A Appendix

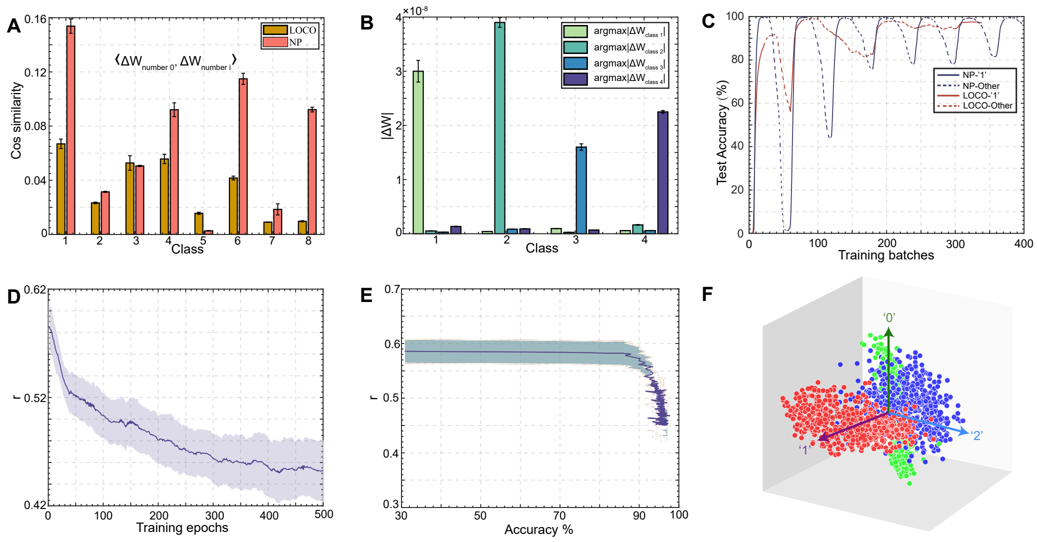


Figure S1: **LOCO restricts training in the orthogonal space.** **(A)** Weight modification implemented by LOCO is more orthogonal between different classes than that of NP. Each bar indicates the cosine similarity between weight modifications when learning the digit 0 and those made when learning the digits from 1 to 8. A lower cosine similarity suggests that the modifications in weights are more orthogonal to each other. **(B)** The most relevant weight modification in each class is selected. $argmax|\Delta W_{class\ i}|$ indicates the weights with the largest magnitude of change during the training of the i-th category. Bar graph indicates that each class has its own most relevant weight, suggesting that LOCO implements training in orthogonal space of weights. **(C)** Accuracy in recognizing digit 1 along the train process. LOCO exhibits smaller interference between training of different digits compared to NP. **(D)** The cosine similarities of activities between different categories r (see Appendix) decreases during training, suggesting that neuronal activity tends to be more orthogonal along the course of training. **(E)** r decreases as accuracy increases. This suggests that neuronal activity tends to be more orthogonal as accuracy increases. **(F)** Activity representations of three categories in the hidden layer during the training process. After dimensionality reduction via PCA, the activity representations of the three categories (color-coded) are positioned in a way tend to be orthogonal to each other. Each data point represents the average activity per batch.

Method	Main rules	Accuracy	
		MNIST	NETtalk
DiehlDiehl & Cook (2015)	STDP	91.20±1.69%	—-
SOMHazan et al. (2018)	Hebb	91.07±1.79%	—-
BBTZhang et al. (2018)	Balanced V+STDP	93.67±0.40%	84.26±0.20%
SNN-SBPZhang et al. (2021a)	STDP+SBP	95.14±0.12%	85.58±0.10%
BRPZhang et al. (2021b)	RP	95.42±0.13%	80.33±4.52%
NRRJia et al. (2021)	RP+BPTT	94.19±0.11%	77.73±0.46%
NP	Perturbation	95.14±0.10%	84.07±0.52%
LOCO	LOCO+Perturbation	**96.40±0.07%**	**86.01±0.35%**
LOCO (10layers)	LOCO+Perturbation	93.80±0.12%	82.00±1.95%

Table S1: **The comparison of accuracy for different algorithms.** For algorithms without specific annotations regarding their parameter information, the quantity of parameters is identical. For the MNIST dataset, the network architecture is structured as 784-500-10. For the NETtalk dataset, the network configuration is 189-500-26.

