kulkarni. A survey on multiply accumulate unit. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–5. IEEE, 2018.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

Aruni RoyChowdhury, Prakhar Sharma, and Erik G. Learned-Miller. Reducing duplicate filters in deep neural networks. 2018.

Franco Scarselli and Ah Chung Tsoi. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural networks*, 11(1):15–37, 1998.

James A Sethian and David Chopp. Motion by intrinsic laplacian of curvature. *Interfaces and Free boundaries*, 1(1):107–123, 1999.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Jörg Stork, Martin Zaefferer, and Thomas Bartz-Beielstein. Improving neuroevolution efficiency by surrogate model-based optimization with phenotypic distance kernels. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pp. 504–519. Springer, 2019.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Steven Weinberg. *The quantum theory of fields*, volume 2. Cambridge university press, 1995.

Rui Zhang, Steven A Redford, Paul V Ruijgrok, Nitin Kumar, Ali Mozaffari, Sasha Zemsky, Aaron R Dinner, Vincenzo Vitelli, Zev Bryant, Margaret L Gardel, et al. Spatiotemporal control of liquid crystal structure and dynamics through activity patterning. *Nature materials*, 20(6): 875–882, 2021.

APPENDIX

A SOME PRINCIPLES AND PROOFS

Theorem A.1. Principle of dynamic subsystems *(the existence of global neuronal rules for the dynamic system as a universal approximator): For every $d \in \mathbb{N}$, $k \in \mathbb{N}$, $l \in \mathbb{N}$, $N \in \mathbb{N}$, $M \in \mathbb{N}$, given a system of sub-models with a set of time-variant coordinates $Q^{(t)} = \{q_i^{(t)} \in \mathbb{R}^d | i \in [1, N]\}$ that receive and emit time-variant signals $S_{in}^{(i;t)} \in \mathbb{R}^k$ and $S_{out}^{(i;t)} \in \mathbb{R}^l$, then for arbitrary sequential mapping F: $S_{in}^{(i;1..M)} \in \mathbb{R}^{(k\times M)} \to S_{out}^{(i;1...M)} \in \mathbb{R}^{(l\times M)}$, there exists $A \in \mathbb{R}^{(k\times l)}$, $B \in \mathbb{R}^{(d\times l)}$, $C \in \mathbb{R}^{(k\times l)}$, $D \in \mathbb{R}^{(d\times l)}$, $Q^{(t)}$, and a 2-form $\varphi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$, such that for any $t \in [1, M]$:*

$$
\begin{aligned}
S_{out}^{(i;t)} &= A^T S_{in}^{(i;t-\delta)} + B^T \frac{d}{dt} q_i^{(t)} \\
\frac{d}{dt} q_i^{(t)} &= C^T S_{in}^{(i;t-\delta)} + D^T q_i^{(t)} \\
S_{in}^{(i;t)} &= \sum_{j\neq i}^{N} S_{out}^{(i;t-\delta)} \cdot \varphi(q_i^{(t)}, q_j^{(t-\delta)})
\end{aligned}
\tag{11}
$$

where the 2-form $\varphi(x, y) = \theta(y - x) \cdot \|y - x\|$ denotes the path integral between x and y, and is applied on a polar field $\theta : \mathbb{R}^d \to \mathbb{R}^d$ that turns the translational variation of y to the angular variation of x such that the spatial displacement of other sub-models has nonlinear effect (e.g., logistic activation function) on the given sub-model as an observer.

Proof. According to Lemma A.2 and Lemma A.3, an arbitrary linear transformation requires finite distinct subsystems, and an arbitrary nonlinear transformation can be achieved by a universal rule of dynamics H obeyed by each sub-model $q_i^{(t)}$:

$$
S_{ii}^{(t+\delta)}, q_i^{(t+\delta)} = H(\Sigma_i^{(t)}, q_i^{(t)}) \tag{12}
$$

where $S_{ii}^{(t)}$ is the signal emitted from the sub-model $q_i^{(t)}$ itself, and $\Sigma_i^{(t)}$ is the resultant signals received by $q_i^{(t)}$ from all the other sub-models; note that $S_{ij}^{(t)}$ refers to the signal emitted from $q_i^{(t)}$ and received by $q_j^{(t)}$. The Eq. 12 can be regarded as the static form of Theorem A.1. The dynamic form will be proved by induction: given a set of signals $S^{(t)}$ received by the DyN system and an arbitrary 2-form $F : Q^{(t)} \times S^{(t)} \to Q^{(t+1)} \times S^{(t+1)}$, there exists a set of static states of sub-models $Q^{(t)}$ such that

$$
F(S^{(t+1)}, Q^{(t+1)}) = H(\Sigma^{(t)}, Q^t) \tag{13}
$$

Eq. 13 is obviously consistent with Eq. 12 based on Lemma A.3. This fact implies that the sub-models with specific dynamic states can approximate arbitrary instant nonlinear transformation, and such specified dynamic states can be obtained by a specific predecessor states under the constraint that the received signals are provided. This recursive step finally constructs a complete dynamic form of Lemma A.3.

□

Lemma A.2. *The weighted sum of H distinct distance matrices is sufficient to approximate any matrix $T \in \mathbb{R}^{m\times n}$ in any degree of precision. Specifically, the upper-bound of an optimized H is given by*

$$
H^{optimized} \leq \frac{mn}{d \cdot (m + n)} + 1 \tag{14}
$$

where d refers to the dimension of units (sub-models) whose relative distances are computed to generate those distance matrices.

