Dynamics-inspired Neuromorphic Visual Representation Learning

Zhengqi Pei [1 2] , Shuhui Wang [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.

Background & Motivation

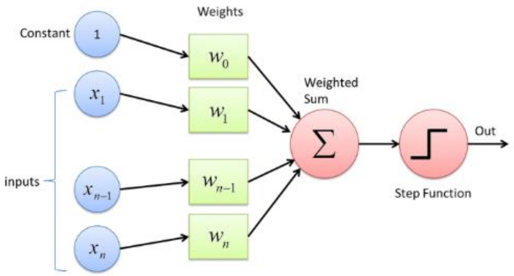
Learning mechanisms and processes in the nervous system

Donald Olding Hebb

Hebb's law (1949) describes the principle of synaptic plasticity: sustained and repeated stimulation of presynaptic neurons to postsynaptic neurons can lead to an increase in synaptic transmission efficacy.

Weight or Force

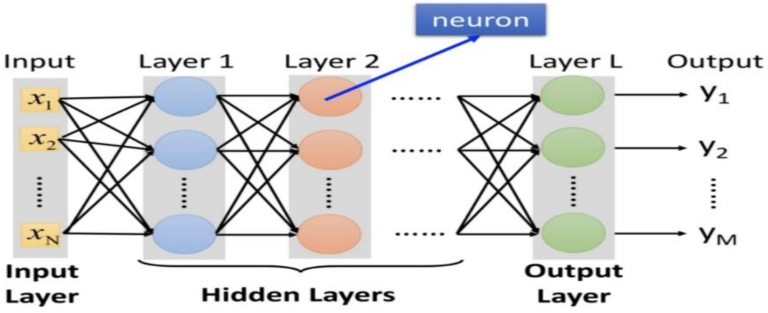
Neurons

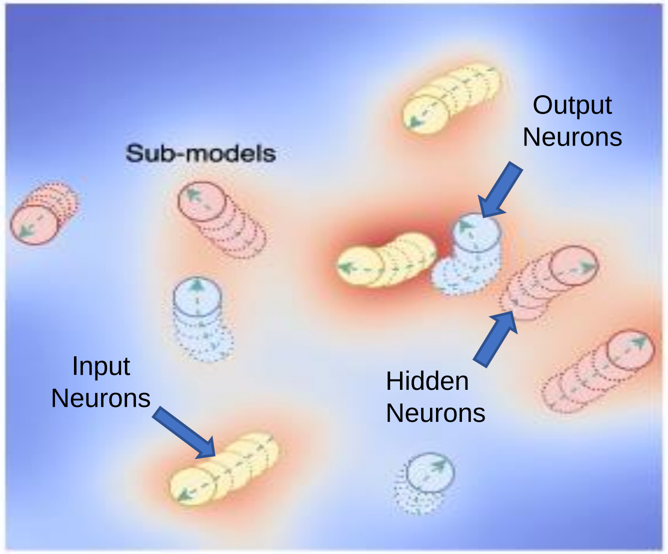


Rosenblatt, 1958

Gradients gone!
Structure fixed!
Computationally redundant!
Model black box!

