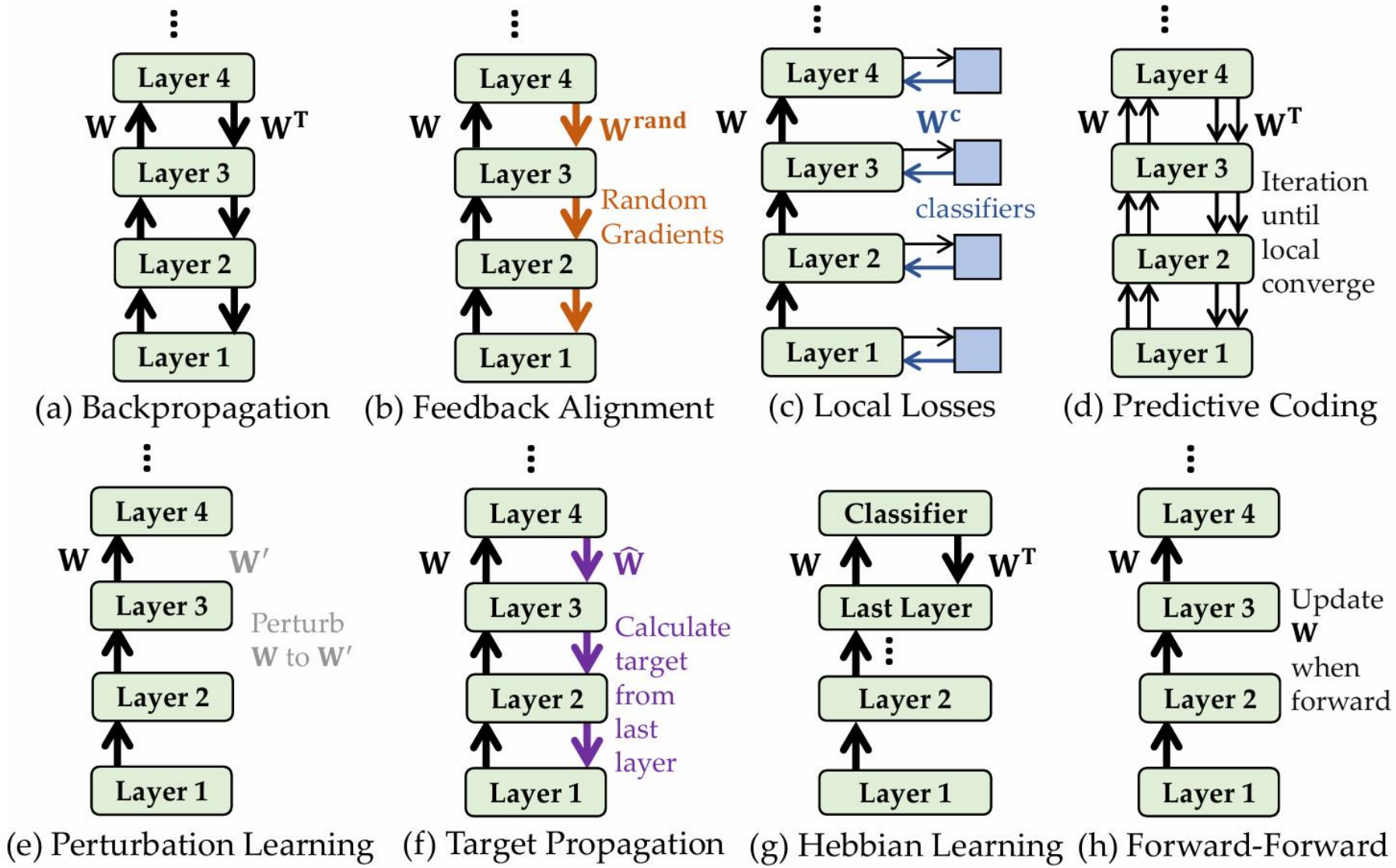
Dendritic Localized Learning: Toward Biologically Plausible Algorithm

Changze Lv[1,*], Jingwen Xu[1,*], Yiyang Lu[1,*], Xiaohua Wang[1], Zhenghua Wang[1], Zhibo Xu[1], Di Yu[2], Xin Du[2], Xiaoqing Zheng[1,$], Xuanjing Huang[1]

[1]Fudan University, [2]Zhejiang University, [*]Equal Contribution, [$]Corresponding Author

