Data-free Neural Representation Compression with Riemannian Neural Dynamics

Zhengqi Pei [1,2], Anran Zhang [1,2], Shuhui Wang *[1,3], Xiangyang Ji [4], Qingming Huang [5,1,3]

[1] Institute of Computing Technology, Chinese Academy of Sciences
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences
[3] Peng Cheng Laboratory
[4] Department of Automation, Tsinghua University
[5] School of Computer Science and Technology, University of Chinese Academy of Sciences

* Corresponding Author

ICML 2024 Oral Presentation (Poster Location: Hall C 4-9 #1902)

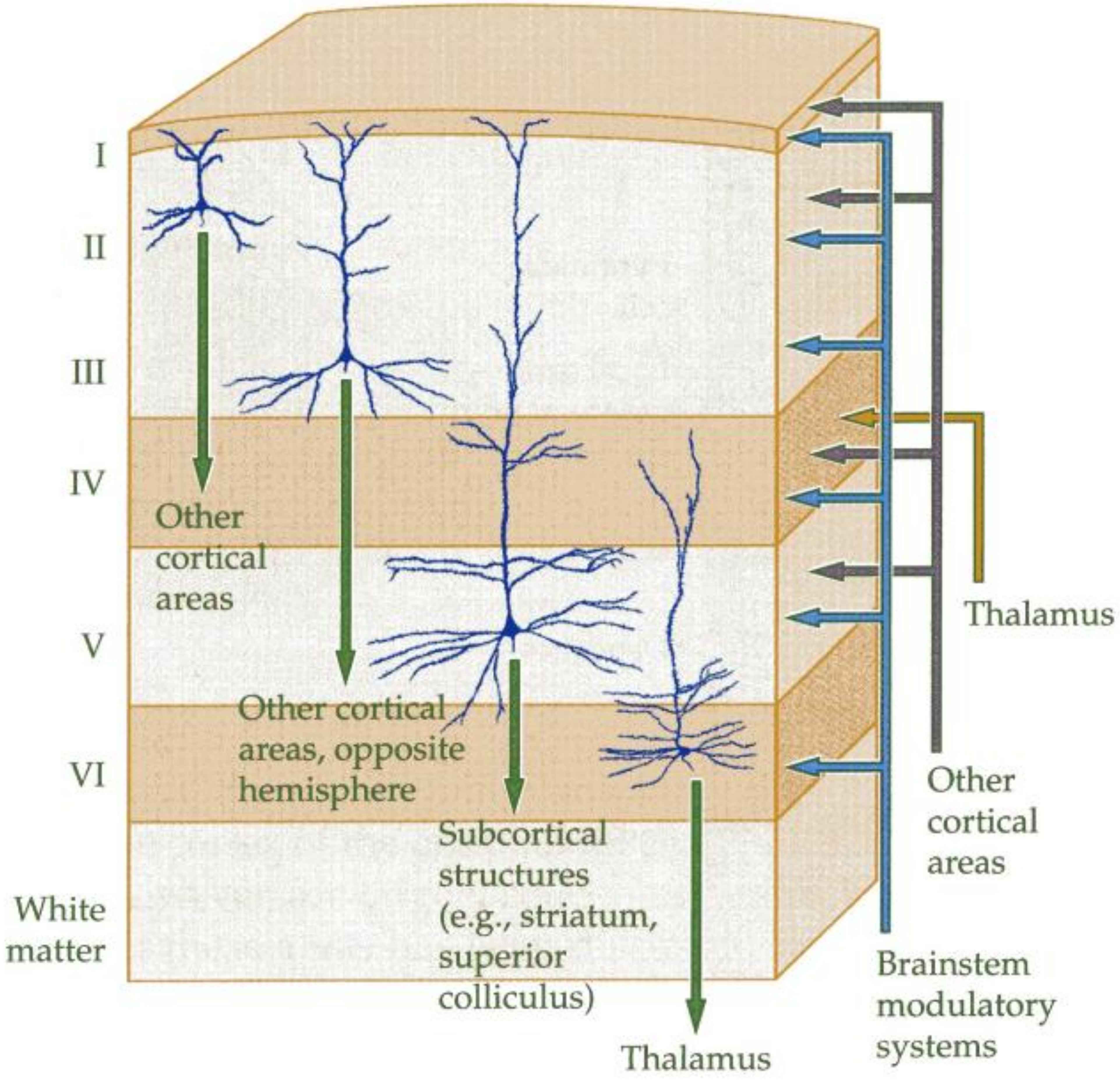
- The human brain is better described as a **Flat neural model**, according to the ***Shallow Brain Hypothesis*** [1].

- **Flat models** have

  - **Higher computational efficiency**: parallel computing;

  - **Stronger interpretability**: cortical partitioning;

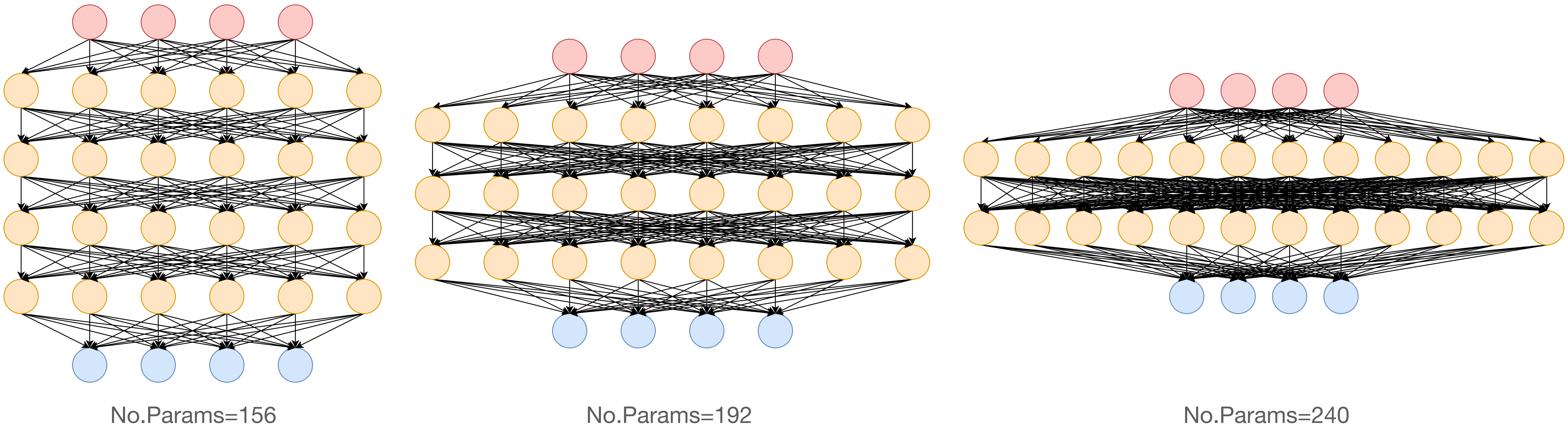
  - **Better suitability for certain tasks**: learning is easier for smaller dataset;



[1] ***How deep is the brain? The shallow brain hypothesis*.** Mototaka Suzuki, et al. **Nature Reviews 2023**