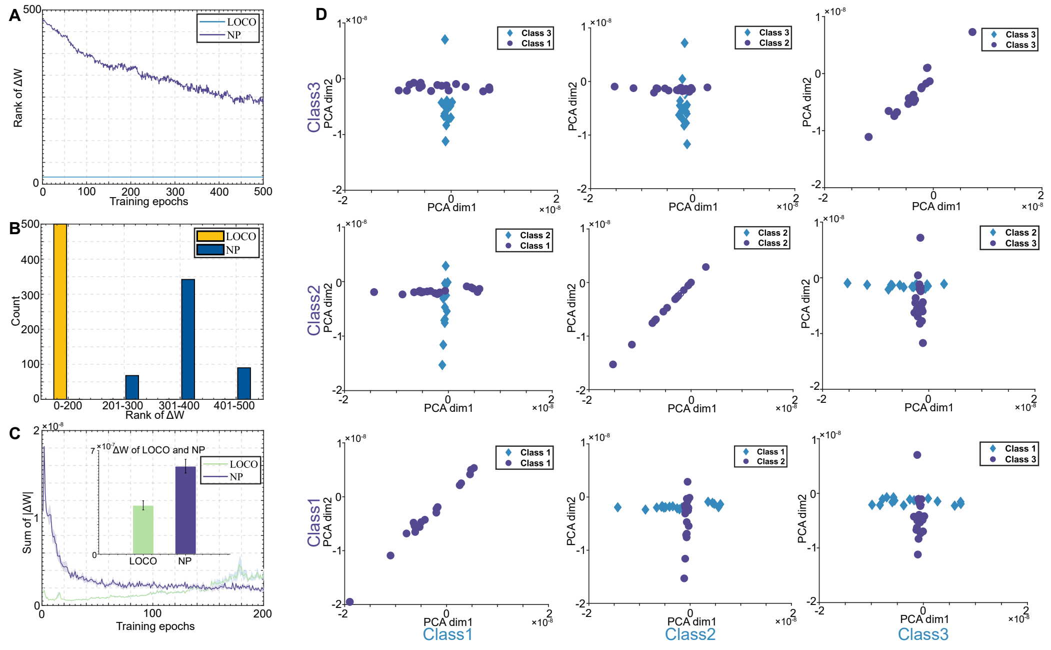
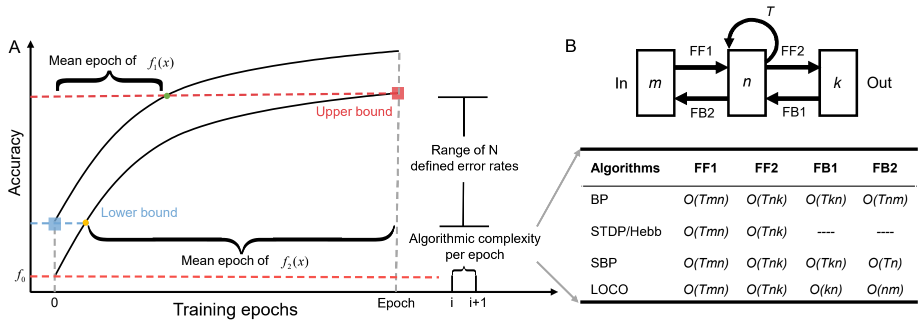


Figure S2: **LOCO restricts training in the low-rank space.** **(A)** Weight modification implemented by LOCO is in a subspace with lower rank than that of NP. **(B)** The distribution of rank of the weights modification space for LOCO and NP. **(C)** The magnitude of weight changes in LOCO is smaller than that in NP. This implies that LOCO is more stable and energy-efficient. **(D)** Weight modifications for recognizing a pair of digits (1-2, 2-3, and 1-3) are dimension reduced via PCA to be plotted in a 2D plane. The weight modifications for each class are clustered around a line in the plane, indicating that weight modifications are occurring within a low-dimensional space. In addition, weight modifications for different classes tend to be orthogonal to each other.