Contents

Biological Plausible Algorithms



Three Criteria

C1. Asymmetry of Forward and Backward Weights

Backward weight should not be $\mathbf{W}^T$

C2. Local Error Representation

Adjust solely on local information, without relying on a global error signal

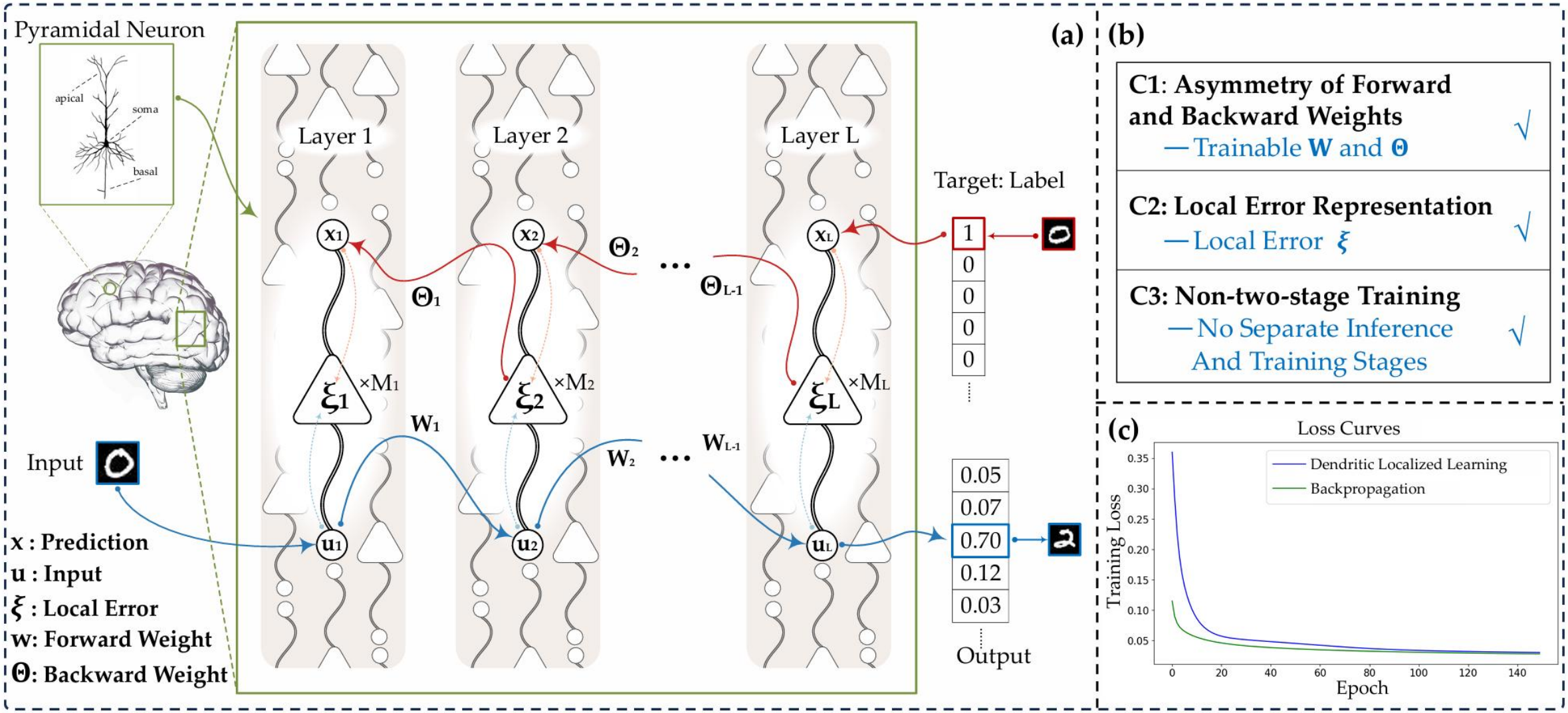
C3. Non-two-stage Training

No separate forward (inference) and backward (updating) phases

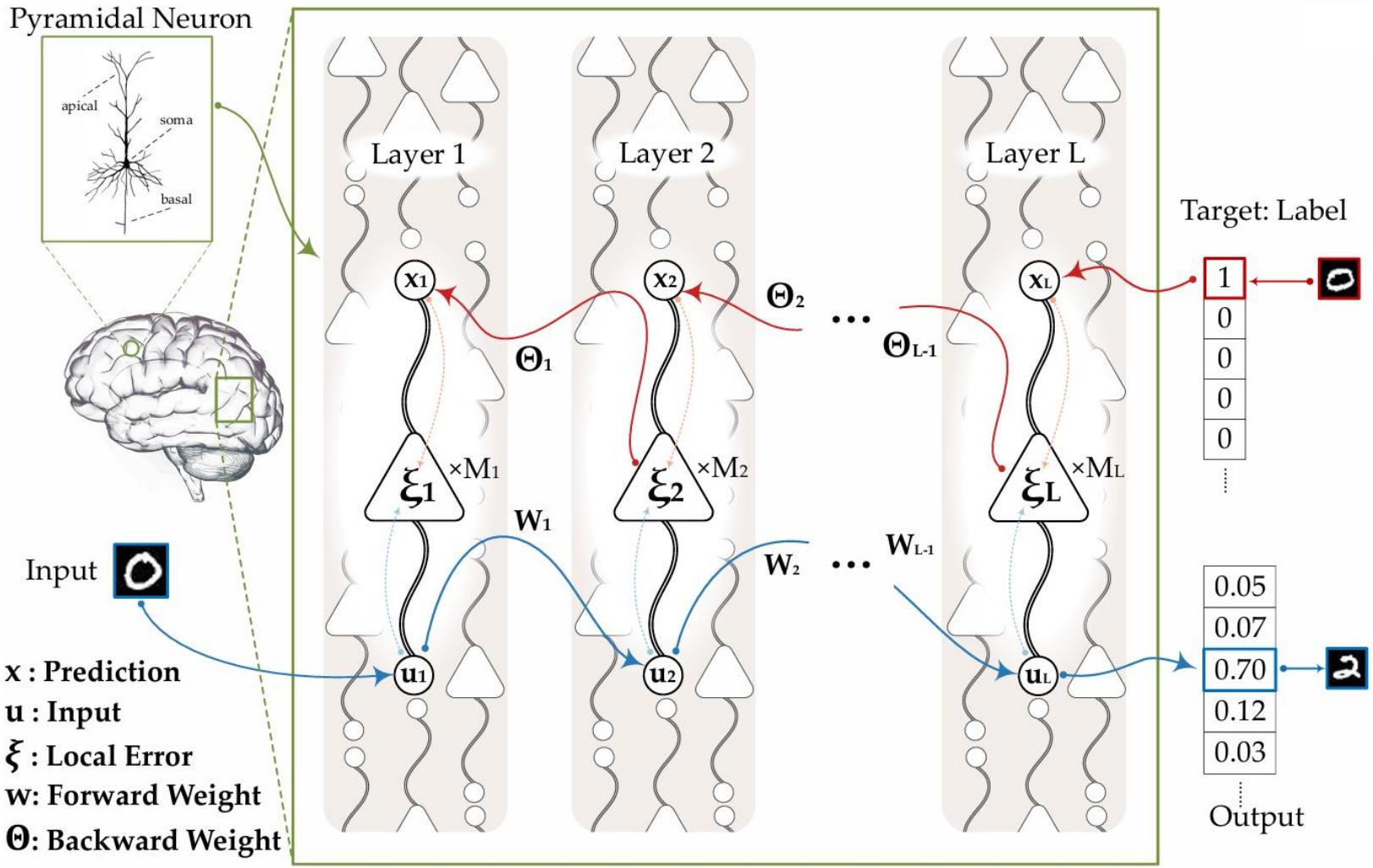
Contents

- **Criteria for Biological Plausibility**
  - Current Algorithms; Three Criteria
- **Dendritic Localized Learning**
  - Overview; Training MLPs/CNNs; Training RNNs
- **Experiments**
  - Image Recognition; Sequential Tasks; Analysis

Overview


Pyramidal Neuron
apical