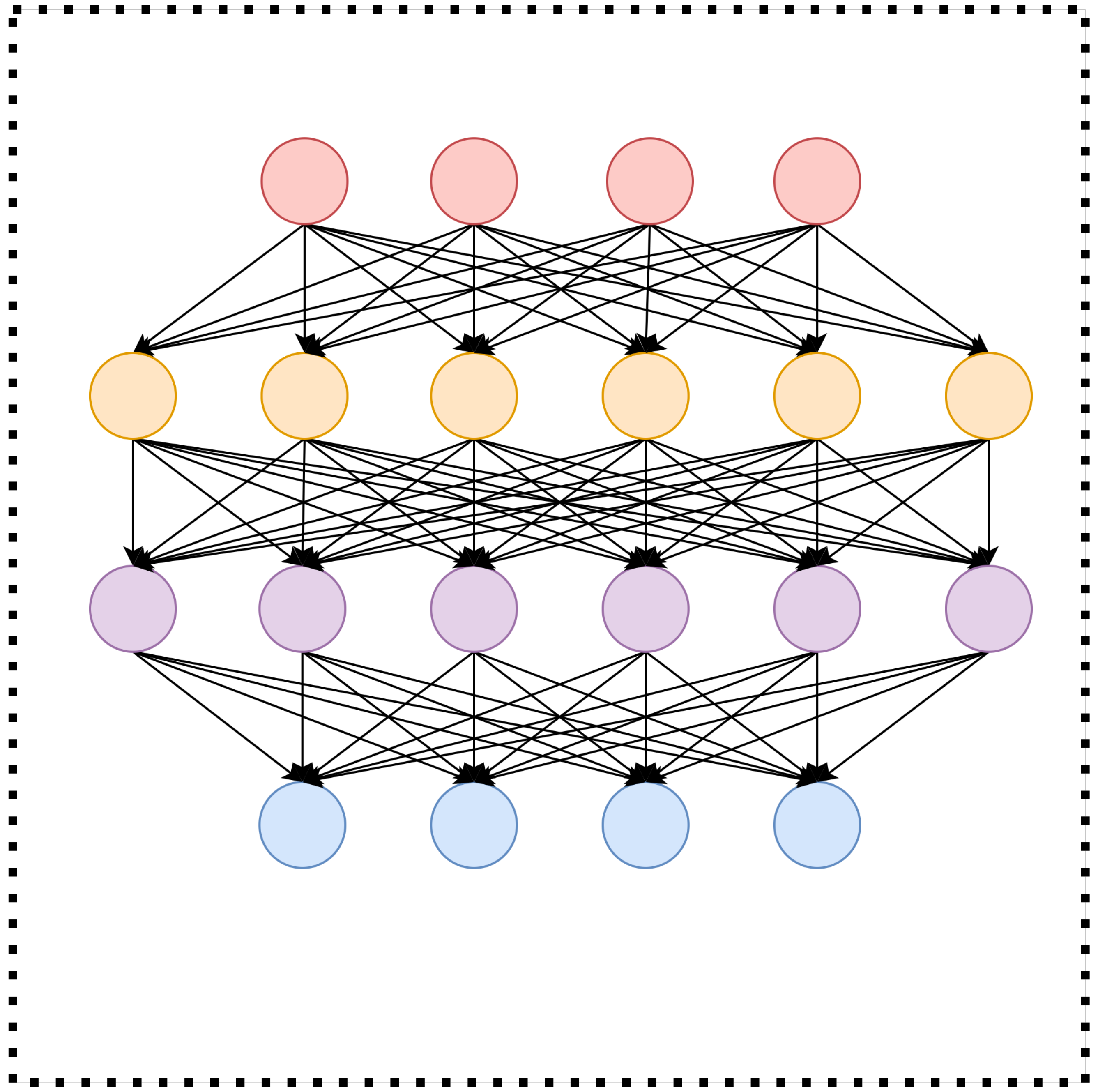
However …

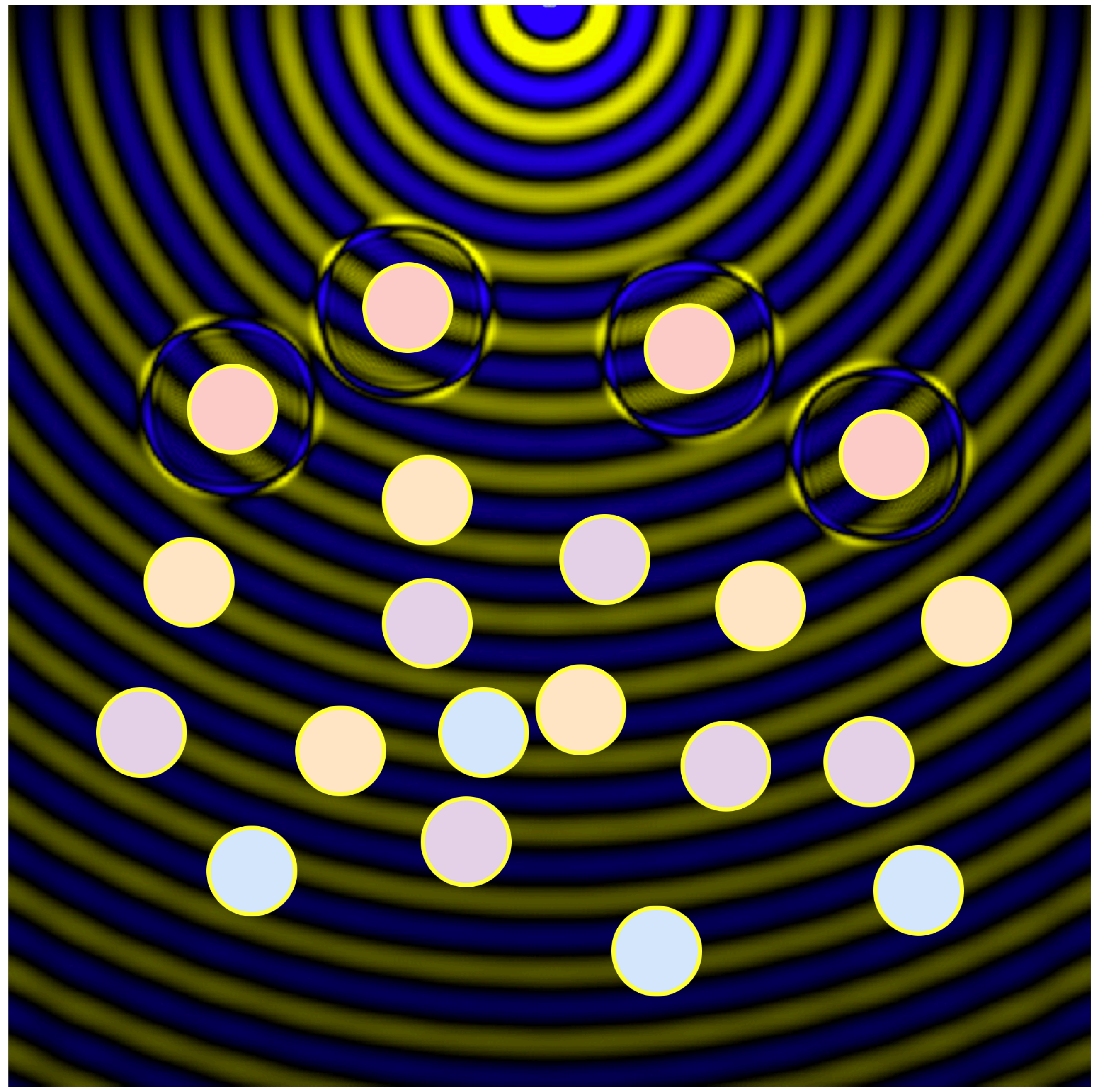
A Flatter neural model refers to an increasing No.Params and Complexity, the trainable neural weights becomes increasingly toublesome and "ugly."