Phases	Operation	Time Complexity
Forward	propagation	$O(bn^2)$
	add direction X	$O(bn)$
	update $\mathbf{U}$	$O(kpnc)$
	nearest $\mathbf{u}_i$	$O(bnc)$
	delete $\mathbf{u}_i$	$O(bc)$
	calculate P_l	$O(bc^2n + bc^3 + bnc^2)$
Backward	weight grad	$O(bn^2 + bnc + bcn + bn^2)$
	propagation	$O(bn^2)$

Table S2: **Time complexity of introducing projection matrix.** The time complexity of MLP is $O(bn^2)$ and the time complexity introduced by projection matrix does not exceed $O(bn^2)$. b is batch size. $b = 32$ in all experiment settings. $n = 500$ is the number of neurons in the hidden layer. $p = 50$is buffer size of direction X. $c = 10$ is the number of cluster centers and $k = 10$ is the number of iterations of k-means.



Algorithms	FF1	FF2	FB1	FB2
BP	O(Tmn)	O(Tnk)	O(Tkn)	O(Tnm)
STDP/Hebb	O(Tmn)	O(Tnk)	----	----
SBP	O(Tmn)	O(Tnk)	O(Tkn)	O(Tn)
LOCO	O(Tmn)	O(Tnk)	O(kn)	O(nm)

Figure S3: **The convergence efficiency during learning. (A)** Diagram depicting calculation of the mean epoch in N levels ($N = 5$) for curves of $f_1(x)$ and $f_2(x)$ to achieve some defined accuracy levels between an upper bound and a lower bound. The upper bound and lower bound represent the highest and lowest values of the accuracy curves at the beginning and the end of learning epochs, respectively, among the algorithms under comparison (see Methods for more details). The convergence efficiency was calculated by averaging the epochs at five accuracy levels (including upper and lower bounds). **(B)** Algorithmic complexity $O(\cdot)$ in each epoch during learning. It includes feedforward propagation (FF) and feedback propagation (FB). m, n, and k are numbers of neurons in network's input, hidden, and output layers, respectively. The compared algorithms include BP, STDP (or Hebb), self-BP (SBP) and LOCO.

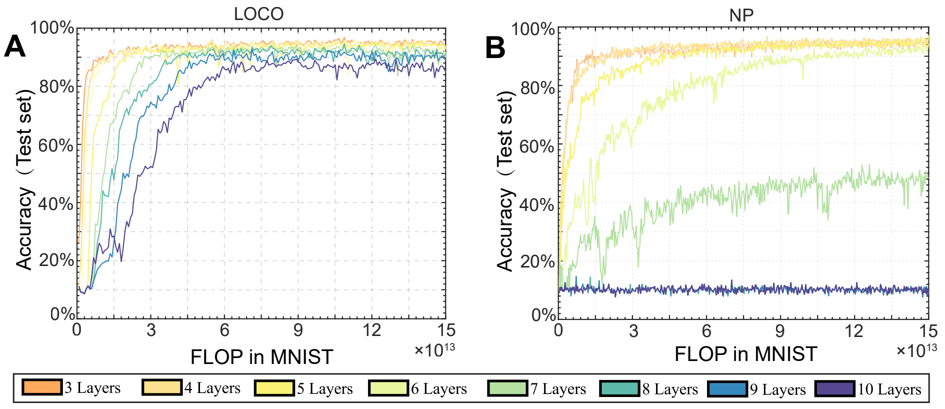


Figure S4: **FLOP efficiency during learning.** LOCO demonstrates faster convergence compared to NP across networks with 3 to 10 layers, highlighting its high efficiency when considering equivalent computational workloads. The FLOP calculation includes forward propagation, backpropagation, projection matrix computation, and the projection process for each layer.

A.1 MATHEMATICAL BASIS FOR NP

Node perturbation is unbiased, but noisy. We first introduce and formulate NP in the context of deep spiking learning. Consider a deep feedforward network

$$s^l(t) = f_l\left(W^l s^{l-1}(t)\right), l = 1, 2, ..., L \tag{7}$$

where $s^0(t)$ and $s^L(t)$ are the input spike and the output spike respectively, and $f_l(\cdot)$ represents the propagation dynamics of the Leaky Integrate-and-Fire (LIF) neurons. which is an element-wise function. Adding a small Gaussian perturbation $\sigma\xi^l$ to each layer gives us a perturbed network.