soma
basal
Input
x : Prediction
u : Input
ξ : Local Error
w: Forward Weight
Θ: Backward Weight
(a)
Layer 1
Layer 2
Layer L
Target: Label
Output
(b)
C1: Asymmetry of Forward and Backward Weights
— Trainable W and Θ
C2: Local Error Representation
— Local Error ξ
C3: Non-two-stage Training
— No Separate Inference And Training Stages
(c)
Loss Curves
Dendritic Localized Learning
Backpropagation
Training Loss
Epoch

Training MLPs/CNNs



Init

$$\mathbf{u}_{i+1} = f(\mathbf{W}_i \mathbf{u}_i), \quad \mathbf{x}_{i+1} = \mathbf{u}_{i+1}$$

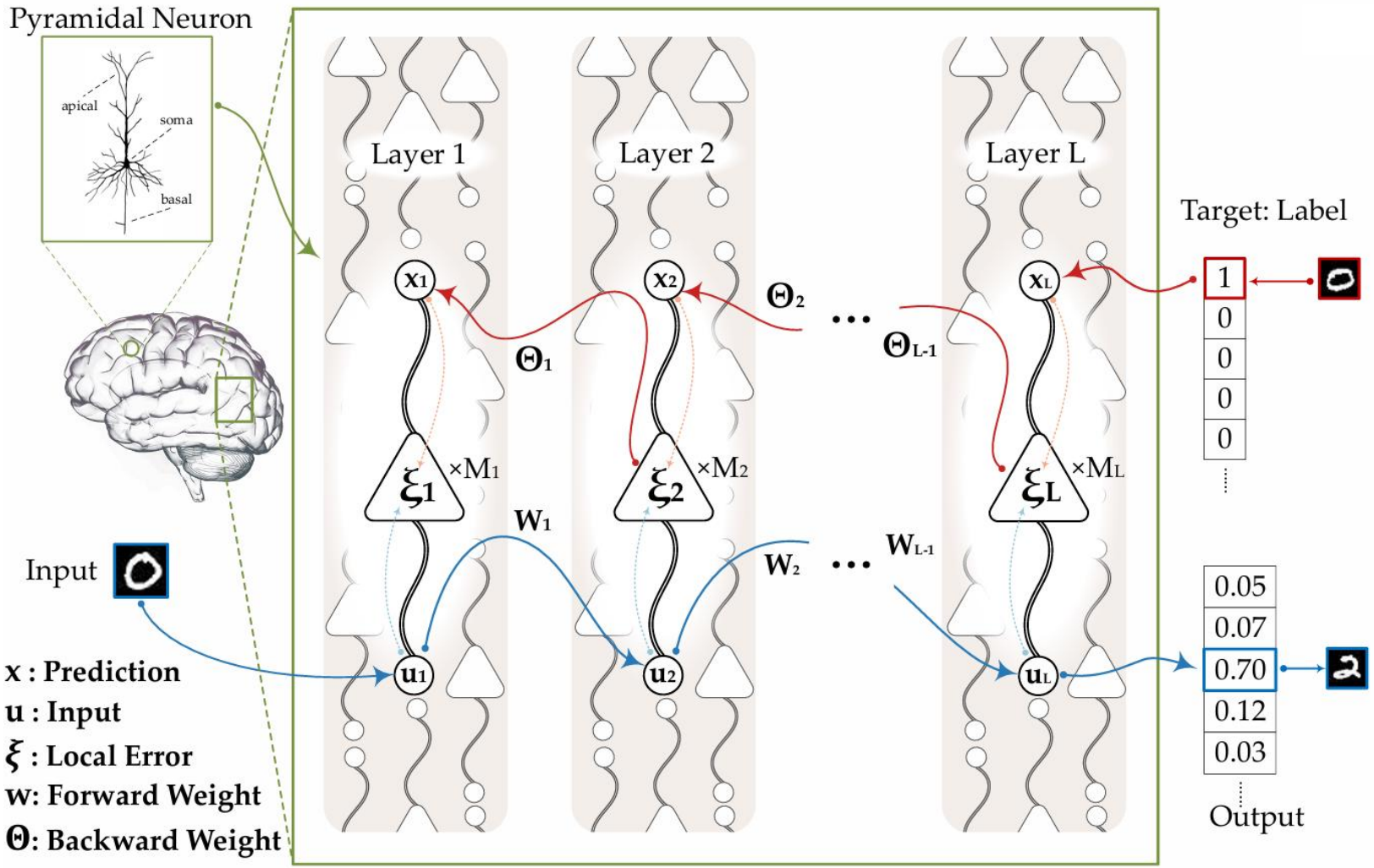
Total Loss:

$$\mathcal{L} = -\frac{1}{2}\sum_{i=1}^{L} \boldsymbol{\xi}_i^2 = -\frac{1}{2}\sum_{i=1}^{L} (\mathbf{x}_i - \mathbf{u}_i)^2$$