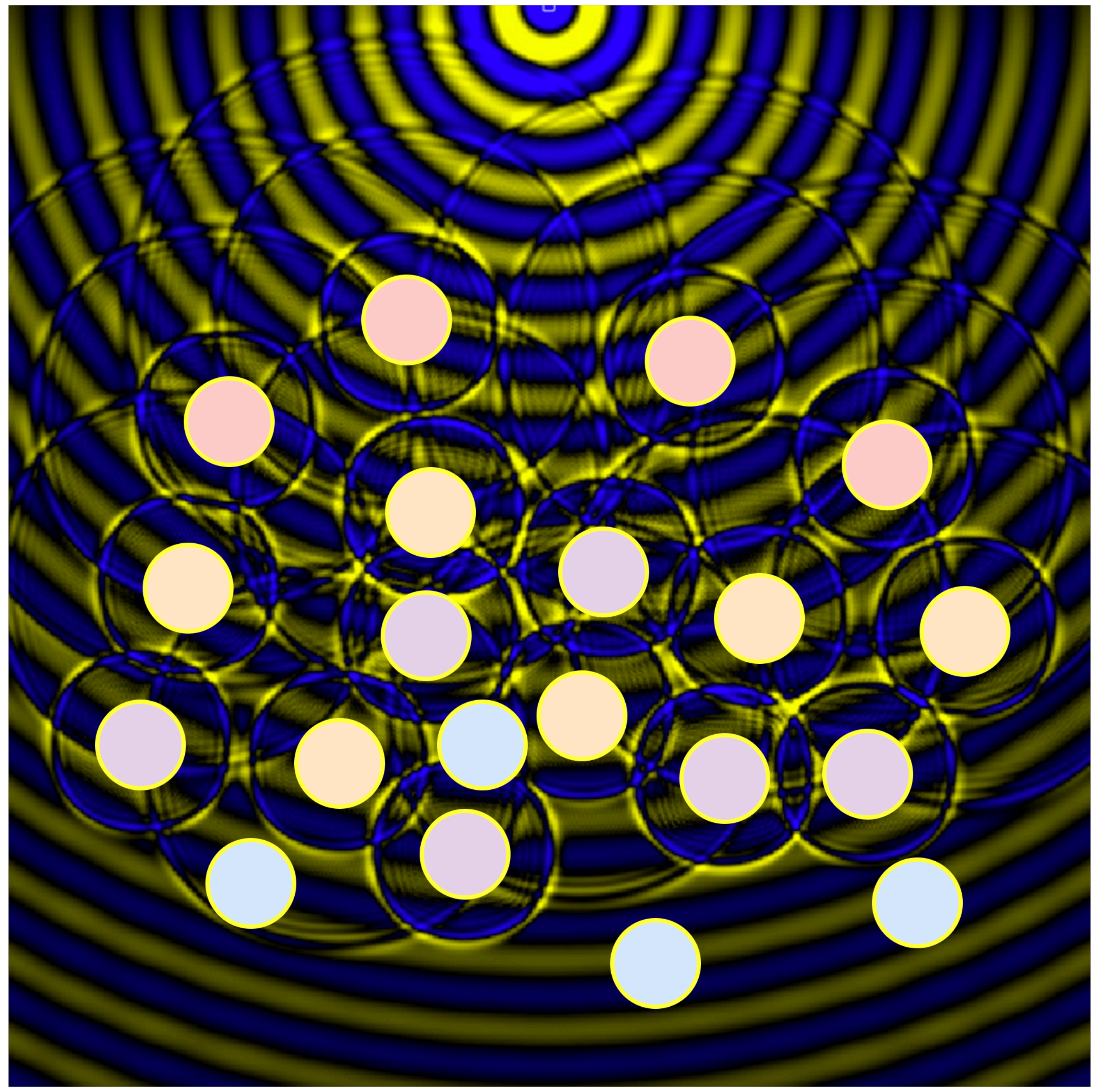


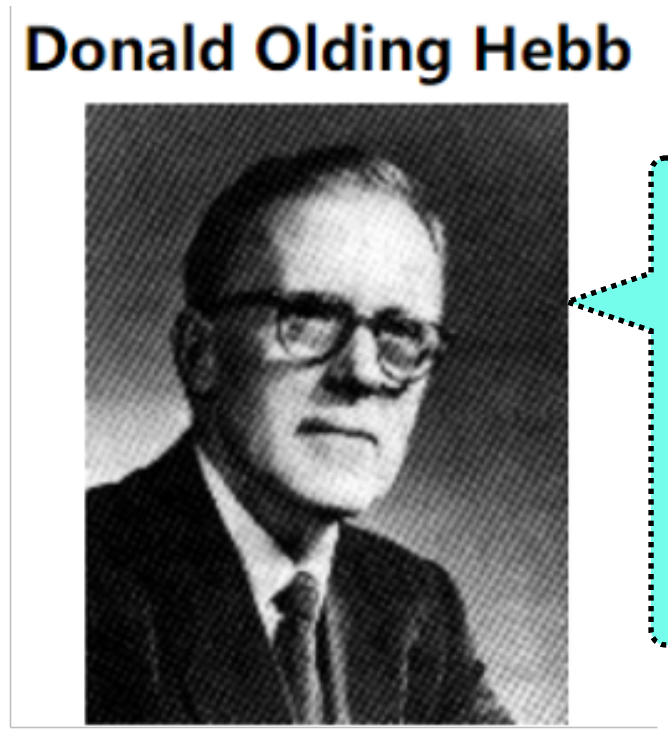
ANN and BioNN's development paths are not align…

Intuitively, if we replace those neural weights with local and global **Neuronal fields**, everything becomes “prettier" and better.









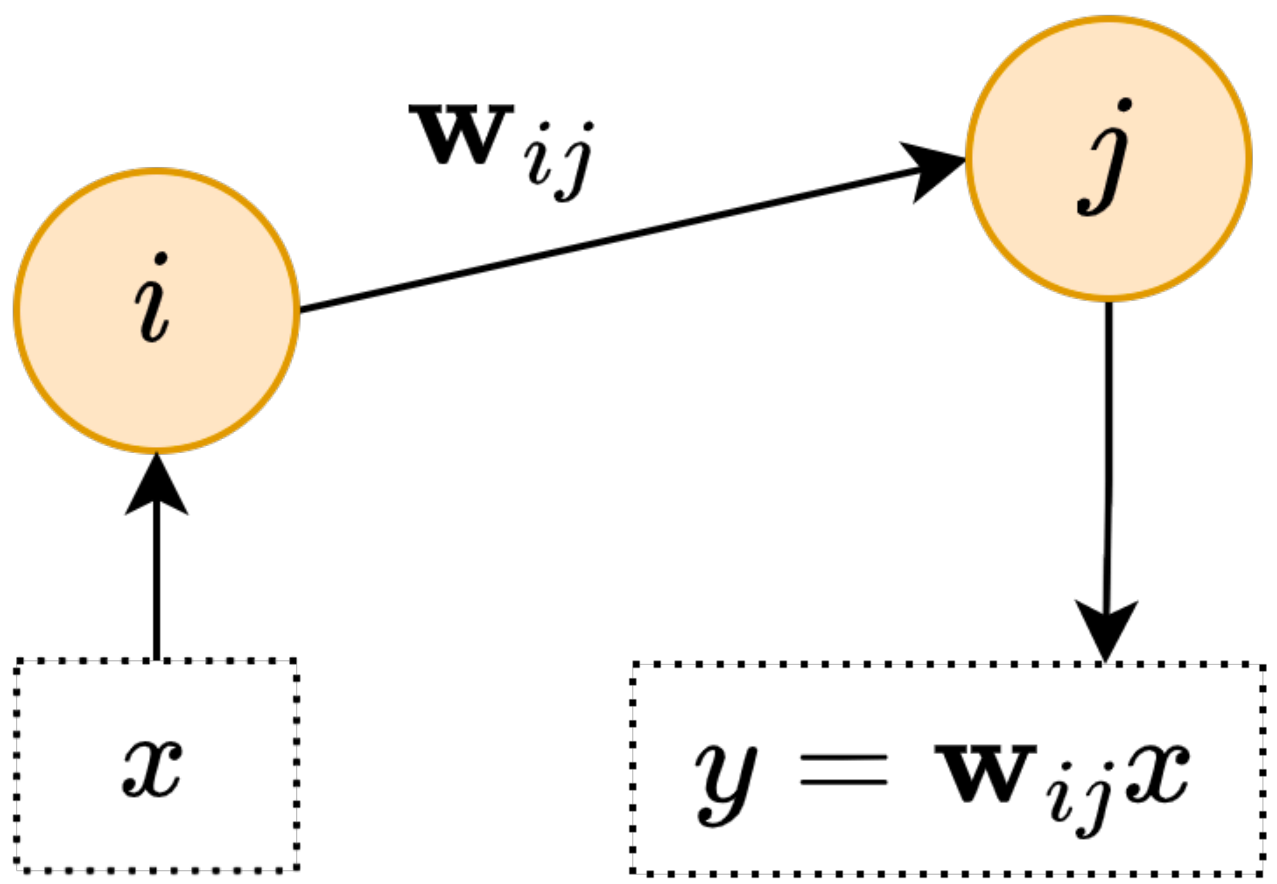
Hebb's Rule (1949) describes the principle of synaptic plasticity: an increase in synaptic efficacy arises from a presynaptic cell's repeated and persistent stimulation of a postsynaptic cell.