$$\tilde{s}^l(t) = f_l\left(W^l \tilde{s}^{l-1}(t) + \sigma\xi^l\right), l = 1, 2, ..., L \tag{8}$$

where $\left\langle \xi^k {\xi^l}^T \right\rangle = \delta_{kl} I_k$, with I_k the identity matrix in the appropriate dimension, and $\sigma \ll 1$. Under the loss function $\ell(s^L, s^0)$ which is defined consistently with that in Eq. 27, the node perturbation update is

$$\Delta W_l^{\text{NP}} = -\frac{\eta}{\sigma}(\sigma\xi^l)(\tilde{\ell}(\tilde{s}^L, s^0) - \ell(s^L, s^0)){\mathbf{x}_{l-1}}^T \tag{9}$$

If the perturbation decreases the error (i.e., $\tilde{\ell} - \ell < 0$), the weights are shifted towards the direction of the perturbation (ξ^l), and vice versa. Importantly, in this update rule, the network only needs to know how much the loss changes when a perturbation is added to the network. This is in contrast to SGD and most of its biologically plausible variants, which require a supervised signal telling what the correct answer was. It is straightforward to show that, at the small perturbation limit $\sigma \to 0$

$$\tilde{\ell} = \ell + \sigma \sum_{l=1}^{L} \sum_{t=1}^{T} \frac{\partial \ell}{\partial h_{l,t}} \xi^{l,t} \tag{10}$$

where

$$h_{l,t} = W^l s^{l-1}(t) \tag{11}$$

Thus, denoting in the small limit the NP update becomes

$$\begin{aligned}
\Delta W_l^{\text{NP}} &= \sum_{t=1}^{T} \Delta W_{l,t}^{\text{NP}} \\
&= \sum_{t=1}^{T} -\tfrac{\eta}{\sigma}\xi^{l,t}(\tilde{\ell}(\tilde{s}^L, s^0) - \ell(s^L, s^0)){s^{l-1}(t)}^T \\
&= \sum_{t=1}^{T} -\tfrac{\eta}{\sigma}\xi^{l,t}\left(\sigma \sum_{k=1}^{L}\sum_{t'=1}^{T} \tfrac{\partial \ell}{\partial h_{l,t'}}\xi^{k,t'}\right){s^{l-1}(t)}^T \\
&= -\eta \sum_{t=1}^{T} \xi^{l,t} \sum_{k=1}^{L}\sum_{t'=1}^{T} \tfrac{\partial \ell}{\partial h_{l,t'}}\xi^{k,t'}{s^{l-1}(t)}^T \\
&= -\eta \sum_{t=1}^{T}\sum_{k=1}^{L}\sum_{t'=1}^{T} \tfrac{\partial \ell}{\partial h_{l,t'}}\xi^{l,t}\xi^{k,t'}{s^{l-1}(t)}^T
\end{aligned} \tag{12}$$

As mentioned above, taking expectation over ξ gives us

$$\left\langle \Delta W_l^{\text{NP}} \right\rangle_\xi = -\eta \sum_{t=1}^{T} \frac{\partial \ell}{\partial h_{l,t}} {s^{l-1}(t)}^T \tag{13}$$

On the other hand, the SGD is given by

$$\begin{aligned}\Delta W_l^{SGD} &= -\eta \tfrac{\partial \ell}{\partial W_l} \\ &= -\eta \sum_{t=1}^{T} \tfrac{\partial \ell}{\partial h_{l,t}} \tfrac{\partial h_{l,t}}{\partial W_l} \\ &= -\eta \sum_{t=1}^{T} \tfrac{\partial \ell}{\partial h_{l,t}} s^{l-1}(t)\end{aligned} \tag{14}$$

Therefore, NP is unbiased against SGD. Moreover, because SGD with i.i.d. samples is unbiased against the true gradient, NP is also unbiased against it. To simplify the model, the perturbation introduced in each layer during each propagation is identical.

A.2 Convergence for NP and LOCO

For other synaptic plasticity rules, there is often a lack of convergence guarantees at the network level. However, given that NP provide an unbiased estimation of gradients, convergence can be assured under conditions of problem convexity. The LOCO algorithm, derived from NP and augmented with additional constraints, does not alter the fundamental convergence properties of the algorithm. A rigorous proof of the convergence properties of NP is provided below.