Direction of adjustment:

$$\begin{aligned}
\Delta \mathbf{x}_i &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}_i} \\
&= \frac{\partial(-\frac{1}{2}\boldsymbol{\xi}_i^2 - \frac{1}{2}\boldsymbol{\xi}_{i+1}^2)}{\partial \mathbf{x}_i} \\
&= -\boldsymbol{\xi}_i + \frac{\partial \mathbf{u}_{i+1}}{\partial \mathbf{x}_i}\boldsymbol{\xi}_{i+1} \\
&= -\boldsymbol{\xi}_i + \mathbf{W}_i^T[\boldsymbol{\xi}_{i+1} \odot f'(\mathbf{W}_i \mathbf{u}_i)]
\end{aligned}$$

Training MLPs/CNNs



Replacement

$$\Delta \mathbf{x}_i = -\boldsymbol{\xi}_i + \boldsymbol{\Theta}_i^T [\boldsymbol{\xi}_{i+1} \odot f'(\mathbf{W}_i \mathbf{u}_i)]$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \eta_{\mathbf{x}} * \Delta \mathbf{x}_i$$

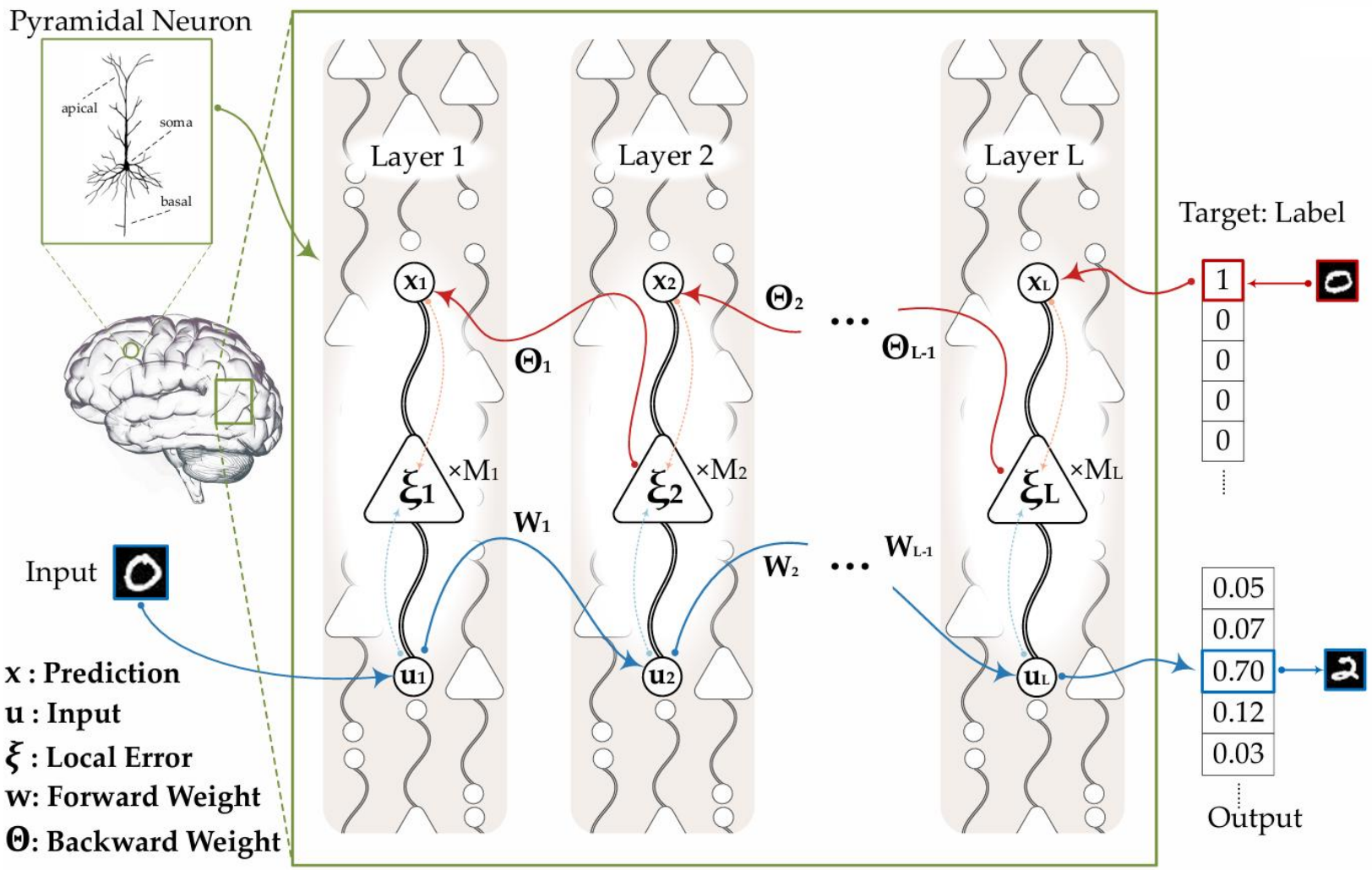
When local convergence, we have

$$\Delta \mathbf{x}_i = 0$$

Therefore

$$\boldsymbol{\xi}_i = \boldsymbol{\Theta}_i^T [\boldsymbol{\xi}_{i+1} \odot f'(\mathbf{W}_i \mathbf{u}_i)]$$