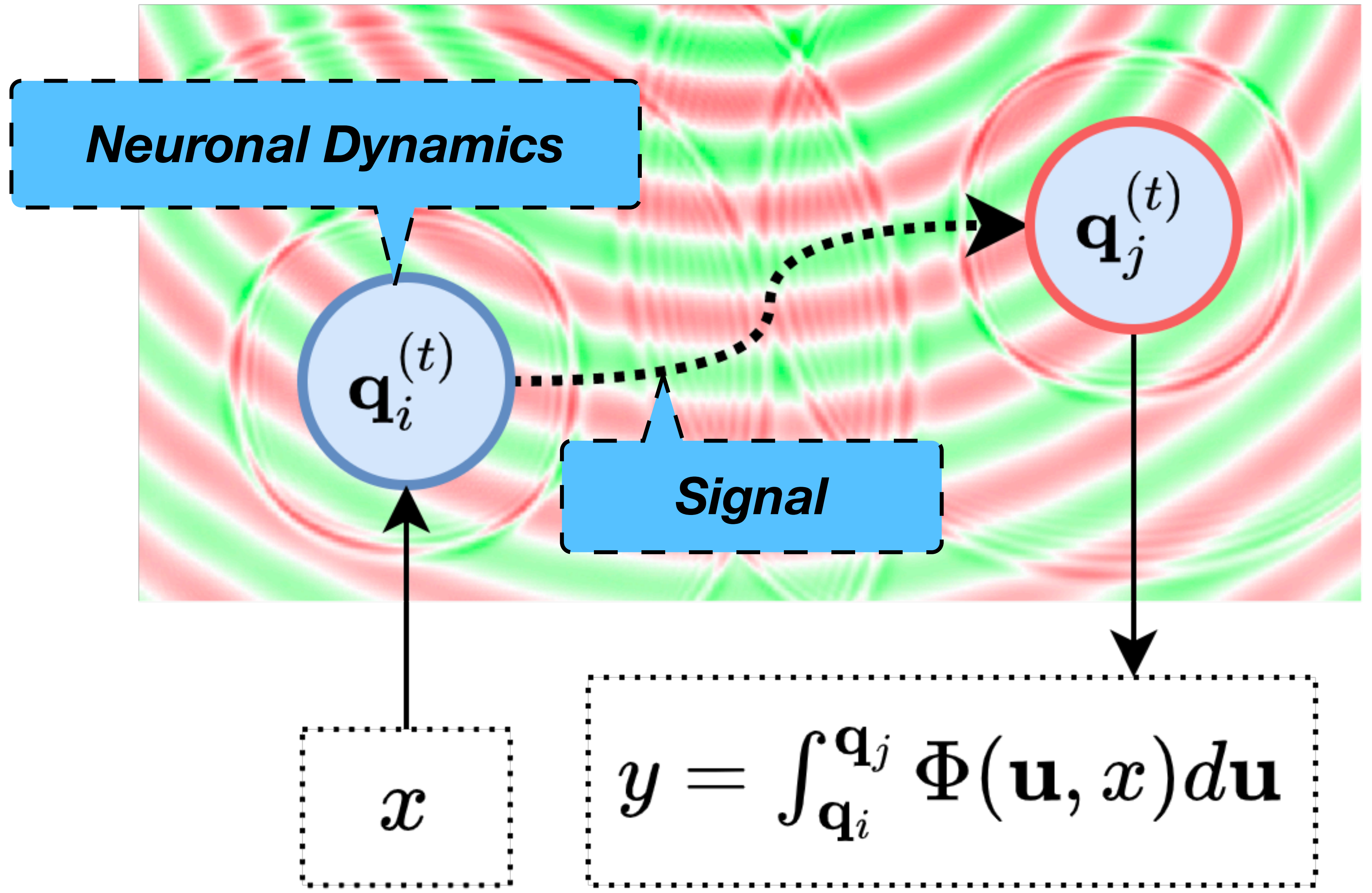
Feedforward through neural layers refers to Wave-Propagation amongst neuronal field

Mathematically …

A **Neuronal field** $\Phi : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ refers to:

- First, embedding each neuron into a **d-dimensional manifold**
- Each neuron corresponds to a d-dimensional vector $q_i^{(t)} \in \mathbb{R}^d$
- Then, interpreting $\mathbf{w}_{ij}x$ as the process of **signal transmission** between neurons.





How to simulate the signal transmission within Neuronal Fields

- Solution[2]: we design a ruler, i.e., a **metric function** defined via piecewise linearities to measure the ***dynamical relations*** between neurons

$$y = \int_{\mathbf{q}_i}^{\mathbf{q}_j} \Phi(\mathbf{u}, x) d\mathbf{u} = \mu(\mathbf{q}_i, \mathbf{q}_j) \cdot x$$

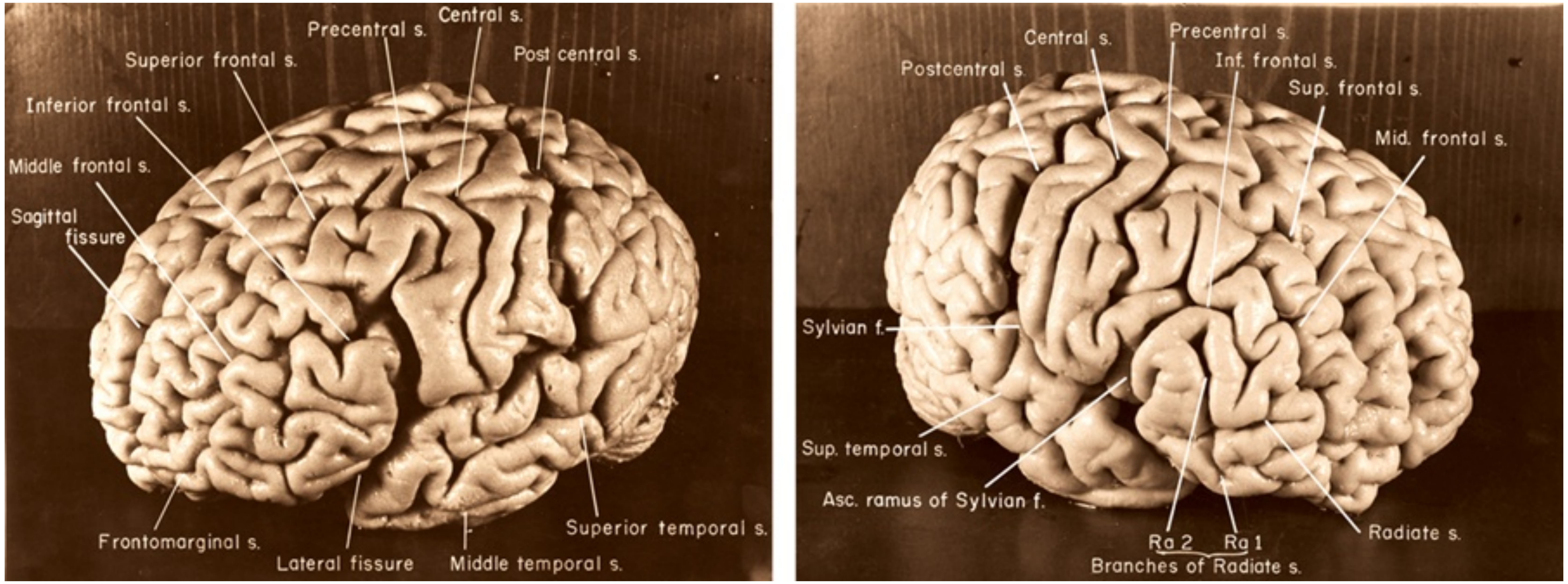
$$\mu(\mathbf{q}_i, \mathbf{q}_j) = \sum_{h=1}^{H} \lambda_h \cdot \left\| \mathbf{q}_i\left[\frac{dh-d}{H} : \frac{dh}{H}\right] - \mathbf{q}_j\left[\frac{dh-d}{H} : \frac{dh}{H}\right] \right\|_p$$