Proof. First we count the number of possibilities an arbitrary matrix might be. The elements of the matrix T range from 0 to 1 with a precision scale ϵ that divides the domain into ϵ partitions. Then the total number of possibilities is $\Phi(T) = \epsilon^{mn}$. Likewise, the permutations of m sub-models

$Q^{row} \in \mathbb{R}^{m\times d}$ and n sub-models $Q^{col} \in \mathbb{R}^{n\times d}$ are $\Phi(Q^{row}) = \epsilon^{md}$ and $\Phi(Q^{col}) = \epsilon^{nd}$, implying that the number of combinations of Q^{row} and Q^{col} is $\Phi([Q^{row}; Q^{col}]) = \Phi(Q) = \epsilon^{d(m+n)}$. Then we need to eliminate the duplicate states of Q, which can be categorized into three cases, *i.e.*, self-permutation (SP), transitional invariance (TI) and rotational invariance (RI).

$$\begin{aligned}\Phi_{SP}(Q) &= (m!\cdot n!)^d\\ \Phi_{TI}(Q) &= \epsilon\cdot\prod_{k=1}^{d}(1-L_k^{row}L_k^{col})\\ \Phi_{RI} &= S^{(d)}(max(L_k^{row}L_k^{col})\cdot\epsilon)\end{aligned} \tag{15}$$

where $L_k^{row} = \max(Q^{row}[:,k]) - \min(Q^{row}[:,k])$ measures the maximum range of Q^{row} in the k-th dimension, and $S^{(d)}$ is the spherical area of d-sphere. Therefore, each pair of Q^{row} and Q^{col} covers $\Psi(Q)$ non-duplicate states, which are given by

$$\Psi(Q) = \frac{\Phi(Q)}{\Phi_{SP}(Q)\cdot\Phi_{TI}(Q)\cdot\Phi_{RI}(Q)}$$

Then the total number of non-duplicate states achieved by H subsystems is

$$\Phi([Q^{(1)}; Q^{(2)}...Q^{(H)}]) = \Phi_H(Q) = \frac{(\Psi(Q))^H}{(H!)^d}$$

Our goal is to find a proper H such that $\Phi_H(Q) = \Phi(T)$, yielding

$$\lim_{\epsilon\to\infty} H = \frac{mn}{d\cdot(m+n)} \le \frac{mn}{d\cdot(m+n)} + 1 \tag{16}$$

which is consistent with Eq. 14. □

Lemma A.3. Existence of universal rule for static subsystems: *given a DyN system composed of sub-models whose rules of dynamics determine their successive states interacted with emitted or received signals, for any continuous function that maps the receival signals to the emitted signals of all the sub-models, there exists a universal rule of dynamics followed by every sub-model, ensuring that the relation between the receival signals and the emitted signals of all the sub-models approximates the provided continuous function. The universal rule of dynamics is mathematically equivalent to a set of linear transformations.*

Proof. Suppose there exists a fake sub-model $q_*^{(t)}$ that complements the signals received by each real sub-model, and an expected universal rule H if it exists. Each real sub-model $q_i^{(t)}$ is equipped with an exclusive rule of dynamics ψ_i that might be different from H, we have

$$S_{ii}^{(t+\delta)}, q_i^{(t+\delta)} = \psi_i(\Sigma_i^{(t)}, q_i^{(t)}) = H(\Sigma_i^{(t)} + S_{*i}^{(t)}, q_i^{(t)}) \tag{17}$$

Taking the total derivatives over time on the middle and right sides of Eq. 17

$$\frac{\partial\psi_i}{\partial q_i^{(t)}}\cdot\frac{\partial q_i^{(t)}}{\partial t} + \frac{\partial\psi_i}{\partial\Sigma_i^{(t)}}\cdot\frac{\partial\Sigma_i^{(t)}}{\partial t} = \frac{\partial H}{\partial q_i^{(t)}}\cdot\frac{\partial q_i^{(t)}}{\partial t} + \frac{\partial H}{\partial\Sigma_i^{(t)}}\cdot\frac{\partial\Sigma_i^{(t)}}{\partial t} + \frac{\partial H}{\partial S_{*i}^{(t)}}\cdot\frac{\partial S_{*i}^{(t)}}{\partial t} \tag{18}$$

The signal emitted from the fake sub-model is expected to be a zero constant, therefore,

$$\Big(\frac{\partial\psi_i}{\partial q_i^{(t)}} - \frac{\partial H}{\partial q_i^{(t)}}\Big)\cdot\frac{\partial q_i^{(t)}}{\partial t} = \Big(\frac{\partial H}{\partial\Sigma_i^{(t)}} - \frac{\partial\Sigma_i^{(t)}}{\partial t}\Big)\cdot\frac{\partial\Sigma_i^{(t)}}{\partial t} \tag{19}$$

since $q_i^{(t)}$ and $\Sigma_i^{(t)}$ cannot be constant variables, the Eq. 19 is satisfied for all cases if and only if

$$\frac{\partial\psi_i}{\partial q_i^{(t)}} - \frac{\partial H}{\partial q_i^{(t)}} = \frac{\partial H}{\partial\Sigma_i^{(t)}} - \frac{\partial\Sigma_i^{(t)}}{\partial t} = 0 \tag{20}$$

which implies that

$$H(\Sigma_i^{(t)}, q_i^{(t)}) = W_q\cdot q_i^{(t)} + W_\Sigma\cdot\Sigma_i^{(t)} + b = \psi_i(\Sigma_i^{(t)}, q_i^{(t)}) + b^{'} \tag{21}$$

Hence, if each sub-model's rule of dynamics refers to its exclusive linear transformation ψ_i, then there exists a universal linear transformation substituting for each specified ψ_i such that the expected nonlinear transformation of the DyN system remains unchanged based on Lemma A.4. □