Training MLPs/CNNs



Update

$$\mathbf{W}_i \leftarrow \mathbf{W}_i + \eta_{\mathbf{W}} * \Delta \mathbf{W}_i$$

$$\begin{aligned}\Delta \mathbf{W}_i &= \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i} \\ &= \frac{\partial(-\frac{1}{2}\boldsymbol{\xi}_{i+1}^2)}{\partial \mathbf{W}_i} \\ &= -\boldsymbol{\xi}_{i+1}(-\frac{\partial \mathbf{u}_{i+1}}{\partial \mathbf{W}_i}) \\ &= \boldsymbol{\xi}_{i+1} \odot f'(\mathbf{W}_i \mathbf{u}_i)\mathbf{u}_i\end{aligned}$$

$$\boldsymbol{\Theta}_i \leftarrow \boldsymbol{\Theta}_i + \eta_{\boldsymbol{\Theta}} * \Delta \boldsymbol{\Theta}_i$$

$$\begin{aligned}\Delta \boldsymbol{\Theta}_i^T &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Theta}_i^T} \\ &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}_i}\frac{\partial \boldsymbol{\xi}_i}{\partial \boldsymbol{\Theta}_i^T} \\ &= \frac{\partial\left(-\frac{1}{2}\boldsymbol{\xi}_i^2\right)}{\partial \boldsymbol{\xi}_i}\frac{\partial \boldsymbol{\Theta}_i^T[\boldsymbol{\xi}_{i+1} \odot f'(\mathbf{W}_i \mathbf{u}_i)]}{\partial \boldsymbol{\Theta}_i^T} \\ &= -\boldsymbol{\xi}_i[\boldsymbol{\xi}_{i+1} \odot f'(\mathbf{W}_i \mathbf{u}_i)].\end{aligned}$$

Algorithm

Algorithm 1 Algorithm of Dendritic Localized Learning

Input: data D, number of layers L, number of neurons per layer N, learning rate $\eta_{\mathbf{W}}$, $\eta_{\mathbf{\Theta}}$
Initialize $\mathbf{W}$, $\mathbf{\Theta}$ randomly for all layers
for epoch = 0 to max_epochs **do**
  for each batch $\mathbf{B}$ in D **do**
    Forward Pass (when input stimuli are fed to the basal dendrite):
    Assign the input values to the basal dendrites of the input layer neurons: $\mathbf{u}_0 = \mathbf{B}$, and initialize $\mathbf{x}_0 = \mathbf{u}_0$
    for $i = 0$ to $L - 1$ **do**
      Compute forward pass $\mathbf{u}_{i+1} = f_i(\mathbf{W}_i\mathbf{u}_i)$, where f_i is the activation function for the i-th layer
      Initialize $\mathbf{x}_{i+1} = \mathbf{u}_{i+1}$
    end for

    Compute Local Errors (when apical dentrite receives the top-down feedback):
    Assign the target values to the apical dendrites of the output layer neurons: $\mathbf{x}_L = \mathbf{target}$
    Compute output error $\xi_L = -\nabla_{\mathbf{u}_L}\mathcal{L}(\mathbf{u}_L, \mathbf{x}_L)$
    Compute input error $\xi_0 = \mathbf{x}_0 - \mathbf{u}_0$
    for $i = L - 1$ down to 1 **do**
      Compute local error $\xi_i = \mathbf{\Theta}_i^T[\boldsymbol{\xi}_{i+1} \odot f_i{}'(\mathbf{W}_i\mathbf{u}_i)]$
    end for

    Update Weights and Thetas Simultaneously:
    for $i = 0$ to $L - 1$ **do**
      if $\boldsymbol{\xi}_{i+1} \neq 0$ **then**
        Update $\mathbf{W}_i \leftarrow \mathbf{W}_i + \eta_{\mathbf{W}} \cdot (\boldsymbol{\xi}_{i+1} \odot f_i{}'(\mathbf{W}_i\mathbf{u}_i)\mathbf{u}_i)$
        Update $\mathbf{\Theta}_i \leftarrow \mathbf{\Theta}_i + \eta_{\mathbf{\Theta}} \cdot \left\{-\boldsymbol{\xi}_i\left[\boldsymbol{\xi}_{i+1} \odot f_i{}'(\mathbf{W}_i\mathbf{u}_i)\right]\right\}^T$
      end if
    end for
  end for
end for

Training RNNs

RNNs: generate outputs every time step.