where $\lambda \in \mathbb{R}$ are trainable coefficients, and $H \in \mathbb{N}^+$ are the number of linearities required.

- Then, a neural layer with m input and n output neurons, which requires $m \times n$ trainable parameters, now only needs $d \times (m + n)$ trainable parameters.

[2] ***Dynamics-inspired Neuromorphic Visual Representation Learning***. Z. Pei, S. Wang. *CAS-ICT*. **ICML 2023**.

- However, this **Euclidean** piecewise linearity is overly simplistic to capture the complexity of the neuronal dynamics as in the human brain.
- Ideally, we should upgrade the Euclidean neuronal state space to a ***Riemannian*** one, which is tailored for curved surfaces, much like the convoluted surfaces of the human cerebral cortex.



The Curvature in the Riemannian neuronal state space surface appears to be more significant than the Depth of the neural structure.

Einstein's Brain photographed by Thomas Harvey at Princeton Hospital in 1955

Why use a Riemannian metric? Because…

- Unlike Euclidean metrics, a Riemannian metric can handle the relationships **between different dimensions** of encodings.

Formally, a Riemannian metric g on a smooth manifolds $\mathscr{M}$ is an inner product $g : T_x\mathscr{M} \times T_x\mathscr{M} \mapsto \mathbb{R}$ on each tangent space $T_x\mathscr{M}$ of $\mathscr{M}$ for each $x \in \mathscr{M}$ and

$$g = \sum_{i,j} g_{ij}\, dx[i] \otimes dx[j]$$

where $\otimes$ is the tensor product that combines two tensors to generate a larger tensor.

In the human brain, the neuronal components interact with each other in a cross-dimensional manner, rather than in a one-to-one fashion.

- These cross-dimensional interaction generates a non-Euclidean neuronal space based on curved surfaces

- However, a Riemannian metric requires $O(d^2)$, how to simplify it…?
- ***Our Solution:***

Step 1: design a d-dimensional ***displacement vector***, i.e., a skipping leapfrog, to define the inter-dimensional relation between neurons.

Step 2: obtain an intermediate ***metric vector*** to interpret the neuronal dynamical relation in the Riemannian metric space.

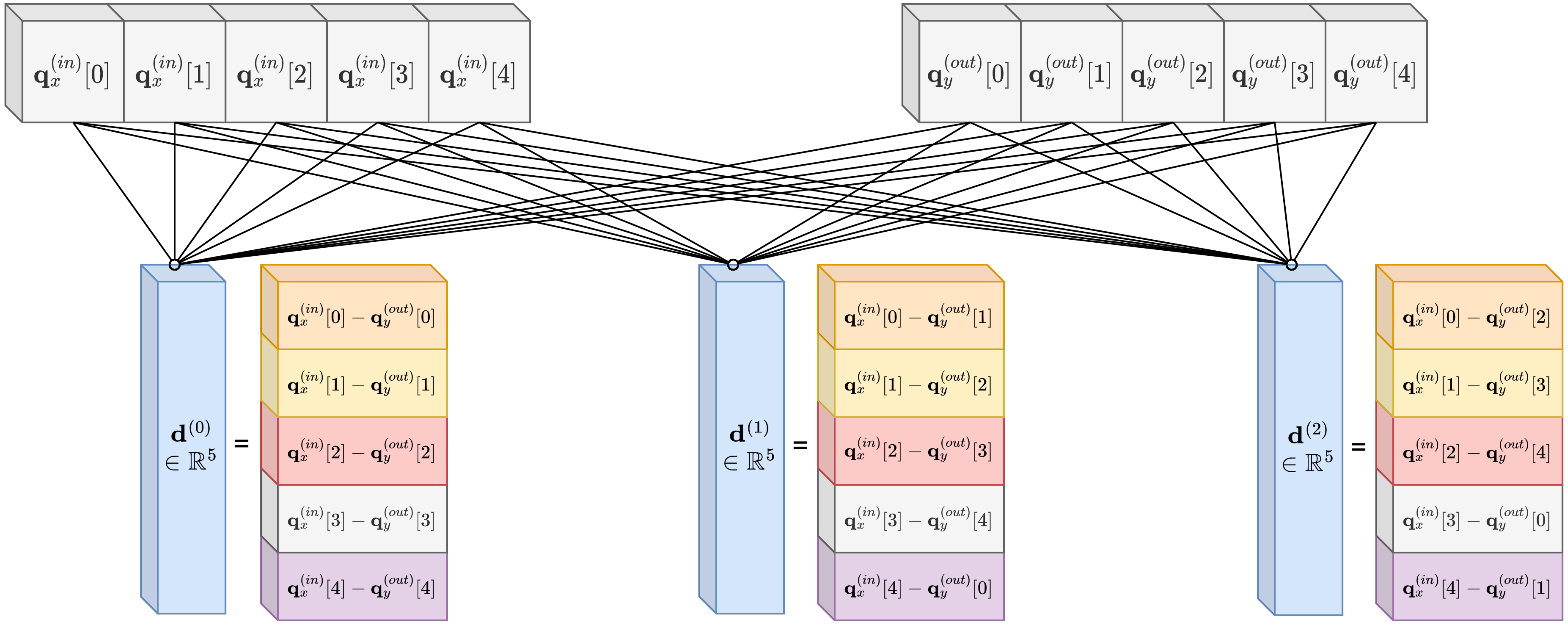
Step 3: compute the final result via a trainable ***linear projection*** that maps the metric vector to the Euclidean scalar space.