Lemma A.4. *If each sub-model's rule of dynamics is a linear transformation, then the configured system can approximate any continuous function to the expected precision.*

Proof. According to Lemma A.2, the configured system can approximate any normalized matrix, which corresponds to the neural weights according to the universal approximation theorem, by manipulating the states of sub-models. The combination of each sub-model's linear transformation and the overall weights between each pair of sub-models is equivalent to a feedforward neural network with arbitrary width as illustrated in the universal approximation theorem (Scarselli & Tsoi, 1998). □

B PATH INTEGRAL FORMULATION FOR NEUROMORPHIC SYSTEM

Recall that the emitting signals are denoted by $E_i^{(l,t)} \in \mathbb{R}^d$, and the receiving signals are denoted by $R_i^{(l,t)} = \Sigma_{j \neq i} E_{ji}^{(l,t)}$, where $E_{ji}^{(l,t)}$ represents the signal emitted from a sub-model $q_j^{(l,t^*)}$ and then arrived at $q_i^{(l,t)}$ at time-step t. Note that the emitting time t^* is irrelevant since we only care about the arriving states; there might be a case that a single signal currently received by a sub-model is the result of the signals emitted by another sub-model at different time steps. Each subsystem refers to a time-variant global field $H^{(l,t)} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ that assigns each possible dynamic state with a scalar-valued vector. Then an easy-to-compute formulation of path integral between $E_i^{(l,t^*)}$ and $E_{ij}^{(l,t)}$ can be evaluated by:

$$E_{ij}^{(l,t)} = \sum_{t^* \in U_{ij}^{(l,t)}} \int_{t^*}^{t} H^{(l,\delta)}\big(r_{ij}^{(l,\delta)}\big) \cdot E_i^{(l,t^*)} d\delta \tag{22}$$

where $U_{ij}^{(l,T)}$ is the collection of all the admissible time-steps t^* such that the signal emitted from $q_i^{(l,t^*)}$ will eventually arrive at $q_j^{(l,t)}$, and $r_{ij}^{(l,t)}$ represents the intermediate dynamic state on the specified path from $q_i^{(l,t^*)}$ to $q_j^{(l,t)}$.

C PINCIPLE OF HIERARCHICAL STRUCTURES

Let's define $S_{ij}^{(k,t)}$ as the signals emitted from $q_i^{(l,t^*)}$ at an unspecified time-step t^* and received by $q_j^{(k,t)}$ at a specified time-step t. Then the directed edge from $P^{(l)}$ to $P^{(k)}$ means that there exists a linear mapping $W^{(lk)} : S_i^{(l,t)} \times q_i^{(l,t)} \times q_j^{(k,t+1)} \rightarrow S_{ij}^{(k,t+1)}$ and a linear mapping $V^{(k)} : q_j^{(k,t)} \times S_{ij}^{(k,t)} \rightarrow q_j^{(k,t+1)}$ such that

$$S_j^{(k,t+1)} = \sum_{i \neq j} W^{(lk)}(S_i^{(l,t)}, q_i^{(l,t)}, V^{(k)}(q_j^{(k,t)}, W^{(lk)}(S_i^{(l,t-1)}, q_i^{(l,t-1)}, q_j^{(k,t)})))$$

According to Lie algebra homomorphism, there exists a nonlinear mapping $U^{(lk)}$ such that

$$\frac{\partial}{\partial t} S_j^{(k,t+1)}, \frac{\partial}{\partial t} q_j^{(k,t)} = \sum_{i \neq j} U^{(lk)}(\frac{\partial}{\partial t} S_i^{(l,t-1)}, \frac{\partial}{\partial t} q_i^{(l,t)}) \tag{23}$$

The Eq. 23 is called the principle of hierarchical structures (PoHS). Suppose there is a tree-based structure that describe the recursive relations between the root system and its subsystems, sub-subsystems, ... *etc.*. Then PoHS states that this tree-based structure is equivalent to a linearly hierarchical structure containing a set of subsystems. Furthermore, Eq. 23 reveals that a well-formed neuromorphic system does not require a specified set of discrete trainable units that are isolated from each other.

D CONSISTENCY WITH BACK-PROPAGATION

First let's deduce Eq. 5 using approaches applied in back-propagation. This equation is initially derived from a dynamics-inspired view in the main paper. Specifically, we will deduce Eq. 5 by computing the gradient of the loss function with respect to each trainable parameter by the chain rule. The loss function between two arbitrary sub-models $q_{i;k}^{(l)}$ and $q_{j;k}^{(l+1)}$ of distinct subsystems is defined as

$$\begin{gathered} E_{ij} = (A_{ij} - D_{ij}^{(l;l+1)})^2 \\ E_i = \sum_{j=1}^{M} E_{ij} \end{gathered} \tag{24}$$

where $A_{ij} \in \mathbb{R}^{M \times N}$ and $D_{ij} \in \mathbb{R}^{M \times N}$ are, respectively, the target matrix and the weighted distance between sub-models (defined in Eq. 4). Then calculating the partial derivative of the loss function E_{ij} with respect to a sub-model $q_{i;k}^{(l)}$.