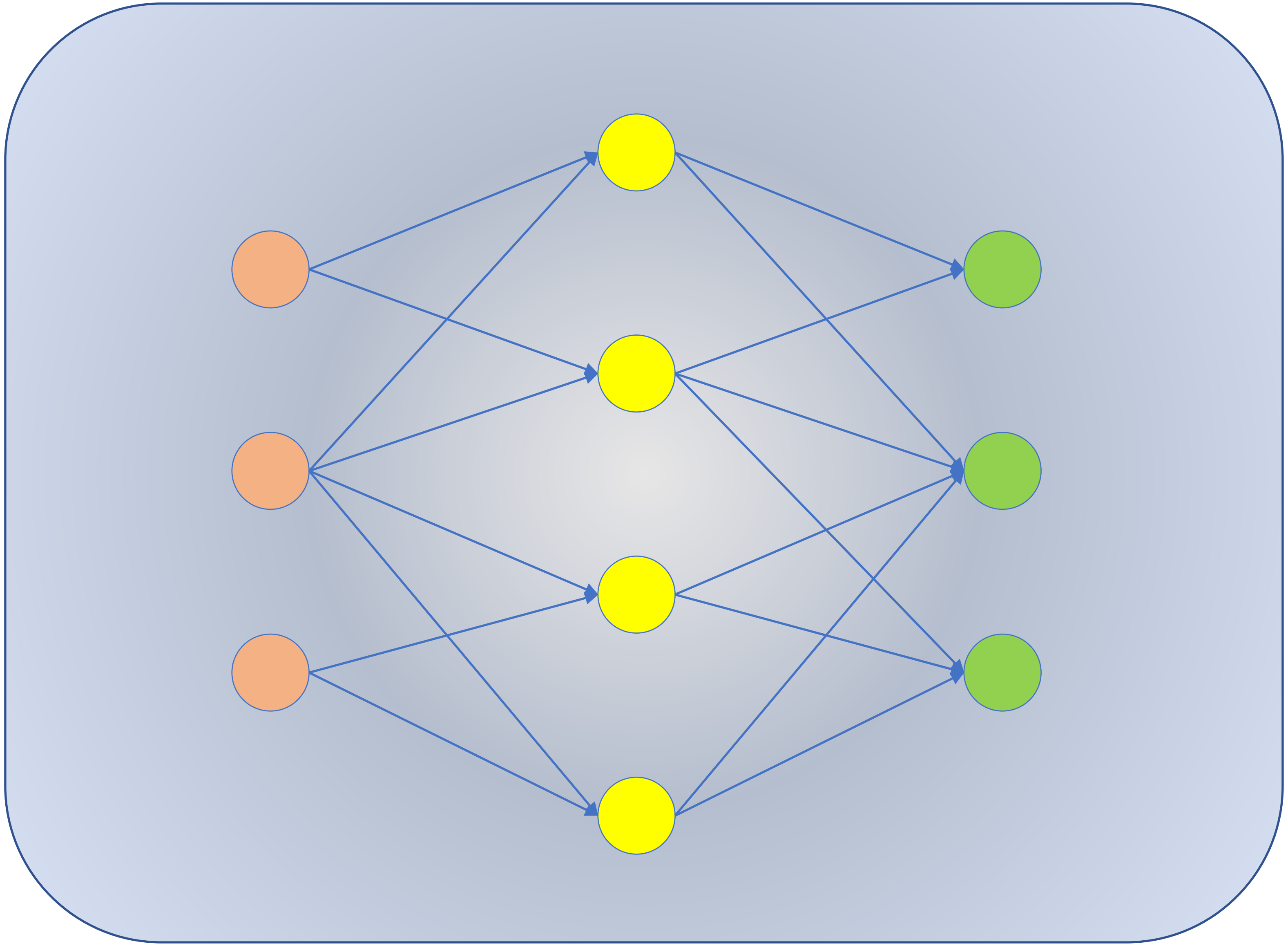
Deep neural network





Flat neural network

The neuronal dynamics without weights



The neuron is placed in hyperspace and the coordinate information represents its neuronal dynamics

Neurons in hyperspace are called sub-models and can adjust their dynamics in response to signals

The object being trained is no longer the weights between neurons, but the dynamics of the neurons themselves

During training, signals will be converted into forces that change the dynamics of neurons