Step 1: construct displacement vectors with different step sizes

A displacement vector $\mathbf{d}_{xy}^{(s)} \in \mathbb{R}^d$ defines the inter-dimensional relation between neurons

$$\mathbf{d}_{xy}^{(s)}[i] = \mathbf{q}_x[i] - \mathbf{q}_y[i+s]\ , \quad s \in \mathcal{S}$$

where $\mathcal{S} \subseteq \{0,\ 1...\ d\}$ are pre-defined displacement steps, e.g., $\mathcal{S} = \{0,\ 1,\ 2\}$ or $\mathcal{S} = \{1,\ 3,\ 5\}$

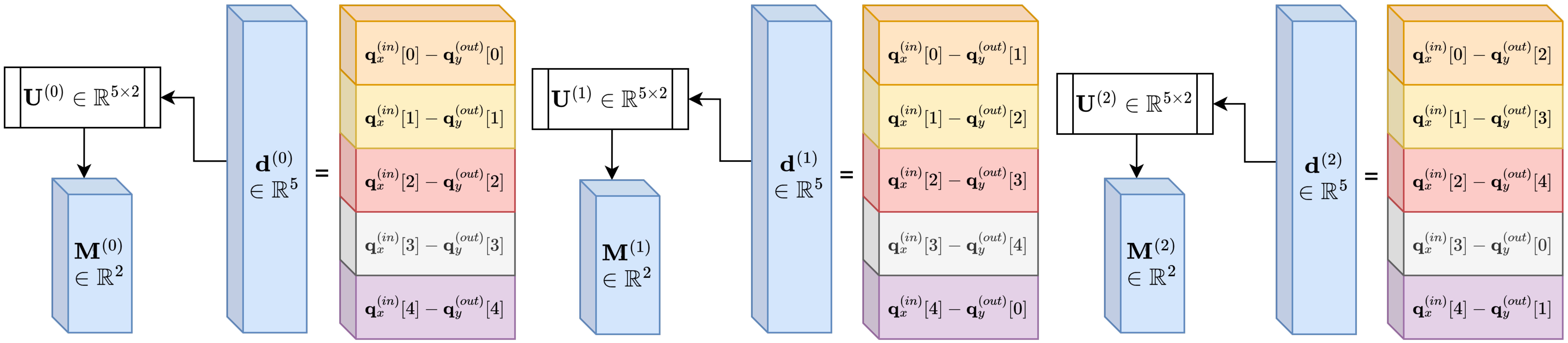


Step 2: map to the metric space

The intermediate metric vectors between neurons x and y defined as

$$\mathbf{M}_{xy}^{(s)} = \mathbf{d}_{xy}^{(s)}\mathbf{U}^{(s)} \in \mathbb{R}^{d_\mu}$$

where $\mathbf{U}^{(s)} \in \mathbb{R}^{d \times d_\mu}$ is a trainable projection, and $d_\mu \in \mathbb{N}^+ < d$ is the pre-defined dimension of the Riemannian metric space.

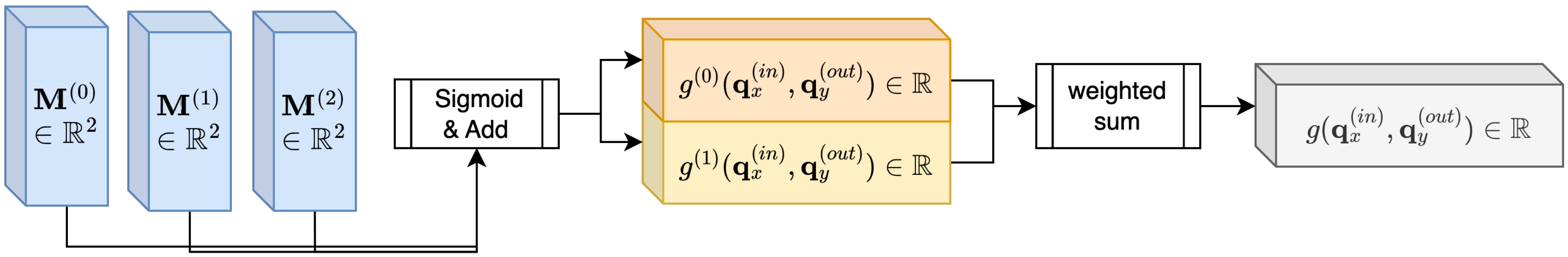


Step 3: weight the dimensions to obtain final value

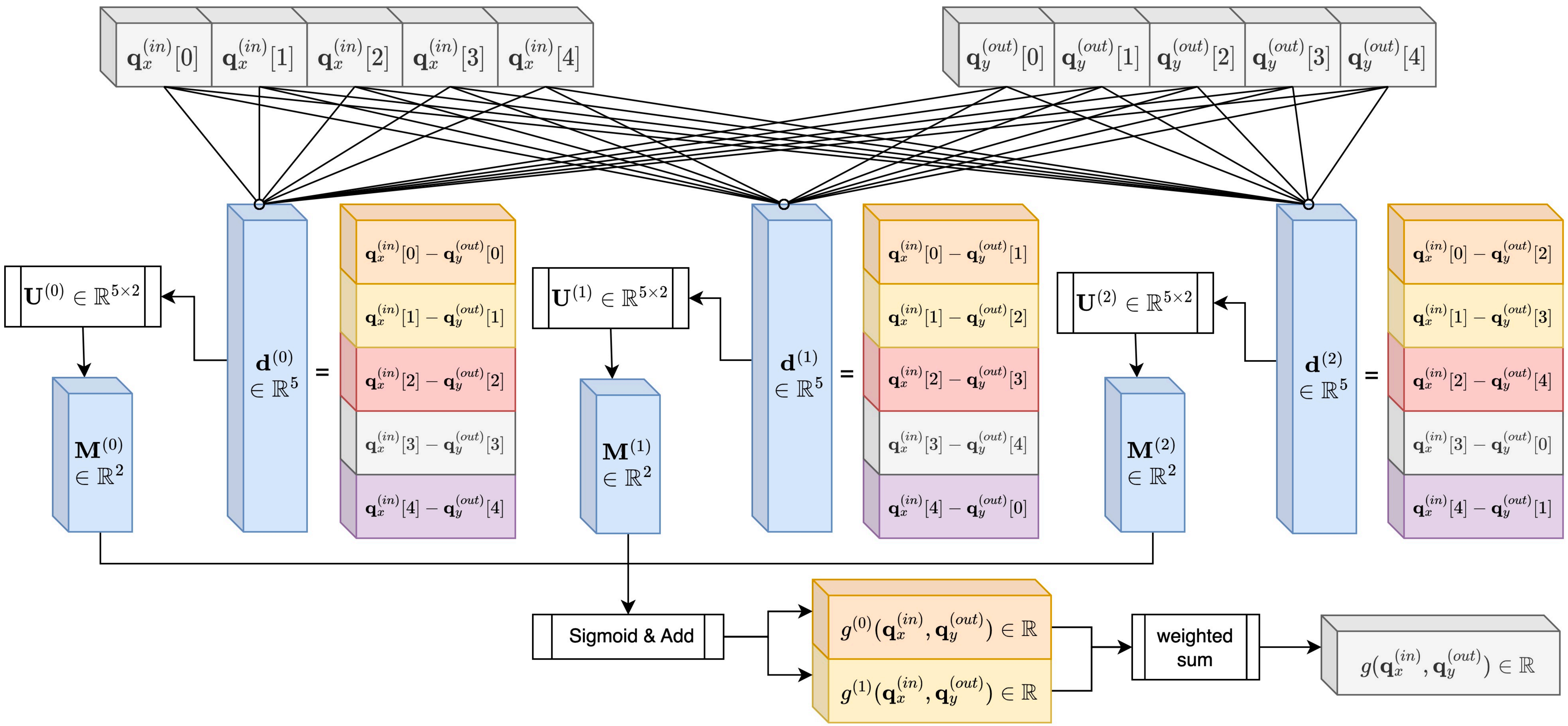
Add the activated metric vectors and sum the components via a trainable linear projection to obtain the final result

$$g(\mathbf{q}_x, \mathbf{q}_y) = \sum_{\alpha=0}^{d_\mu} \rho_\alpha \cdot \Big(\sum_{s \in \mathcal{S}} \mathbf{M}_{xy}^{(s)} \Big)[\alpha]$$

where $\rho \in \mathbb{R}^{d_\mu}$ refers to the trainable projection.