$$\frac{\partial E_{ij}}{\partial q_{i;k}^{(l)}} = 2 f_{ij;k} \cdot h_k^{(l;l+1)} \cdot \frac{v_{ij;k}}{\|v_{ij;k}\|} \tag{25}$$

where $v_{ij;k} = q_{i;k}^{(l)} - q_{j;k}^{(l+1)}$, and the stress force between sub-models under a target $\mathbf{A}$ is denoted by $f_{ij;k} = A_{ij} - \|q_{i;k}^{(l)} - q_{j;k}^{(l+1)}\|$. Since the collective loss function for a sub-model $q_{i;k}^{(l)}$ is E_i, thus,

$$\frac{\partial q_{i;k}^{(l)}}{\partial t} = -\sum_{j=1}^{M} \frac{\partial E_{ij}}{\partial q_{i;k}^{(l)}} = -h_k^{(l;l+1)} \cdot \sum_{j=1}^{M} \frac{v_{ij;k}}{\|v_{ij;k}\|} \cdot f_{ij;k} \tag{26}$$

which is consistent with Eq. 5. Similarly, the update rules for $q_{j;k}^{(l+1)}$ and $h_k^{(l;l+1)}$ are accessible via computing the gradient of the relevant loss function. These facts guarantee that these dynamics-inspired update rules are consistent with the rules derived via computing gradient descent with respect to a specified loss function like back-propagation does. Therefore, we can extend Eq. 5 to a detailed formulation by applying back-propagation on the stress force $f_{ij;k}$, which is replaced with the gradient of loss function with respect to the sub-models rather than the neural weights.

Given a multilayer perceptron that contains two tensor-formed weights $W^{(ih)} \in \mathbb{R}^{N_{in} \times N_h}$ and $W^{(ho)} \in \mathbb{R}^{N_h \times N_{ou}}$. The transmitting signals along with each layer are defined using a sequence: $[S^{(in)}, \sigma(S^{(in)}), S^{(h)}, \sigma(S^{(h)}), S^{(ou)}]$. First, we define a computation-friendly formulation of integral path between two arbitrary sub-models $q_i^{(x)}$ and $q_j^{(y)}$ as follows

$$I_{ij}^{(xy)} = \frac{1}{2}(q_i^{(x)} - q_j^{(y)})^2 = \frac{1}{2} v_{ij}^{(xy)^2} = \frac{1}{2} \sum_{k}^{H} \lambda_k \cdot v_{ij;k}^{(xy)^2} \tag{27}$$

where H is the number of shared coefficients required for converting non-linearity into proper set of linearity. The number of H is discussed in Lemma A.2. Recall that the signals along with each layer are computed as follows

$$\begin{gathered} S_j^{(h)} = \sum_i \sigma(S_i^{(in)}) \cdot I_{ij}^{(in;h)} \\ S_k^{(ou)} = \sum_j \sigma(S_j^{(h)}) \cdot I_{jk}^{(h;ou)} \end{gathered} \tag{28}$$

The loss function is defined by

$$L = \sum_k \frac{1}{2}(S_k^{(ou)} - T_k^{(ou)})^2 = \sum_k \frac{1}{2} \varepsilon_k^2 \tag{29}$$

The resultant signals received by different sub-models are defined by

$$\begin{gathered} \Phi_j^{(xy)} = \sum_i E_{ij}^{(xy)} = \sum_i \sigma(S_i^{(x)}) \cdot v_{ij}^{(xy)} \\ \Phi_i^{(in)} = \sigma(S_i^{(in)}) \\ \Phi_k^{(ou)} = T_k^{(out)} - S_k^{(ou)} \end{gathered} \tag{30}$$

Instead of computing the gradient of loss function with respect to the weights (path integral I_{ij}), we compute the gradients with respect to the dynamic states of sub-models, *e.g.*, $q_k^{(ou)}$

$$\begin{aligned}\frac{\partial q_k^{(ou)}}{\partial t} &= \frac{\partial L}{\partial q_k^{(ou)}} = \sum_j \frac{\partial L}{\partial I_{jk}^{(h;ou)}} \frac{\partial I_{jk}^{(h;ou)}}{\partial q_k^{(ou)}} \\ &= \varepsilon_k \cdot \sum_j \sigma(S_j^{(h)}) \cdot v_{jk}^{(h;ou)} \\ &= \Phi_k^{(ou)} \cdot \Phi_k^{(h;ou)}\end{aligned} \tag{31}$$

Similarly, we compute the gradient of loss function with respect to $q_i^{(in)}$,

$$\begin{aligned}\frac{\partial q_i^{(in)}}{\partial t} &= \frac{\partial L}{\partial q_i^{(in)}} = \sum_j \frac{\partial L}{\partial I_{ij}^{(in;h)}} \frac{\partial I_{ij}^{(in;h)}}{\partial q_i^{(in)}} \\ &= \sum_j \frac{\partial L}{\partial S_k^{(ou)}} \frac{\partial S_k^{(ou)}}{\partial S_j^{(h)}} \frac{\partial S_j^{(h)}}{\partial I_{ij}^{(in;h)}} \frac{\partial I_{ij}^{(in;h)}}{\partial q_i^{(in)}} \\ &= \sum_j \sum_k \varepsilon_k \cdot I_{jk}^{(h;ou)} \cdot \frac{\partial \sigma(S_j^{(h)})}{\partial S_j^{(h)}} \cdot \sigma(S_i^{(in)}) \cdot v_{ij}^{(in;h)} \\ &= \sigma(S_i^{(in)}) \cdot \sum_j v_{ij}^{(in;h)} \cdot \frac{\partial \sigma(S_j^{(h)})}{\partial S_j^{(h)}} \cdot \sum_k \varepsilon_k \cdot I_{jk}^{(h;ou)} \\ &= \Phi_i^{(in)} \cdot \sum_j v_{ji}^{(h;in)} \cdot \sigma(S_j^{(h)}) \cdot (1 - \sigma(S_j^{(h)})) \cdot \Phi_j^{(ou;h)}\end{aligned} \tag{32}$$