The classical descent lemma uses a Taylor expansion to study how SGD reduces the loss at each optimization step. There is a Descent Lemma. Let $\ell(\theta)$ be convex and ℓ-smooth. To simplify the expression, θ is subset of parameters with the same color like in Fig. 1A. B is a batch of data. For any unbiased gradient estimate $\mathbf{g}(\theta, \mathrm{B})$

$$\mathrm{E}[\ell(\theta_{t+1})|\theta_t] - \ell(\theta_t) \leq -\eta_{NP} \|\nabla \ell(\theta_t)\|^2 + \frac{1}{2}\eta_{NP}^2 \ell \cdot \mathrm{E}[\|\mathbf{g}(\theta, \mathrm{B}_t)\|^2] \tag{15}$$

Unbiased gradient estimate means $\mathrm{E}[\mathbf{g}(\theta, \mathrm{B})] = \nabla \ell(\theta)$.The descent lemma also shows that to guarantee loss decrease, one needs to choose the learning rate as

$$\eta_{NP} \leq \frac{2\|\nabla \ell(\theta_t)\|^2}{\ell \cdot \mathrm{E}[\|\mathbf{g}(\theta, \mathrm{B})\|^2]} = \frac{1}{r}\frac{2}{\ell} = \frac{1}{r}\eta_{SGD} \tag{16}$$

where r is local r-effective rank. Sadhika Malladi et al. (2024) extensively discuss the convergence speed of NP in comparison to SGD (stochastic gradient descent), elucidating how the presence of low effective rank, denoted as r, ensures that NP do not suffer from undue slowness. This is attributed to the fact that the dimensionality of the search space required by the task, r, is significantly smaller than the number of model parameters d, i.e., $r \ll d$. Generally, the value of r is task-dependent. This paper proposes that not only Hessian matrix has low effective ranks but also the gradient is low rank. We hypothesize that the upper bound for the ranks of both is r. The rank of the gradient is the dimensionality of the space spanned by the principal components of a certain class of input data. Empirical findings prove that the rank associated with the gradient are also markedly small (Fig. 3A, B, H).

Since NP provide an unbiased estimation of the gradient and represent the direction of steepest descent, the convergence properties of LOCO can be assessed by calculating the angle between the gradient estimated by NP and the weight modification direction in LOCO. For LOCO, the modification only involves the addition of a projection matrix P_l in front of the gradient. This aspect makes it straightforward to demonstrate the convergence properties of the LOCO algorithm.

$$\begin{aligned}\cos\left\langle \Delta W_l^{\mathrm{np}}, \Delta W_l^{LOCO} \right\rangle &= \tfrac{tr\left(\Delta W_l^{\mathrm{np}} \cdot \Delta W_l^{LOCO}\right)}{\left\|\Delta W_l^{\mathrm{np}}\right\|_F \left\|\Delta W_l^{LOCO}\right\|_F} \\ &= \tfrac{tr\left(\Delta W_l^{\mathrm{np}} P_l \Delta W_l^{\mathrm{np}}\right)}{\left\|\Delta W_l^{\mathrm{np}}\right\|_F \left\|\Delta W_l^{LOCO}\right\|_F} \\ &> 0\end{aligned} \tag{17}$$

Given that P_l is a projection matrix, it is necessarily a positive definite matrix. Consequently,$\cos\left\langle \Delta W_l^{\mathrm{NP}}, \Delta W_l^{LOCO} \right\rangle > 0$, indicating that the angle between the gradient estimated by NP and the weight modification direction in LOCO is within 90°. Moreover, as NP provides an unbiased estimation of the true gradient, the direction of weight modification in LOCO always forms an acute angle with the true gradient. This alignment continues to satisfy the conditions for convergence.