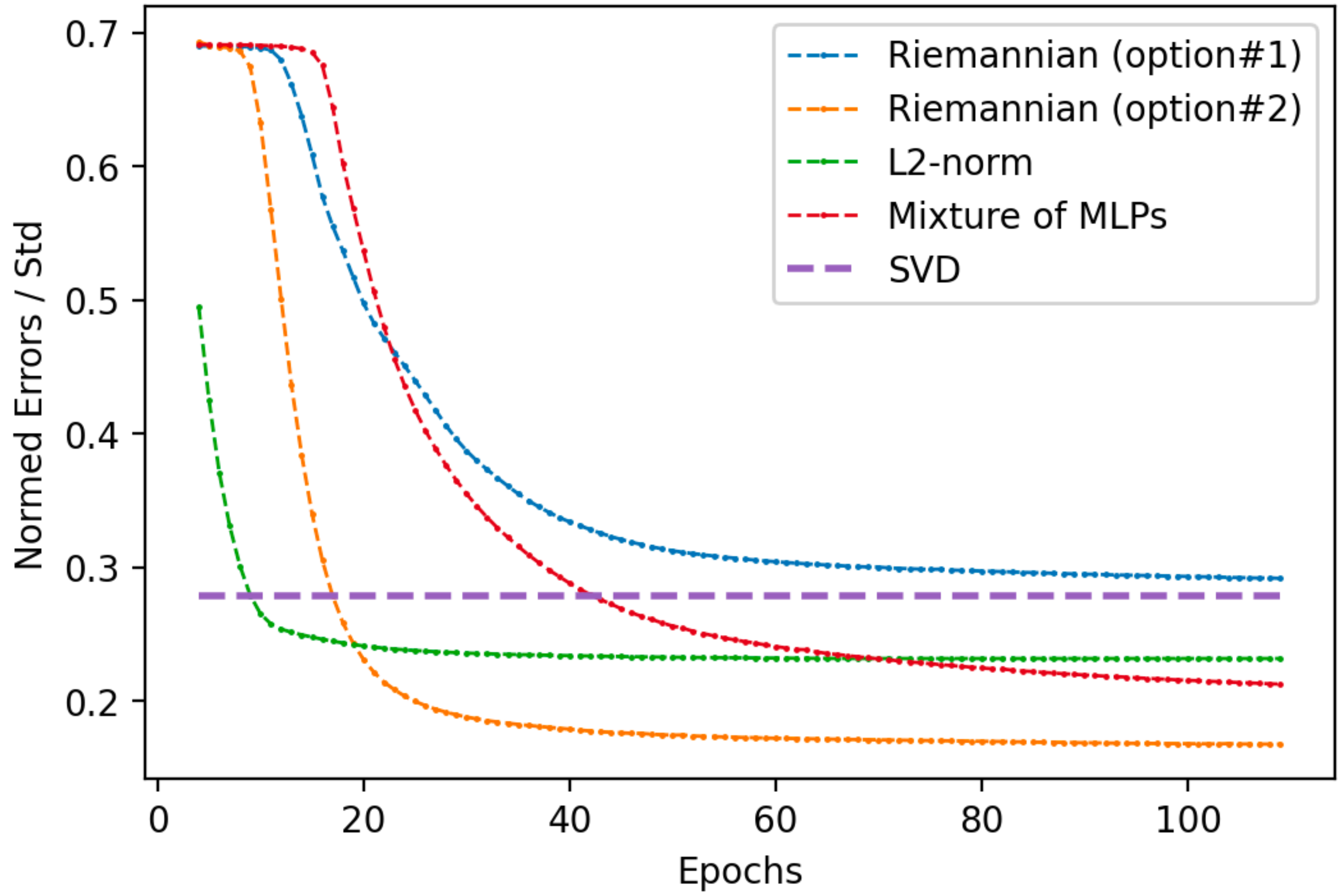
An Overview *our Method's Pipeline*



We refer to this Neuronal Riemannian Metric as *RieM*

- Theoretically and empirically, ***RieM*** can achieve stronger representational capability.
- Based on this, we propose a data-free neural compression method to transform neural structures into a more parameter efficient dynamical system without neural weights.
- The compression process is further optimized using techniques such as our proposed *Shared Correlation Counts* and *dynamical merging mechanism*.

	Metric	No.Params (FC Layer)	Top-1 (%)	
			MNIST	CIFAR100
LeNet-5	N/A	59.3K	99.10	44.30
LeNet-5	L1-norm	6.3K	98.95	44.35
LeNet-5	L2-norm	6.3K	99.17	44.45
LeNet-5	L3-norm	6.3K	99.10	44.40
LeNet-5	RieM	7.2K	**99.28**	**44.78**
ResNet-9	N/A	102.8K	99.62	67.58
ResNet-9	L1-norm	32.0K	99.58	67.54
ResNet-9	L2-norm	32.0K	99.64	67.63
ResNet-9	L3-norm	32.0K	99.62	67.53
ResNet-9	RieM	33.4K	99.69	68.12
ResNet-9	RieM	51.5K	**99.72**	**68.15**



Empirical Results on Vision Benchmarks

	Method	Data-free	Size (MB)	W/A-bit	Top-1 (%)
ResNet-18	Original	×	46.83	32/32	71.47
	DFQ (Nagel et al., 2019)	√	8.36	6/6	66.30
	UDFC (Bai et al., 2023)	√	8.36	6/6	**72.70**
	RieM (Ours)	√	8.36	8/16	71.80
	DDAQ (Li et al., 2022c)	√	5.58	4/4	58.44
	DSG (Zhang et al., 2021)	×	5.58	4/4	34.33
	UDFC (Bai et al., 2023)	√	5.58	4/4	63.49
	Lp-norm (Pei & Wang, 2023)	√	5.58	8/16	64.52
	RieM (Ours)	√	5.58	8/16	**66.30**
ResNet-50	Original	×	102.53	32/32	77.72
	OSME (Choukroun et al., 2019)	√	12.28	4/32	67.36
	GDFQ (Xu et al., 2020)	×	12.28	4/4	55.65
	SQuant (Guo et al., 2022)	√	12.28	4/4	70.80
	UDFC (Bai et al., 2023)	√	12.28	4/4	72.09
	Lp-norm (Pei & Wang, 2023)	√	12.28	8/16	72.96
	RieM (Ours)	√	12.28	8/16	**73.26**
DenseNet-121	Original	×	32.34	32/32	74.36
	OMSE (Choukroun et al., 2019)	√	6.00	4/32	64.40
	UDFC (Bai et al., 2023)	√	6.00	4/32	70.15
	Lp-norm (Pei & Wang, 2023)	√	6.00	8/16	71.66
	RieM (Ours)	√	6.00	8/16	**73.15**

	Method	Prune-Ratio	W-bit	Size (MB)	FLOPs (G)	Top-1 (%)
ResNet-34	Original	0%	32	87.32		73.27
	Neuron Merge (Kim et al., 2020)	10%	32	78.8	6.84	67.10
	UDFC (Bai et al., 2023)	10%	6	14.8	6.84	69.86
	RieM (Ours)	10%	6	14.8	5.30	**72.216**
	Neuron Merge (Kim et al., 2020)	30%	32	61.6	5.30	39.40
	UDFC (Bai et al., 2023)	30%	6	11.6	5.30	59.25
	RieM (Ours)	30%	6	11.6	5.30	**70.144**
ResNet-101	Original	0%	32	178.81		77.31
	Neuron Merge (Kim et al., 2020)	10%	32	154.4	3.24	72.46
	UDFC (Bai et al., 2023)	10%	6	28.8	3.24	74.69
	RieM (Ours)	10%	6	28.8	2.52	**76.032**
	Neuron Merge (Kim et al., 2020)	30%	32	112.4	2.52	38.44
	UDFC (Bai et al., 2023)	30%	6	21.2	2.52	65.76
	RieM (Ours)	30%	6	21.2	2.52	**73.296**