In the equilibrium state (meaning that the feedback signal $\Phi_j^{(ou;h)}$ is extremely weak and stable), term $(1 - \sigma(S_j^{(h)})) \cdot \Phi_j^{(ou;h)}$ is degenerated to a specific constant independent of index j, so that Eq. 32 can be approximated as

$$\begin{aligned}\frac{\partial q_i^{(in)}}{\partial t} &\propto \Phi_i^{(in)} \cdot \sum_j v_{ji}^{(h;in)} \cdot \sigma(S_j^{(h)}) \\ &= \Phi_i^{(in)} \cdot \Phi_i^{(h;in)}\end{aligned} \tag{33}$$

this Eq. 33 is obviously consistent with Eq. 6. Now we have the dynamical forms of update rules for sub-models toward a specific loss function. In the other word, we can approximate arbitrary nonlinear function via training the sub-models rather than the neural weights connecting them.

E GENERALIZED RULES OF DYNAMICS IN DyN SYSTEMS

The neuromorphic dynamics are derived from the Hamilton's principle and the Euler-Lagrange equation:

$$\frac{d}{dt} \frac{\partial L_i^{(l,t)}}{\partial \dot{q}_i^{(l,t)}} - \frac{\partial L_i^{(l,t)}}{\partial q_i^{(l,t)}} = 0 \tag{34}$$

where the Lagrangian $L_i^{(l,t)} = S_i^{(l,t)} \cdot \psi_i^{(l,t)}$ measures the energy distribution of signals $S_i^{(l,t)}$ and structural entropy $\psi_i^{(l,t)}$. According to Lagrangian mechanics described in Eq. 34, where the non-relativistic Lagrangian L for sub-models in a specific subsystem is defined by

$$L = T - V = T = \frac{1}{2} m_0 \dot{r}^2 \tag{35}$$

where r represents the dynamic state of a sub-model. Thus

$$\frac{\partial L}{\partial \dot{q}} = \frac{\partial L}{\partial \dot{r}} = m_0 \frac{\partial r}{\partial t} \tag{36}$$

Then substitute Eq. 36 into Eq. 34, obtaining

$$m_0 \frac{\partial r}{\partial t} \frac{\partial^2 r}{\partial t^2} = \frac{\partial L}{\partial t} \tag{37}$$

Summing both sides of Eq. 38

$$\begin{aligned} & m_0 \cdot \sum_{k \neq x} \frac{\partial r_{xk}^{(t)}}{\partial t} \cdot \frac{\partial^2 r_{xk}^{(t)}}{\partial t^2} \\ = & \frac{m_0}{2} \cdot \sum_{k \neq x} \frac{\partial}{\partial t} \left(\frac{\partial r_{xk}^{(t)}}{\partial t} \right)^2 = \sum_{k \neq x} \frac{\partial L_{xk}^{(t)}}{\partial t} \end{aligned} \tag{38}$$

To satisfy the conservation of momentum and Newton's third law derived from Rule a), we have

$$\begin{aligned} \sum_{k \neq x} \left(\frac{\partial r_{xk}^{(t)}}{\partial t} \right)^2 &= \left(\frac{\partial r_{xx}^{(t)}}{\partial t} \right)^2 \\ \sum_{k \neq x} \frac{\partial}{\partial t} \left(\frac{\partial r_{xk}^{(t)}}{\partial t} \right)^2 &= -\frac{\partial}{\partial t} \left(\frac{\partial r_{xx}^{(t)}}{\partial t} \right)^2 \end{aligned} \tag{39}$$

Then the middle term of Eq. 38 can be simplified to

$$\begin{aligned} \frac{m_0}{2} \cdot \sum_{k \neq x} \frac{\partial}{\partial t} \left(\frac{\partial r_{xk}^{(t)}}{\partial t} \right)^2 &= -\frac{m_0}{2} \cdot \frac{\partial}{\partial t} \left(\frac{\partial r_{xx}^{(t)}}{\partial t} \right)^2 \\ &= -m_0 \cdot \frac{\partial r_{xx}^{(t)}}{\partial t} \cdot \frac{\partial^2 r_{xx}^{(t)}}{\partial t^2} \end{aligned} \tag{40}$$

Summing both sides of Eq. 35

$$\sum_{k \neq x} L_{xk}^{(t)} = \frac{m_0}{2} \cdot \sum_{k \neq x} \left(\frac{\partial r_{xk}^{(t)}}{\partial t} \right)^2 \tag{41}$$

Then according to Eq. 39, Eq. 41 can be simplified to

$$\sum_{k \neq x} L_{xk}^{(t)} = \frac{m_0}{2} \cdot \left(\frac{\partial r_{xx}^{(t)}}{\partial t} \right)^2 \tag{42}$$