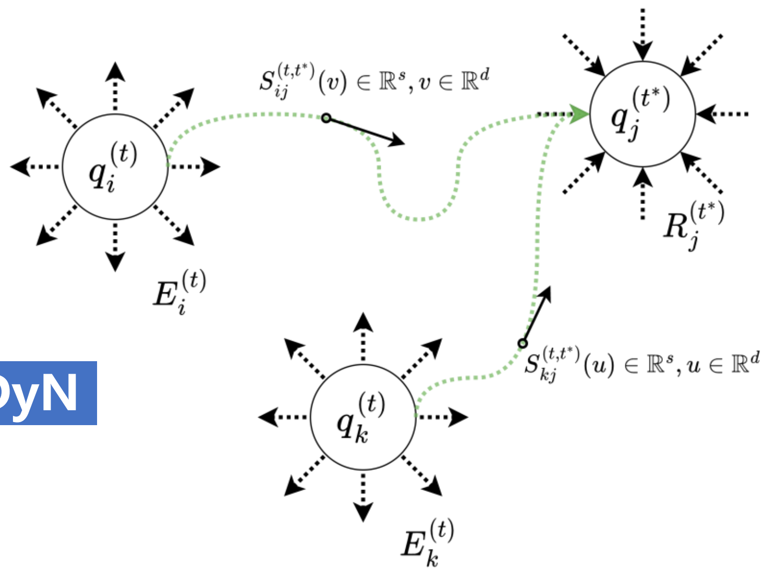
Dynamical UAT

$$R_i^{(t)} = \sum_{j \neq i}^{N} E_j^{(t-\epsilon)} \cdot \varphi\left(q_j^{(t-\epsilon)}, q_i^{(t)}\right)$$

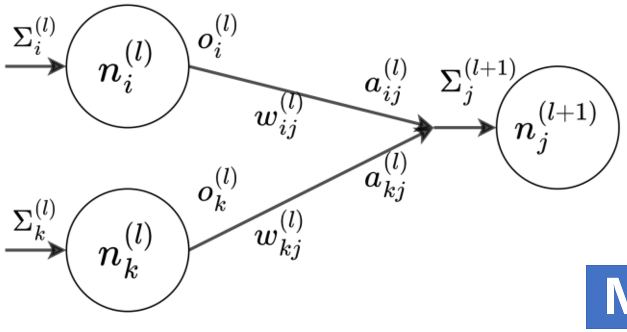
$$E_i^{(t)} = \mathcal{A} R_i^{(t)} + \mathcal{B} q_i^{(t)} + \mathcal{C} \frac{d}{dt} q_i^{(t)}$$

$$\frac{d}{dt} q_i^{(t)} = \mathcal{D} R_i^{(t)} + \mathcal{E} E_i^{(t)} + \mathcal{F} q_i^{(t)}$$

Interpreting an MLP as a DyN system



DyN



MLP

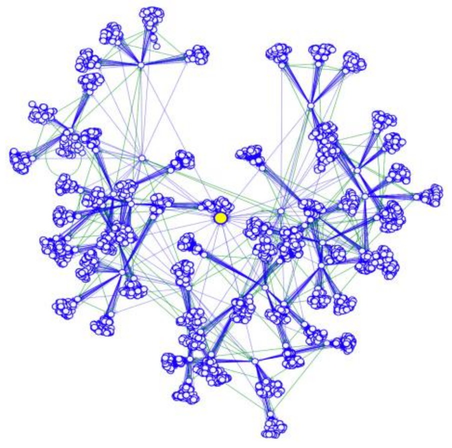
$$E_i^{(t)}(q_i^{(t)}) = \sigma \circ R_i^{(t)}(q_i^{(t)}) \quad \longleftrightarrow \quad o_i^{(l)} = \sigma \circ \Sigma_i^{(l)}$$

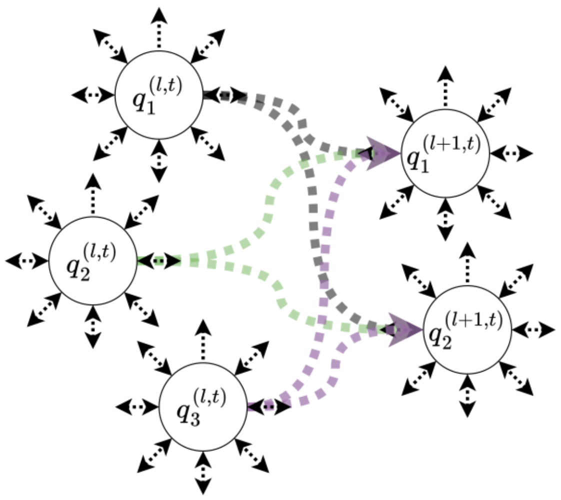
$$R_j^{(t^*)}(q_j^{(t^*)}) = \sum_i S_{ij}^{(t,t^*)}(q_j^{(t^*)}) \quad \longleftrightarrow \quad \Sigma_j^{(l+1)} = \sum_x a_{xj}^{(l)}$$