Init

$$\mathbf{h}_i^p = \mathbf{h}_i^s = f(\mathbf{W_h}\mathbf{h}_{i-1}^s + \mathbf{W_x}\mathbf{x}_i)$$

Error of hidden states

$$\boldsymbol{\xi}_i^{\mathbf{h}} = \mathbf{h}_i^p - \mathbf{h}_i^s$$

Error of outputs

$$\boldsymbol{\xi}_i^{\mathbf{y}} = \mathbf{t}_i - \mathbf{y}_i$$

Total Loss

$$\mathcal{L} = -\frac{1}{2}\sum_{i=1}^{n}[(\boldsymbol{\xi}_i^{\mathbf{h}})^2 + (\boldsymbol{\xi}_i^{\mathbf{y}})^2] = -\frac{1}{2}\sum_{i=1}^{n}\left[(\mathbf{h}_i^p - \mathbf{h}_i^s)^2 + (\mathbf{t}_i - \mathbf{y}_i)^2\right]$$

Training RNNs

Direction of adjustment

$$
\begin{aligned}
\Delta \mathbf{h}_i^p &= \frac{\partial \mathcal{L}}{\partial \mathbf{h}_i^p} \\
&= \frac{\partial \{-\frac{1}{2}\sum_{j=i}^{n}\left[(\mathbf{h}_j^p - \mathbf{h}_j^s)^2 + (\mathbf{t}_j - \mathbf{y}_j)^2\right]\}}{\partial \mathbf{h}_i^p} \\
&= \frac{\partial\left[-\frac{1}{2}(\mathbf{h}_i^p - \mathbf{h}_i^s)^2 - \frac{1}{2}(\mathbf{t}_i - \mathbf{y}_i)^2\right]}{\partial \mathbf{h}_i^p} + \frac{\partial \{-\frac{1}{2}\sum_{j=i+1}^{n}\left[(\mathbf{h}_j^p - \mathbf{h}_j^s)^2 + (\mathbf{t}_j - \mathbf{y}_j)^2\right]\}}{\partial \mathbf{h}_{i+1}^p}\frac{\partial \mathbf{h}_{i+1}^p}{\partial \mathbf{h}_i^p} \\
&= -\boldsymbol{\xi}_i^{\mathbf{h}} + \frac{\partial \mathcal{L}}{\partial \mathbf{y}_i}\frac{\partial \mathbf{y}_i}{\partial \mathbf{h}_i^p} + \frac{\partial \mathcal{L}}{\partial \mathbf{h}_{i+1}^p}\frac{\partial \mathbf{h}_{i+1}^p}{\partial \mathbf{h}_i^p} \\
&= -\boldsymbol{\xi}_i^{\mathbf{h}} + \frac{\partial\left[-\frac{1}{2}(\mathbf{t}_i - \mathbf{y}_i)^2\right]}{\partial \mathbf{y}_i}\frac{\partial g(\mathbf{W}_{\mathbf{y}}\mathbf{h}_i^p)}{\partial \mathbf{h}_i^p} + \Delta \mathbf{h}_{i+1}^p \frac{\partial \mathbf{h}_{i+1}^p}{\partial \mathbf{h}_i^p} \\
&= -\boldsymbol{\xi}_i^{\mathbf{h}} + \mathbf{W}_{\mathbf{y}}^T\left[\boldsymbol{\xi}_i^{\mathbf{y}} \odot g'(\mathbf{W}_{\mathbf{y}}\mathbf{h}_n^s)\right] + \mathbf{W}_{\mathbf{h}}^T\left[\boldsymbol{\xi}_{i+1}^{\mathbf{h}} \odot f'(\mathbf{W}_{\mathbf{h}}\mathbf{h}_i^p + \mathbf{W}_{\mathbf{x}}\mathbf{x}_{i+1})\right] \\
&= -\boldsymbol{\xi}_i^{\mathbf{h}} + \boxed{\mathbf{W}_{\mathbf{y}}^T}\boldsymbol{\xi}_i^{\mathbf{y}} + \boxed{\mathbf{W}_{\mathbf{h}}^T}\left[\boldsymbol{\xi}_{i+1}^{\mathbf{h}} \odot f'(\mathbf{W}_{\mathbf{h}}\mathbf{h}_i^p + \mathbf{W}_{\mathbf{x}}\mathbf{x}_{i+1})\right].
\end{aligned}
$$

Replacement

$$
\Delta \mathbf{h}_i^p = -\boldsymbol{\xi}_i^{\mathbf{h}} + \boldsymbol{\Theta}_{\mathbf{y}}^T\boldsymbol{\xi}_i^{\mathbf{y}} + \boldsymbol{\Theta}_{\mathbf{h}}^T\left[\boldsymbol{\xi}_{i+1}^{\mathbf{h}} \odot f'(\mathbf{W}_{\mathbf{h}}\mathbf{h}_i^p + \mathbf{W}_{\mathbf{x}}\mathbf{x}_{i+1})\right]
$$

Training RNNs

When local convergence, we have

$$\boldsymbol{\xi}_i^{\mathbf{h}} = \boldsymbol{\Theta}_{\mathbf{y}}^T \boldsymbol{\xi}_i^{\mathbf{y}} + \boldsymbol{\Theta}_{\mathbf{h}}^T [\boldsymbol{\xi}_{i+1}^{\mathbf{h}} \odot f'(\mathbf{W_h} \mathbf{h}_i^s + \mathbf{W_x} \mathbf{x}_{i+1})].$$

Update

$$\begin{aligned}
\Delta \mathbf{W_x} &= \frac{\partial \mathcal{L}}{\partial \mathbf{W_x}} = \sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial \mathbf{h}_i^s} \frac{\partial \mathbf{h}_i^s}{\partial \mathbf{W_x}} \\
&= \sum_{i=1}^{n} [\frac{\partial \mathcal{L}}{\partial \mathbf{h}_i^s} \odot f'(\mathbf{W_h} \mathbf{h}_{i-1}^s + \mathbf{W_x} \mathbf{x}_i)] \mathbf{x}_i \\
&= \sum_{i=1}^{n} [\boldsymbol{\xi}_i^{\mathbf{h}} \odot f'(\mathbf{W_h} \mathbf{h}_{i-1}^s + \mathbf{W_x} \mathbf{x}_i)] \mathbf{x}_i.
\end{aligned}$$