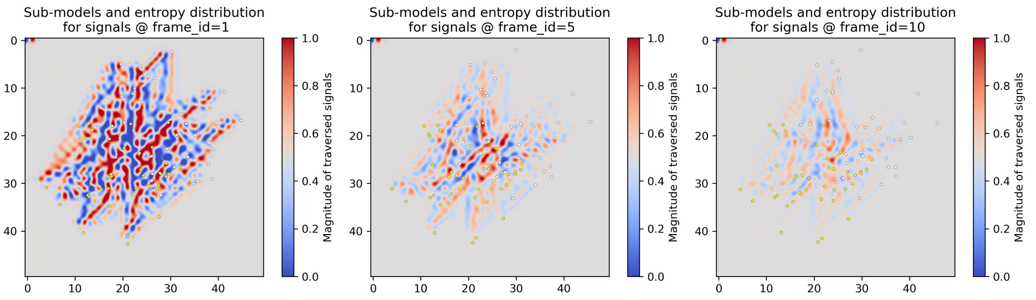
Then we substitute Eq. 42 into Eq. 40 to eliminate m_0, obtaining

$$\begin{aligned} \frac{\partial^2 r_{xx}^{(t)}}{\partial t^2} &= -\frac{1}{2} \cdot \frac{\Sigma_{k \neq x} \frac{\partial}{\partial t} L_{xk}^{(t)}}{\Sigma_{k \neq x} L_{xk}^{(t)}} \cdot \frac{\partial r_{xx}^{(t)}}{\partial t} \\ &= -\frac{\Lambda_x^{(t)}}{2} \cdot \frac{\partial r_{xx}^{(t)}}{\partial t} \end{aligned} \tag{43}$$

Note that L can be approximated as time-invariant when $\partial t \to 0$ since L varies with the combination of all sub-models and signals, whose overall dynamics are relatively static with respect to a particular sub-model. Then we solve the differential equation 43, which yields

$$\frac{\partial r_{xx}^{(t)}}{\partial t} = \eta_x \cdot exp\left(-\frac{\Lambda_x^{(t)}}{2} \cdot t \right) \tag{44}$$

where the entropy indicator $\Lambda_x^{(t)}$ measures the structural entropy (can be evaluated via methods similar to Eq. 9) of L over the system of sub-models, and η_x is a constant value related to its corresponding sub-model $q_x^{(t)}$.



(a) Frame-id=1 (b) Frame-id=5 (c) Frame-id=10

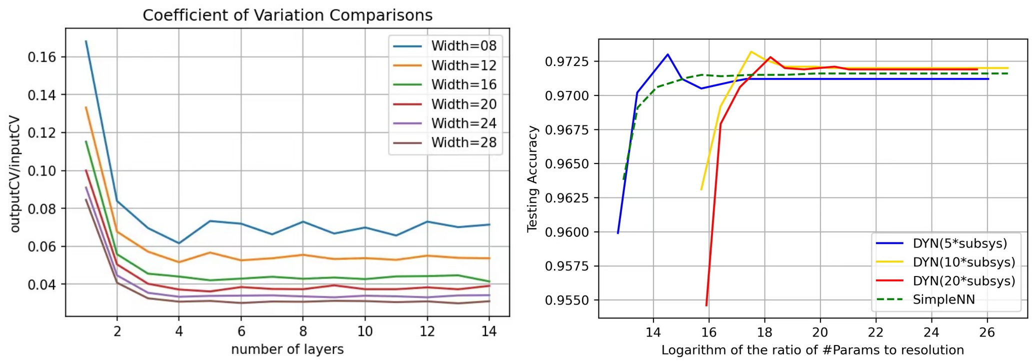
Figure 5: **Equivalence between neuromorphic learning and entropy reduction.** As presented by Eq. 45, the sub-models tend to move toward the region with lower structural entropy, which is visualized by colored spatial distribution.

The comprehensive form of Eq. 44 is as follows:

$$\frac{\partial r_i^{(L,t)}}{\partial t} = \eta_i \cdot exp\left(- \frac{\Sigma_{k \neq i} \frac{\partial}{\partial t} L_{ik}^{(L,t)}}{\Sigma_{k \neq i} L_{ik}^{(L,t)}} \cdot t \right) = \eta_i \cdot \exp\left(- \Lambda_i^{(L,t)} \cdot t \right) \tag{45}$$

where $r_i^{(L,t)}$ is the positional vector of a sub-model, and η_i is a constant related to $q_i^{(L,t)}$. This equation is equipped with an unspecific Lagrangian L, for instance, $L_{ik}^{(L,t)} = S_{ik}^{(L,t)} \Phi_i^{(L,t)}$, where $S_{ik}^{(L,t)}$ is emitted from $q_i^{(L,t^*)}$ and received by $q_k^{(L,t)}$, being influenced by the resultant potential field $\Phi_i^{(L,t)}$ around $q_i^{(L,t)}$. The signals $S_{ik}^{(L,t)}$ refers to the feedback control correlated with the loss function for current task, and the potential field $\Phi_i^{(L,t)}$ is a trainable parameter related to distinct sub-models; note that we can simplify $\Phi_i^{(L,t)}$ as a constant field by adding shared coefficients applied in Eq. 4.

F Conservation of workload and computational complexity as the number of sub-models increases



(a) (b)

Figure 6: **a.** The ratio of output C_v to input C_v as the number of layers increases with different number of sub-models in the hidden layer (width). **b.** The horizontal axis measures the logarithm of the ratio of parameters' size and resolution (LRPR); the vertical axis measures the testing accuracy of the truncated models with W_{tr} corresponding to each specific LRPR.