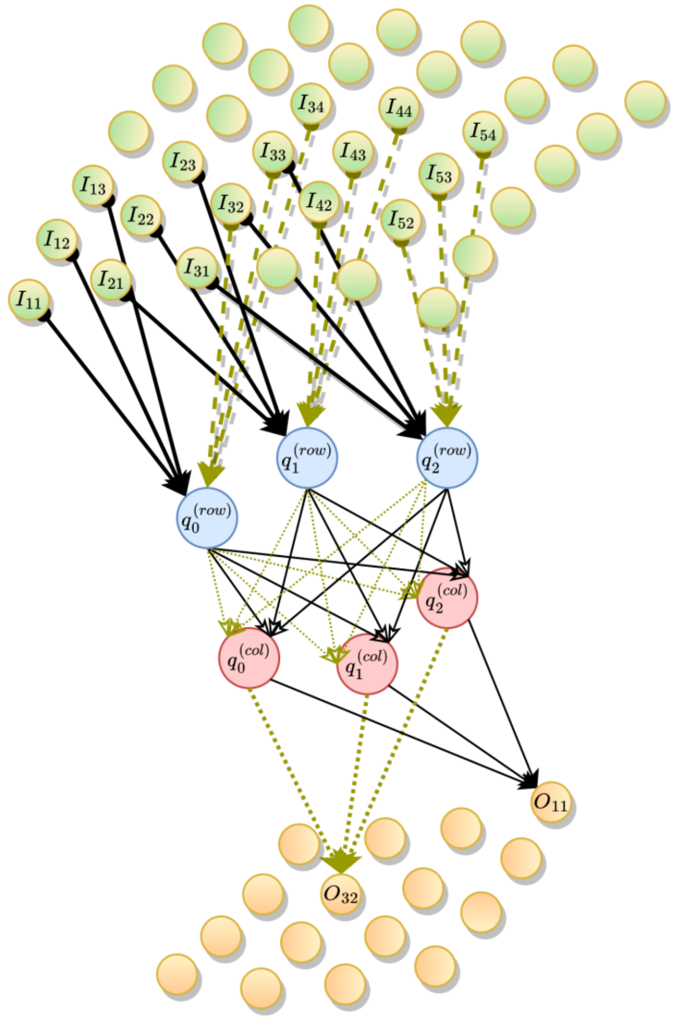
$$\mu_s(S_{ij}^{(t,t^*)}(q_j^{(t^*)})) = \mu_s(S_{ij}^{(t,t^*)}(q_i^{(t)})) \cdot \frac{c}{\mu_q(q_j^{(t^*)} - q_i^{(t)})} \quad \longleftrightarrow \quad a_{ij}^{(l)} = o_i^{(l)} \cdot w_{ij}^{(l)}$$

Interpreting arbitrary neural structures as DyN systems

Every tensors-based neural structure (e.g., attention, convolutional layer, FC layer) can be represented by a set of subsystems that deal with time-variant signals







From neural layer to DyN. We denote $\mathrm{P}(x)$ as a subsystem containing x sub-models.

Models	Layer Types	DyN Types
MLP	$M_{FC} \in \mathbb{R}^{m\times n}$	$P(m)+P(n)$
CNN	$M_C \in \mathbb{R}^{k\times k\times N_{in}\times N_{out}}$	$2k \cdot P(N_{in}+N_{out})$
Trans-former	$M_Q \in \mathbb{R}^{T\times d_k}$ $M_K \in \mathbb{R}^{T\times d_k}$ $M_V \in \mathbb{R}^{T\times d_v}$	$P(2d_k+d_v)+P(T)$

Training arbitrary neural layers with DyN mechanism

Step 1:Configure the (arbitrary) target neural structures using DyN forms.

Method 1

Step 2:Compute the path integrals amongst sub-models.

Step 3:Recover the weight-based model and fine-tune it based on labels.

Step 4:Find the stress forces f between fine-tuned weights and path integrals.