$$\begin{aligned}
\Delta \mathbf{W_h} &= \frac{\partial \mathcal{L}}{\partial \mathbf{W_h}} \\
&= \sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial \mathbf{h}_i^s} \frac{\partial \mathbf{h}_i^s}{\partial \mathbf{W_h}} \\
&= \sum_{i=1}^{n} [\boldsymbol{\xi}_i^{\mathbf{h}} \odot f'(\mathbf{W_h} \mathbf{h}_{i-1}^s + \mathbf{W_x} \mathbf{x}_i)] \mathbf{h}_{i-1}^s.
\end{aligned}$$

$$\mathbf{W_x} \leftarrow \mathbf{W_x} + \eta_{\mathbf{W_x}} * \Delta \mathbf{W_x}.$$

$$\mathbf{W_h} \leftarrow \mathbf{W_h} + \eta_{\mathbf{W_h}} * \Delta \mathbf{W_h}.$$

Training RNNs

Update

$$\begin{aligned}\Delta\Theta_{\mathbf{y}}^T &= \frac{\partial \mathcal{L}}{\partial \Theta_{\mathbf{y}}^T} \\ &= \sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}_i^{\mathbf{h}}} \frac{\partial \boldsymbol{\xi}_i^{\mathbf{h}}}{\partial \Theta_{\mathbf{y}}^T} \\ &= \sum_{i=1}^{n} -\boldsymbol{\xi}_i^{\mathbf{h}} [\boldsymbol{\xi}_i^{\mathbf{y}} \odot g'(\mathbf{W}_{\mathbf{y}} \mathbf{h}_n^s)]^T \\ &= \sum_{i=1}^{n} -\boldsymbol{\xi}_i^{\mathbf{h}} (\boldsymbol{\xi}_i^{\mathbf{y}})^T.\end{aligned}$$

$$\Theta_{\mathbf{y}} \leftarrow \Theta_{\mathbf{y}} + \eta_{\Theta_{\mathbf{y}}} * \Delta(\Theta_{\mathbf{y}}^T)^T.$$

$$\Delta\Theta_{\mathbf{h}}^T = \sum_{i=1}^{n-1} -\boldsymbol{\xi}_i^{\mathbf{h}} [\boldsymbol{\xi}_{i+1}^{\mathbf{h}} \odot f'(\mathbf{W}_{\mathbf{h}} \mathbf{h}_i^s + \mathbf{W}_{\mathbf{x}} \mathbf{x}_{i+1})]^T.$$

$$\Theta_{\mathbf{h}} \leftarrow \Theta_{\mathbf{h}} + \eta_{\Theta_{\mathbf{h}}} * \Delta(\Theta_{\mathbf{h}}^T)^T.$$

For more details, please refer to Appendix B.

Contents

- **Criteria for Biological Plausibility**
  - Current Algorithms; Three Criteria
- **Dendritic Localized Learning**
  - Overview; Training MLPs/CNNs; Training RNNs
- **Experiments**
  - Image Recognition; Sequential Tasks; Analysis

Image Recognition

Table 1. We show both the biological plausibility of various algorithms with the proposed criteria and the accuracy of image classification achieved by various bio-plausible learning algorithms. Our proposed DLL achieves the highest performance among the algorithms satisfying all criteria. "C1, C2, C3" stand for three criteria proposed in Section 2.1. All results are averaged across 4 random seeds.

Method	C1	C2	C3	Model	MNIST	FashionMNIST	SVHN	CIFAR-10	Avg.
Backpropagation	✗	✗	✗	MLPs	**98.62%**±0.17%	88.54%±0.64%	**60.91%**±0.42%	**48.74%**±0.56%	**74.20%**
				CNNs	**99.56%**±0.14%	**92.68%**±0.42%	**95.35%**±1.53%	**75.10%**±0.54%	**90.67%**
Feedback Alignment	✓	✗	✗	MLPs	91.87%±0.08%	82.16%±0.14%	54.91%±0.23%	48.46%±0.11%	69.35%
				CNNs	97.00%±0.13%	89.74%±0.17%	92.66%±0.26%	59.60%±0.46%	84.75%
Local Losses	✗	✓	✗	MLPs	98.56%±0.19%	88.07%±0.38%	59.12%±0.27%	48.58%±0.35%	73.58%
				CNNs	99.39%±0.06%	91.90%±0.26%	95.08%±0.25%	72.18%±0.10%	89.64%
Predictive Coding	✗	✓	✗	MLPs	98.42%±0.13%	**88.72%**±0.65%	59.05%±0.45%	47.34%±0.24%	73.38%
				CNNs	99.41%±0.40%	92.03%±0.70%	94.53%±1.54%	72.94%±0.32%	89.72%
Perturbation Learning	✓	✓	✗	MLPs	91.44%±0.40%	68.90%±0.47%	48.15%±1.06%	31.07%±0.31%	59.89%
				CNNs	92.61%±0.43%	75.79%±0.83%	57.69%±1.32%	39.72%±0.38%	66.45%
Difference Target Propagation	✓	✓	✗	MLPs	94.01%±0.12%	83.28%±0.31%	54.11%±0.09%	46.10%±0.10%	69.38%
				CNNs	96.40%±0.05%	90.51%±0.18%	69.72%±0.32%	50.88%±0.07%	76.88%
Hebbian Learning	✓	✓	✓	MLPs	78.29%±0.07%	67.40%±0.69%	40.80%±0.44%	19.98%±0.23%	51.62%
				CNNs	83.05%±0.12%	72.03%±0.48%	44.77%±0.33%	29.86%±0.13%	57.43%
R-STDP	✓	✓	✓	MLPs	77.18%±0.17%	70.03%±0.28%	41.76%±0.46%	22.68%±0.30%	52.91%
				CNNs	91.67%±0.04%	74.29%±0.30%	50.02%±0.32%	33.19%±0.38%	62.29%
Forward Forward	✓	✓	✓	MLPs	96.99%±0.14%	80.51%±0.74%	47.52%±0.63%	39.48%±0.10%	66.12%
				CNNs	15.66%±0.08%	10.00%±0.00%	6.70%±0.00%	10.32%±0.00%	10.67%
Equilibrium Propagation	✓	✓	✓	MLPs	93.81%±0.18%	75.65%±0.35%	22.62%±0.39%	16.93%±0.10%	52.25%
				CNNs	26.73%±0.22%	30.26%±0.29%	6.70%±0.00%	10.32%±0.00%	18.50%
DLL (Ours)	✓	✓	✓	MLPs	**97.57%**±0.40%	**87.50%**±0.43%	**56.60%**±0.12%	**45.87%**±0.10%	**71.89%**
				CNNs	**98.87%**±0.30%	**90.88%**±0.40%	**85.81%**±0.17%	**70.89%**±0.58%	**86.61%**