By Rule a), the variables in Eq. 43 and Eq. 44 should have several restrictions, including

$$\begin{gathered}\sum_x \eta_x = C_\eta \\ \sum_x \sum_{k \neq x} L_{xk}^{(t)} = C_L \\ \sum_x \sum_{k \neq x} \frac{\partial}{\partial t} L_{xk}^{(t)} = C_{\partial L}\end{gathered} \tag{46}$$

where C_η, C_L and $C_{\partial L}$ are time-invariant constants. Therefore, the summation of the entropy indicator $\Lambda_x^{(t)}$ can also be approximated as a time-invariant constant

$$\lim_{N \to \infty} \sum_{x=1}^{N} \Lambda_x^{(t)} \approx \frac{C_{\partial L}}{C_L} = N \cdot \bar{\Lambda} \tag{47}$$

where N is defined as the total number of sub-models, and $\bar{\Lambda}$ is a defined to be a constant referring to the averaged entropy indicator. Therefore, the total path length of all sub-models can be approximated in terms of Eq. 43 as a time-variant function $I(t)$

$$I(t) = \sum_{x=1}^{N} \frac{\partial r_x^{(t)}}{\partial t} = \sum_{x=1}^{N} \eta_x \cdot \exp\left(-\Lambda_i^{(t)} \cdot t\right) \approx C_\eta \cdot \exp\left(-\bar{\Lambda} \cdot t\right) \tag{48}$$

The time-step t is a small value since the sub-models of a DyN system generate signals almost instantaneously. The total workload $W(T)$ that is linearly correlated with the computational complexity is evaluated

$$W(T) = \int_0^T I(t) dt = \frac{N C_\eta C_L}{C_{\partial L}} \cdot (1 - e^{-\bar{\Lambda} T}) \tag{49}$$

where T is the total time cost required to reach an equilibrium state. Based on Eq. 49, as the required number of sub-models increases largely to deal with an increasingly complicated computational task, the total workload and the computational complexity does not increase accordingly, but gradually approaches a specific value. This fact also implies a neuro-biological correlation that when the brain arises a concept, the power of the cortical regions related to the concept remains unchanged after several learning events that supply more specified knowledge on this concept.

G Learning with more subsystems and more sub-models

To boost up the computational power of a DyN system without explosive growth of computational complexity, we apply interactive mechanism on the current architecture. The principle of hierarchical structures (Eq. 23) implies that a well-formed neuromorphic system does not require a specified set of discrete trainable units that are isolated from each other. Besides, inspired by the phase space density representation (Abarbanel & Rouhi, 1987), a dynamic system represented by infinite interactive particles can be treated as a linear combination of many shallow layers, each of which is interpreted as an isolated dynamic system of different density. Specifically, each layer of density ρ_i refers to a subsystem with specific shared coefficient h_i, which disassembles the overall neuromorphic system into several partially independent subsystems. Therefore, a discrete sub-model $q_{i;k_i}^{(l;t)}$ is equivalent to an interactive region with density of h_{k_i}; the variational density is denoted by $g_{ij} = \rho_{k_i} - \rho_{k_j} = h_{k_i} - h_{k_j}$, which measures the potential energy generated by the interaction (receiving or emitting signals) between sub-models.

$$m_i \frac{\partial}{\partial t} q_{i;k_i}^{(l,t)} = \int_{q_{j;k_j}^{(l,t)} \in P^{(l)}} \psi(v_{ij}^{(l,t)}, f_{ij}^{(l,t)}, g_{ij}^{(l,t)}) dP^{(l)} \tag{50}$$

where m_i is a constant related to the density h_{k_i}, and ψ is a linear transformation that concatenates the variational dynamics of sub-models. The notations here are consistent with that of Eq. 5. Therefore, a DyN system with infinite sub-models can be approximated as the one with finite subsystems in which the dynamic states of sub-models are interactively correlated with other sub-models from all subsystems. The increase of the number of computing units has lead to some burden in the software implementation process, it leads to no increase in the overall computational complexity. This fact is validated experimentally and mathematically (Appendix F).

H Algorithms with pseudo-codes

Algorithm 1 Update sub-models based on stress force

```
Require: $H, M, N, D > 0$
Ensure: $Q_{in} \in \mathbb{R}^{H\times M\times D}$, $Q_{ou} \in \mathbb{R}^{H\times N\times D}$, $h \in \mathbb{R}^H$, $M_{st} \in \mathbb{R}^{M\times N}$, $S_{thres} \in \mathbb{R}^+$
  function UPDATEQ($Q_{in}$, $Q_{ou}$, $h$, $M_{st}$, $S_{thres}$)
      $V \in \mathbb{R}^{H\times M\times N\times D} \leftarrow rV(Q_{in}, Q_{ou})$    ▷ relative vectors between each $Q_{in}$ and $Q_{ou}$
      $D \in \mathbb{R}^{M\times N} \leftarrow \|h \cdot V\|$    ▷ distance matrix between $Q_{in}$ and $Q_{ou}$
      $S \in \mathbb{R}^{M\times N} \leftarrow M_{st} - D$    ▷ stress force between target matrix and distance matrix
      while $\sum_{i,j} S_{i,j} > MN \cdot S_{thres}$ do
          $\Delta \in \mathbb{R}^{H\times M\times N\times D} \leftarrow h \cdot V \cdot S$
          $Q_{in} \leftarrow Q_{in} - \sum_{k=1}^{H} h_k \cdot \sum_{i=1}^{N} \Delta_{:,:,i,:}$
          $Q_{ou} \leftarrow Q_{ou} + \sum_{k=1}^{H} h_k \cdot \sum_{j=1}^{M} \Delta_{:,j,:,:}$
          $h \leftarrow \sum_{i,j,k} \{V \cdot S\}_{:,i,j,k}$    ▷ update $Q_{in}$, $Q_{ou}$, and $h$ via Eq. 5
          $S \leftarrow M_{st} - \|h \cdot rV(Q_{in}, Q_{ou})\|$    ▷ update current stress force
      end while
  end function
```