Method 2

Step 2:Feedforward the DyN model and get the output signals P.

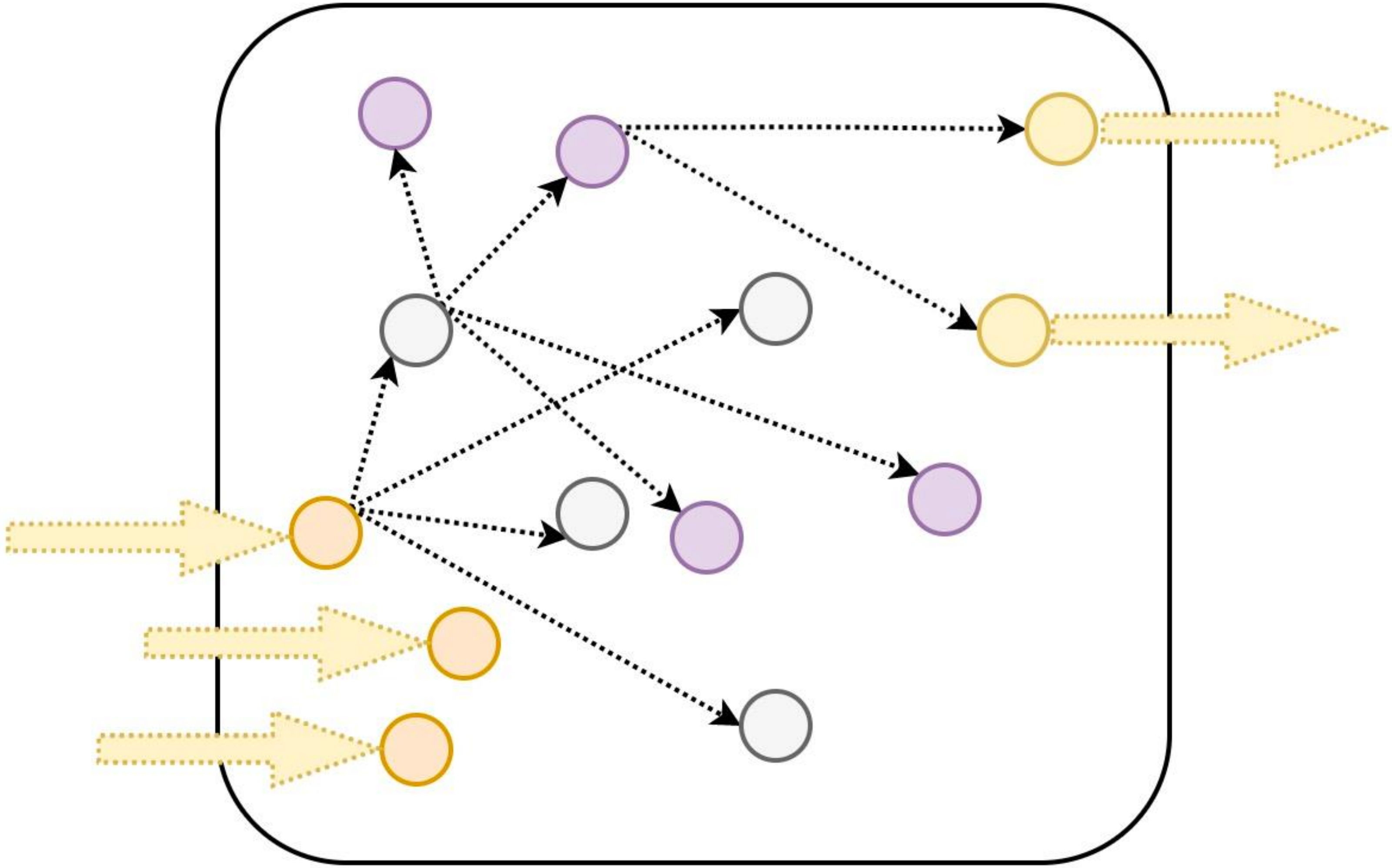
Step 3: Obtain the feedback signals $\mathcal{L} = \| T - P \|_2$ based on labels T.