According to the Descent Lemma, it is still possible to arrive at the conclusion

$$\mathrm{E}[\ell(\theta_{t+1})|\theta_t] - \ell(\theta_t) \leq -\eta_{LOCO}\|\nabla\ell(\theta_t)\|^2 + \frac{1}{2}\eta_{LOCO}^2\ell \cdot \mathrm{E}[\|\mathbf{g}(\theta, \mathrm{B}_t)\|^2] \tag{18}$$

P is a projection matrix. The descent lemma also shows that to guarantee loss decrease, one needs to choose the learning rate as

$$\eta_{LOCO} \leq \frac{2\|\nabla\ell(\theta_t)\|^2}{\ell \cdot \mathrm{E}[\|\mathbf{g}(\theta, \mathrm{B})\|^2]} = \frac{1}{r-o}\frac{2}{\ell} \tag{19}$$

see Eq. 31 for proof. r denotes effective rank of loss. o denotes the number of overlapped dimensionality between the complement of CO space and r-dimensional space. The value of o ranges from 0 to $c-1$. c denotes the dimensions kept by cluster orthogonal weight modification. Previous studies have discovered that the effective rank is significantly lower than the number of parameters in the model Malladi et al. (2024). This phenomenon also accounts for the higher practical convergence efficiency of NP compared to its theoretical optimization efficiency. The size of the effective rank is task-dependent. For instance, experiments on the MNIST task have revealed that the effective rank is approximately 20.

A.3 Proof of Unbiasedness of LOCO

Let $\Delta W^{LOCO}(B_t)$ denote the LOCO weight update and $\Delta W^{NP}(B_t)$ denote the NP weight update for a given batch B_t. We have the relation:

$$\Delta W^{LOCO}(B_t) = P(B_t) \cdot \Delta W^{NP}(B_t) \tag{20}$$

where $P(B_t)$ is a projection matrix associated with batch B_t. Each projection matrix $P(B_t)$ is rank-deficient for any individual batch but, when accumulated across multiple batches, the matrices $P(B_1), P(B_2), ..., P(B_n)$ collectively span the full weight space, resulting in full rank coverage.

Proof

Consider an arbitrary training period with n batches, $B_1, B_2, ..., B_n$, each associated with a projection matrix $P(B_t)$ applied to the NP weight update. The LOCO weight update across these n batches is given by

$$\frac{1}{n}\sum_{t=1}^{n}\Delta W^{LOCO}(B_t) = \frac{1}{n}\sum_{t=1}^{n}P(B_t) \cdot \Delta W^{NP}(B_t) \tag{21}$$

We examine the expectation:

$$E\left[\frac{1}{n}\sum_{t=1}^{n}\Delta W^{LOCO}(B_t)\right] = E\left[\frac{1}{n}\sum_{t=1}^{n}P(B_t) \cdot \Delta W^{NP}(B_t)\right] \tag{22}$$

Since we assume that $P(B_t)$ independently projects in different directions across different batches, we can regard the cumulative effect of $P(B_t)$ as approximating the role of an identity matrix. Thus, we have $E\left[P(B_t)\right] \approx I$. where I is the identity matrix in the weight space. Consequently, we can rewrite the expectation as

$$E\left[\Delta W^{LOCO}\right] \approx \frac{1}{n}\sum_{t=1}^{n} P(B_t)\cdot \Delta W^{NP}(B_t) = E\left[\Delta W^{NP}(B_t)\right] \tag{23}$$

Consequently ΔW^{LOCO} is approximately an unbiased estimator of gradient.

A.4 Convergence speed for LOCO

The above analysis delves into the convergence properties of both NP and LOCO , providing upper bounds for the learning rates of each optimization algorithm. From this analysis, it is possible to establish a relationship between the upper bounds of the learning rates for these two optimization methods. This relationship elucidates how the constraints and modifications inherent in LOCO, relative to NP, influence the maximum permissible learning rates for ensuring convergence within these frameworks.

$$\eta_{LOCO} = \gamma\eta_{NP} \tag{24}$$

where $\gamma = \frac{r}{r-o} > 1$. In the MNIST experiments conducted for this paper for example, r (effective rank) is approximately 10 to 30, and $r-o$ (the dimension of the projection space after LOCO constraints) is around 1 to 20. Consequently, the coefficient γ is around 1.5 to 10. Therefore, the learning rate ensuring the learning rate of the LOCO algorithm is greater than that required for the NP algorithm. This implies that the convergence efficiency of the LOCO algorithm should also be higher than that of the NP algorithm. The specific proofs of their respective convergence rates are presented below.