Algorithm 2 Forward an ANN via dynamics of sub-models

```
Require: $L > 0, H > 0, M > 0, N > 0, D > 0$
Ensure: $\lambda \in \mathbb{R}^{L\times H}$, $Q_0 \in \mathbb{R}^{H\times M\times D}$, $Q_L \in \mathbb{R}^{H\times N\times D}$, $\Phi^{(in)} \in \mathbb{R}^M$
  function FORWARDDYN($[Q_l, l \in [0, L]]$,$\lambda$,$\Phi^{(in)}$)
      $S \leftarrow \Phi^{(in)}$    ▷ initialize $S$ as input signal
      for $l \in (0, L)$ do
          $S \leftarrow \sigma(S \cdot \|\lambda_l \cdot rV(Q_l, Q_{l+1})\|)$    ▷ transmitting signal with relations between $Q$s
      end for
      return $S$
  end function
```

Algorithm 3 Train an ANN via dynamics of sub-models

```
Require: $L > 0, H > 0, M > 0, N > 0, D > 0$
Ensure: $\lambda \in \mathbb{R}^{L\times H}$, $Q_0 \in \mathbb{R}^{H\times M\times D}$, $Q_L \in \mathbb{R}^{H\times N\times D}$, $\Phi^{(in)} \in \mathbb{R}^M$, $T^{(ou)} \in \mathbb{R}^M$
  function LEARNDYN($[Q_l, l \in [0, L]]$,$\lambda$,$\Phi^{(in)}$,$T^{(ou)}$)
      $\Phi^{(ou)} \leftarrow$ FORWARDDYN($[Q_l, l \in [0, L]], \lambda, \Phi^{(in)}$)
      $\Phi^{(ou)} \leftarrow \Phi^{(ou)} - T^{(ou)}$
      for $l \in (0, L)$ do
          $I^{(l;l+1)} \leftarrow \|\lambda_l \cdot rV(Q_l, Q_{l+1})\|$    ▷ evaluate relative vectors amongst $Q$s as hidden states
          UPDATEQ($Q_l, Q_{l+1}, \frac{\partial \Phi^{(ou)}}{\partial I^{(l;l+1)}}$)
      end for
  end function
```

Algorithm 4 Train a Convolutional layer (filter size $F \times F$ with N_{in} input channels and N_{ou} output channels) via dynamics of sub-models

```
Require: $F > 0, D > 0, H > 0, N_{in} > 0, N_{ou} > 0, N > 0$
Ensure: $P^{(in)} \in \mathbb{R}^{F\times F\times H\times N_{in}\times D}$, $P^{(ou)} \in \mathbb{R}^{F\times F\times H\times N_{ou}\times D}$, $\Phi^{(in)} \in \mathbb{R}^{N^2\times N_{in}}$, $\lambda \in \mathbb{R}^{F^2\times H}$,
    $T^{(ou)} \in \mathbb{R}^{N^2\times N_{ou}}$
  function LearnConv($P^{(in)}$,$P^{(ou)}$,$\Phi^{(in)}$,$T^{(ou)}$,$\lambda$)
      $C \in \mathbb{R}^{F\times F\times N_{ou}\times N_{in}}$
      $C[i,j,:,:] \leftarrow \|\lambda \cdot rV(P^{(in)}_{ij}, P^{(ou)}_{ij})\|$   ▷ Initialize $\mathbb{R}^{N_{in}} \to \mathbb{R}^{N_{ou}}$ for each element of a filter
      $\Phi^{(ou)} \in \mathbb{R}^{N^2\times N_{ou}} \leftarrow Conv(\Phi^{(in)}, C)$   ▷ output signal via a convolutional layer
      $\Phi^{(ou)} \leftarrow \Phi^{(ou)} - T^{(ou)}$
      for $i, j \in [1, F]$ do
          UpdateQ($P^{(in)}_{ij}, P^{(ou)}_{ij}, \frac{\partial \Phi^{(ou)}}{\partial C[i,j,:,:]}$)
      end for
  end function
```

Algorithm 5 Train an Attention layer ($W_Q, W_K, W_V \in \mathbb{R}^{N\times N}$) via dynamics of sub-models

```
Require: $N > 0, D > 0, H > 0$
Ensure: $P^{(Qin)}, P^{(Kin)}, P^{(Vin)}, P^{(ou)} \in \mathbb{R}^{H\times N\times D}$, $\lambda \in \mathbb{R}^{3\times H}$, $T^{(ou)} \in \mathbb{R}^{N\times N}$
  function LearnAttn($P^{(Qin)}$,$P^{(Kin)}$,$P^{(Vin)}$,$P^{(ou)}$,$\Phi^{(in)}$,$T^{(ou)}$,$\lambda$)
      $A_Q \leftarrow \|\lambda[0] \cdot rV(P^{(Qin)}, P^{(ou)})\|$   ▷ initialize the Query matrix
      $A_K \leftarrow \|\lambda[1] \cdot rV(P^{(Kin)}, P^{(ou)})\|$   ▷ initialize the Key matrix
      $A_V \leftarrow \|\lambda[2] \cdot rV(P^{(Vin)}, P^{(ou)})\|$   ▷ initialize the Value matrix
      $\Phi^{(ou)} \leftarrow Attn(A_Q, A_K, A_V, \Phi^{(in)})$
      $\Phi^{(ou)} \leftarrow \Phi^{(ou)} - T^{(ou)}$
      UpdateQ($P^{(Qin)}, P^{(ou)}, \frac{\partial \Phi^{(ou)}}{\partial A_Q}$)
      UpdateQ($P^{(Kin)}, P^{(ou)}, \frac{\partial \Phi^{(ou)}}{\partial A_K}$)
      UpdateQ($P^{(Vin)}, P^{(ou)}, \frac{\partial \Phi^{(ou)}}{\partial A_V}$)
  end function
```