Step 2:Compute the stress forces $f = \frac{\partial \mathcal{L}}{\partial Q}$ based on the configured structures.

Step 5:Update the sub-models via $\frac{\partial Q}{\partial t} = F(Q, f)$ until reaches local equilibrium.

Step 6:Repeat step 2 ~ 5 until reaches global equilibrium.

Inference with DyN mechanism



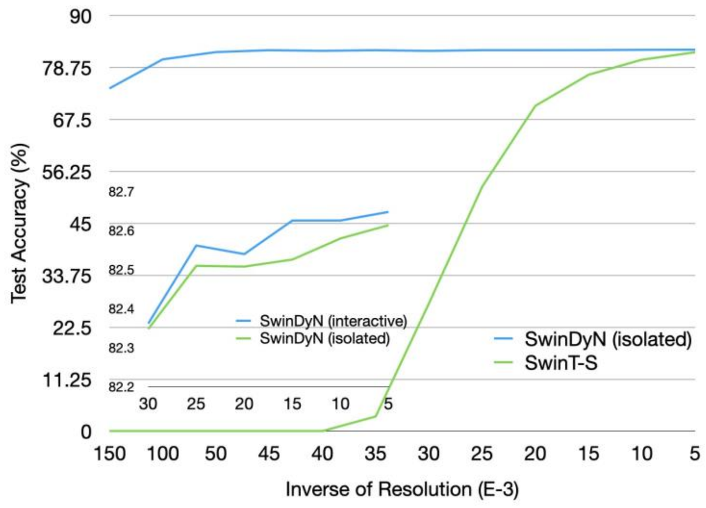
Small-scale Experiments on MNIST

Compared against feedforward neural networks and LeNet-5, our randomly initialized DyN models trained from scratch demonstrate higher accuracy, lower computational complexity, and reduced parameter size.

Model	Layer Type	No.Copies		No.Params		Test Acc. (%)	
		FC	Conv	Memory	Disk	Fixed (Eq. 6)	Unfixed (Alg. 1)
3-Layered NN	FC	-		2,290K		97.89±0.10	
	DyN	50	-	1360K	160K	-	98.32±0.03
	DyN	75	-	2170K	250K	-	**98.36±0.02**
LeNet-5	FC, Conv	-		61.8K		99.06±0.10	
	DyN	2	3	14.50K	2.03K	81.44	99.13±0.10
	DyN	2	5	16.48K	2.25K	84.95	99.15±0.07
	DyN	3	6	23.01K	2.98K	96.28	99.21±0.05
	DyN	5	8	36.04K	4.44K	98.10	99.21±0.09
	DyN	7	7	46.11K	5.56K	98.83	**99.23±0.06**

Experiments on ImageNet+WebVision

Model Configs		No.Params (Millions)	MACs (GFLOPs)	ImageNet (%)		WebVision (%)	
Structure	Layer Type			Ideal	δ=1e^{-3}	Ideal	δ=1e^{-3}
DenseNet-161	FC, Conv	28.68	7.82	75.254	71.336	68.973	61.429
	DyN	**6.05**	3.28 (0.089)	**75.314**	**75.246**	**69.033**	**68.984**
ResNet-152	FC, Conv	60.40	11.58	77.014	75.776	69.879	59.435
	DyN	**6.51**	5.25 (3.5E-3)	**77.203**	**76.604**	**70.005**	**69.998**
ViT-S-224	FC, Conv, Attn	36.38	1.11	80.108	80.038	72.665	72.509
	DyN	**3.71**	0.45 (0.75E-3)	**80.150**	**80.122**	**72.728**	**72.716**
SwinT-S-224	FC, Conv, Attn	49.94	8.52	82.634	82.070	72.755	72.604
	DyN	10.38	3.35 (0.024)	82.646	82.604	72.802	72.740
	DyN	**6.65**	2.37 (0.018)	**82.688**	**82.660**	**72.934**	**72.842**



we use several pre-trained models as backbone networks and convert their FC, convolution, and attention layers into DyN forms.

- Parameters reduces
- Robustness on parameters improves
- Testing accuracy slightly improves
- Computational complexity reduces

Thanks