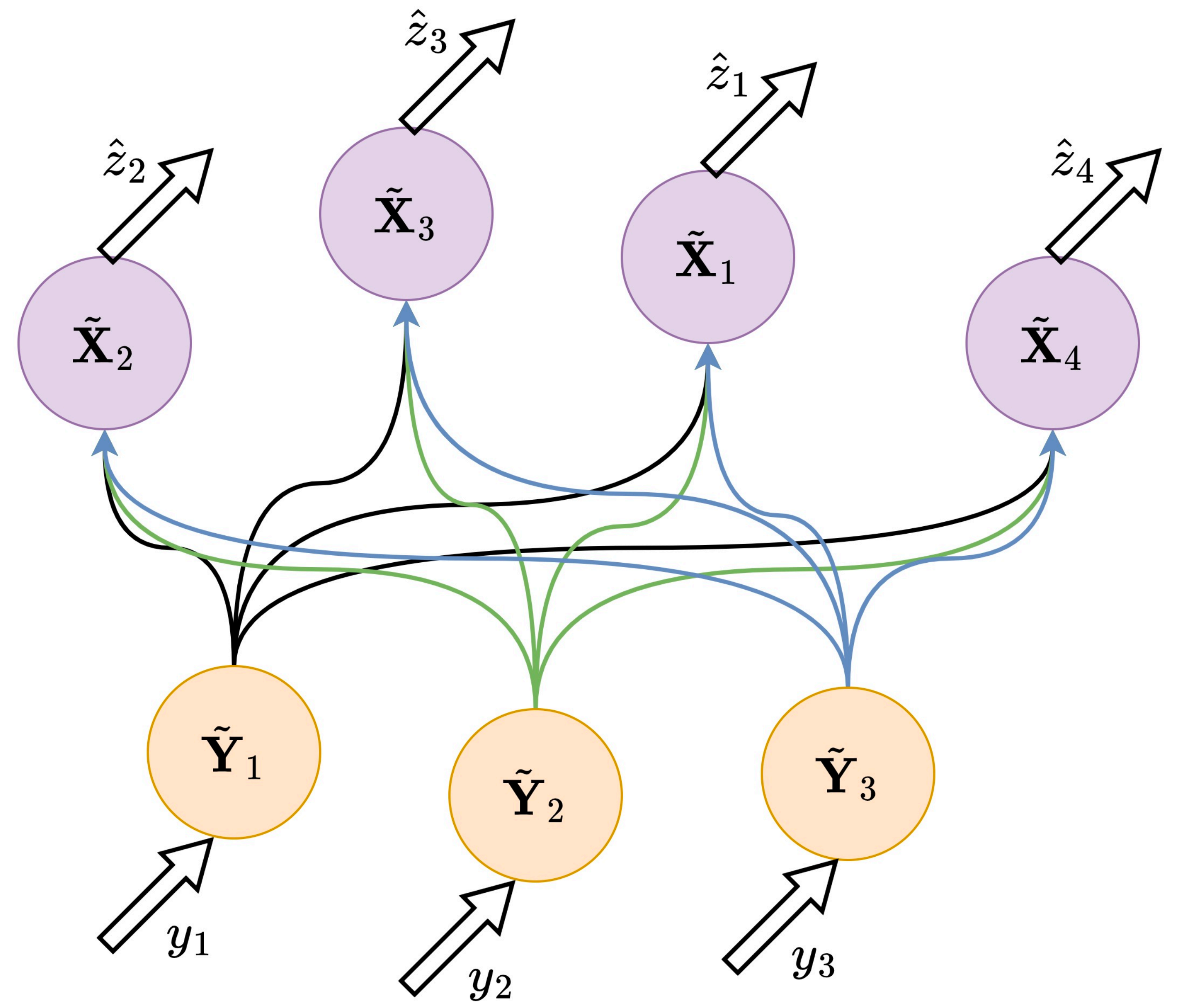
Method	Data-free	W-bit	Size (MB)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DETR	×	32	159.0	40.1	60.6	42.0	18.3	43.3	59.5
T-DETR (Zhen et al., 2022)	×	8	43.6	-0.6	-0.8	-0.4	**+0.5**	-0.9	**-1.5**
T-DETR	×	4	33.4	-2.2	-2.7	-2.2	-1.0	-2.7	-3.2
Quant-DETR	√	8	43.6	-2.2	-1.2	-3.1	-2.5	-2.5	-1.8
SVD-DETR	√	8	33.4	-11.5	-14.2	-12.8	-6.1	-15.1	-11.6
RieM-DETR (Ours)	√	8	43.6	**-0.4**	-0.6	**+0.1**	+0.4	**-0.3**	**-1.5**
RieM-DETR (Ours)	√	8	33.4	-0.7	**-0.5**	-1.2	+0.1	-1.3	-2.1
RieM-DETR (Ours)	√	8	26.7	-2.8	-2.5	-3.4	-2.4	-4.4	-4.1

- Better data-free neural compression on ImageNet-1k compared with other Quantization and Pruning methods.
- Improve the Parameter-efficiency on the COCO object detection benchmark compared with other Compression methods.

A New Paradigm of Dimensionality Reduction Techniques

MATRIX SHAPE	$\mathbb{R}^{1000\times1000}$		$\mathbb{R}^{5000\times5000}$		$\mathbb{R}^{10000\times10000}$	
$T_{comp.}/T_{naive}$	0.1	0.3	0.1	0.3	0.1	0.3
ISOMETRIC MAPPING	1.22E-01	1.23E-01	6.27E-02	6.28E-02	4.55E-02	4.56E-02
AUTOENCODER	4.60E-02	3.66E-02	1.57E-02	2.86E-02	4.43E-02	4.33E-02
DEEP AUTOENCODER	3.16E-02	3.41E-02	1.35E-02	1.58E-02	9.50E-03	3.93E-02
LOCALLY LE	3.16E-02	3.16E-02	1.41E-02	1.41E-02	9.98E-03	9.98E-03
NYSTROM	3.16E-02	3.16E-02	1.41E-02	1.41E-02	9.98E-03	9.98E-03
KERNEL PCA	1.35E-03	1.35E-03	7.70E-04	7.80E-04	7.40E-04	7.50E-04
LP-NORM	2.50E-04	1.31E-04	2.15E-05	1.65E-05	9.87E-06	7.88E-06
RIEM (OURS)	2.20E-04	1.20E-04	1.58E-05	1.26E-05	5.56E-06	6.27E-06

- Normalized matrix-vector production error on a synthetic matrix.
- The ratio $T_{comp.}/T_{naive}$ represents refers to the compression ratio.



- For a vector $\mathbf{y} \in \mathbb{R}^3$ and a matrix $\mathbf{A} \in \mathbb{R}^{4\times3}$, computing $\hat{z} = \mathbf{A}\mathbf{y}$ is equivalent to transmitting signals $\mathbf{y} \in \mathbb{R}^3$ from a set of point groups $\{\tilde{\mathbf{Y}}_1, \ldots, \tilde{\mathbf{Y}}_3\}$ to another set of point groups $\{\tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_4\}$.

Conclusion

- Basically, any matrix of $\mathbb{R}^{a \times b}$ within a neural structure can be converted into $a + b$ neurons interpreted as d-dimensional neuronal dynamics via ***RieM***, enabling better data-free neural compression.
- Moreover, ***RieM***-based neural representation enables better integration of black-box neural models with solid physical interpretations.
- However, ***RieM*** still require time-consuming iterative updates and are sensitive to parameter initialization.
- Therefore, future work involves refining the computational form, reducing the conversion time, and deriving a more accurate physics-inspired framework to enhance neural interpretability and efficiency.

THANKS !