Based on the formula Eq. 15,Plugging in $\mathrm{E}[\|\mathbf{g}(\theta, \mathrm{B}_t)\|^2] = \|\nabla\ell(\theta_t)\|^2 + \frac{1}{B} tr\left(\sum(\theta_t)\right)$ and selecting a learning rate $\eta < \frac{1}{\ell}$ yields

$$\mathrm{E}[\ell(\theta_{t+1})|\theta_t] \leqslant \ell(\theta_t) - \frac{\eta_{NP}}{2}\|\nabla\ell(\theta_t)\|^2 + \frac{\eta_{NP}^2\ell}{2B}\cdot tr\left(\sum(\theta_t)\right)$$

Since $\ell(\theta_t)$ is μ-PL Malladi et al. (2024) satisfy $\frac{1}{2}\|\nabla\ell(\theta_t)\|^2 \leqslant u\left(\ell(\theta_t) - \ell^*\right)$, we get

$$\mathrm{E}[\ell(\theta_{t+1})|\theta_t] \leqslant \ell(\theta_t) - \eta_{NP}u\left(\ell(\theta_t) - \ell^*\right) + \frac{\eta_{NP}^2\ell}{2B}\cdot tr\left(\sum(\theta_t)\right)$$

Since $tr\left(\sum(\theta_t)\right) \leqslant \alpha\left(\ell(\theta_t) - \ell^*\right)$, we have

$$\mathrm{E}[\ell(\theta_{t+1})|\theta_t] \leqslant \ell(\theta_t) - \eta_{NP}u\left(\ell(\theta_t) - \ell^*\right) + \frac{\eta_{NP}^2\ell\alpha}{2B}\cdot\left(\ell(\theta_t) - \ell^*\right)$$

Altogether,

$$\mathrm{E}[\ell(\theta_{t+1})|\theta_t] - \ell^* \leqslant \left(1 - \eta_{NP}u + \frac{\eta_{NP}^2\ell\alpha}{2B}\right)\left(\mathrm{E}[\ell(\theta_t)] - \ell^*\right)$$

Choosing $\eta_{NP} = \min\left(\frac{1}{\ell}, \frac{uB}{\ell\alpha}\right)$, we obtain

$$\mathrm{E}[\ell(\theta_{t+1})|\theta_t] - \ell^* \leqslant \left(1 - \min\left(\frac{u}{2\ell}, \frac{u^2B}{2\ell\alpha}\right)\right)\left(\mathrm{E}[\ell(\theta_t)] - \ell^*\right)$$

Therefore we reach a solution with $\mathrm{E}[\ell(\theta_t)] - \ell^* \leqslant \varepsilon$ after

$$\begin{aligned} t &\approx \max\left(\tfrac{2\ell}{u}, \tfrac{2\ell\alpha}{u^2B}\right)\log\left(\tfrac{\ell(\theta_0)-\ell^*}{\varepsilon}\right) \\ &= O\left(\left(\tfrac{\ell}{u} + \tfrac{\ell\alpha}{u^2B}\right)\log\tfrac{\ell(\theta_0)-\ell^*}{\varepsilon}\right) \end{aligned}$$

iterations.

By Eq. 15, LOCO with $\eta_{LOCO} = \gamma\eta_{NP}$ yields

$$\begin{aligned}&\mathrm{E}[\ell(\theta_{t+1})|\theta_t] - \ell(\theta_t) \leqslant \\ &\gamma\left[-\eta_{NP}\|\nabla\ell(\theta_t)\|^2 + \tfrac{1}{2}\gamma\eta_{NP}^2\ell\cdot\mathrm{E}[\|\mathbf{g}(\theta,\mathrm{B}_t)\|^2]\right]\end{aligned}$$

As in the proof for NP, choosing $\eta_{NP} < \frac{1}{\ell}$ yields

$$\begin{aligned}&\mathrm{E}[\ell(\theta_{t+1})|\theta_t] - \ell(\theta_t) \leqslant \\ &\gamma\left[-\tfrac{\eta_{NP}}{2}\|\nabla\ell(\theta_t)\|^2 + \tfrac{\gamma\eta_{NP}^2\ell}{2B}\cdot tr\left(\sum(\theta_t)\right)\right]\end{aligned}$$