variants

Sequential Tasks

We train RNNs with DLL for two sequential tasks:

1. Text-character prediction
2. Time-series forecasting task

Table 2. Performance of RNNs trained with various learning algorithms on text-character prediction and time-series forecasting. The best results are formatted in **bold** font format. ↑ (↓) indicates the higher (lower) the better. All results are averaged across 3 random seeds.

Method	Harry Potter	Electricity		Metr-la		Pems-bay	
	Pred. Acc. ↑	MSE ↓	MAE ↓	MSE ↓	MAE ↓	MSE ↓	MAE ↓
Backpropagation	**51.9%**±1.0%	0.175±0.007	0.324±0.007	**0.131**±0.004	**0.214**±0.005	**0.164**±0.001	**0.190**±0.002
Predictive Coding	38.8%±1.8%	**0.162**±0.019	**0.312**±0.018	0.141±0.001	0.228±0.005	0.178±0.004	0.202±0.003
DLL (Ours)	33.7%±0.6%	0.172±0.018	0.321±0.013	0.155±0.005	0.264±0.001	0.178±0.005	0.224±0.004

We train TextCNNs with DLL for text classification tasks:

Table 5. Performance of TextCNN on text classification tasks.

Method	Subj	MR
BP_TextCNN	**88.50%**	**74.68%**
DLL_TextCNN	84.40%	70.79%

Analysis

1. Ablations on backward weights

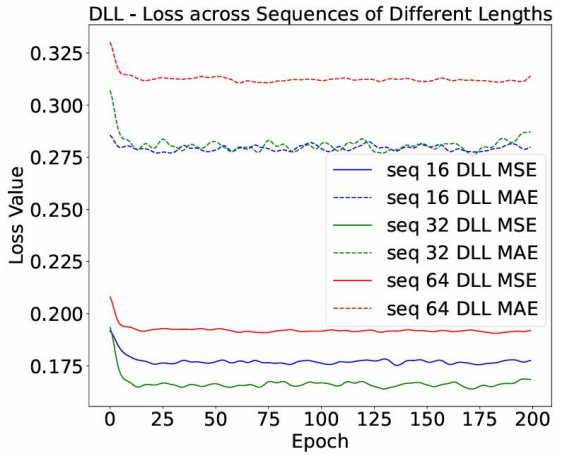
Table 3. Ablation experiments on Θ. "DLL-FA" indicates Θ initializes randomly and will not be updated, combining our proposed DLL and feedback alignment (FA). Numbers with * indicate models fail to converge. ↑ (↓) indicates the higher (lower) the better.

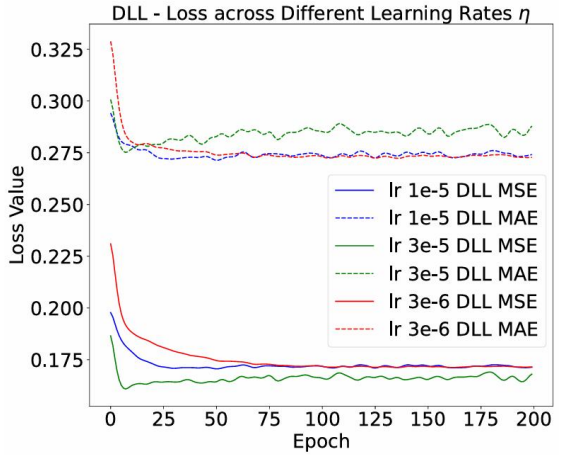
	Metric	DLL	DLL-FA
MLPs on MNIST	Acc. ↑	**97.57%**	97.37%
CNNs on CIFAR-10		**70.89%**	69.85%
RNNs on Harry Potter	Acc. ↑	**33.70%**	0.71%*
RNNs on Electricity	MSE ↓	**0.172**	0.193
	MAE ↓	**0.321**	0.345
RNNs on Pems-bay	MSE ↓	**0.178**	0.198
	MAE ↓	**0.224**	0.251

2. Time Consumption and Memory Usage

Method	Training Time (s/Epoch)	Memory Usage (MB)
BP_MLP	31.6	1286.4
BP_CNN	99.0	1272.9
DLL_MLP	44.7	1595.3
DLL_CNN	169.8	1306.9

3. Sensitivity of sequence length and lr





THANKS !

Contact Us:
czlv24@m.fudan.edu.cn
zhengxq@fudan.edu.cn

Code is available at:
https://github.com/Lvchangze/Dendritic-Localized-Learning