Therefore under μ-PL and the $tr\left(\sum(\theta_t)\right) \leqslant \alpha\left(\ell(\theta_t) - \ell^*\right)$ assumption we obtain

$$\begin{aligned}&\mathrm{E}[\ell(\theta_{t+1})|\theta_t] - \mathrm{E}[\ell(\theta_t)] \leqslant \\ &\gamma\left[\left(-\eta_{NP}u + \tfrac{\gamma\eta_{NP}^2\ell\alpha}{2B}\right)(\mathrm{E}[\ell(\theta_t)] - \ell^*)\right]\end{aligned}$$

$$\begin{aligned}&\Rightarrow \mathrm{E}[\ell(\theta_{t+1})|\theta_t] - \ell^* \leqslant \\ &\left(1 - \gamma\left(\eta_{NP}u + \tfrac{\gamma\eta_{NP}^2\ell\alpha}{2B}\right)\right)(\mathrm{E}[\ell(\theta_t)] - \ell^*)\end{aligned}$$

Choosing $\eta_{NP} = \min\left(\frac{1}{\ell}, \frac{uB}{\gamma\ell\alpha}\right)$ yields

$$\mathrm{E}[\ell(\theta_{t+1})|\theta_t] - \ell^* \leqslant \left(1 - \gamma\min\left(\frac{u}{2\ell}, \frac{u^2B}{2\gamma\ell\alpha}\right)\right)(\mathrm{E}[\ell(\theta_t)] - \ell^*)$$

Therefore we reach a solution with $\mathrm{E}[\ell(\theta_t)] - \ell^* \leqslant \varepsilon$ after

$$\begin{aligned}&t \approx \gamma^{-1}\max\left(\tfrac{2\ell}{u}, \tfrac{2\ell\alpha}{u^2B}\right)\log\left(\tfrac{\ell(\theta_0)-\ell^*}{\varepsilon}\right) \\ &= O\left(\gamma^{-1}\left(\tfrac{\ell}{u} + \tfrac{\ell\alpha}{u^2B}\right)\log\tfrac{\ell(\theta_0)-\ell^*}{\varepsilon}\right)\end{aligned}$$

iterations.

Consequently, based on the lower bounds of iteration counts for LOCO and NP, the relationship between the convergence times of the two algorithms can be deduced.

$$t_{LOCO} \approx \gamma^{-1}t_{NP}$$

Specifically, when r is 30 and c is 10, resulting in γ is around 30/21, the LOCO algorithm is observed to be around 1.5 times faster than the NP algorithm. Empirically, this ratio was verified. Through Eq. 19, the size of γ can be estimated. By analyzing the gradient information of the second layer in the 4-layer MNIST task, we obtained that γ is [1.60 ± 0.21; SD, n= 5]. This is consistent with the results observed in the experiment (Fig. 2G).

A.5 NETWORK MODELS OF SNNS

The architecture of the spiking neural network (SNN) is characterized by a multi-layered, fully connected structure. Following the encoding process, the spike information is fed into Leaky Integrate-and-Fire (LIF) neurons. Neurons in one layer are interconnected with the subsequent layer through fully connected synaptic. Upon the excitation of the LIF neurons, the emitted spike signals are weighted by the synaptic, forming postsynaptic membrane currents. These currents propagate to the neurons in the next layer, altering their membrane potentials. The propagation dynamics are defined by the following equation

$$I_{syn,i}^{l}(t) = \sum_{j} w_{ij} s_j^{l-1}(t) \tag{25}$$

The equation represents the propagation dynamics of spikes from layer $l-1$ to layer l within a spiking neural network. Here, $I_{syn,i}^{l}(t)$ denotes the postsynaptic membrane current at time t for the i-th neuron in layer l. w_{ij} represents the synaptic weight between presynaptic neuron i and postsynaptic neuron j. $s_j^{l-1}(t)$represents the spike emitted at time t by the j-th neuron in the preceding layer $l-1$.

A.6 The LIF propagation in SNNs

The dynamic of LIF use clock driven LIFNode in spikingjelly Fang et al. (2023). The spikes in presynaptic neurons trigger postsynaptic potentials, which are dynamically integrated and generate spikes in the postsynaptic neuron when the firing threshold is reached. The membrane potential $V(t)$ is calculated as follows

$$\begin{aligned} &\tau_m \tfrac{dV(t)}{dt} = -\left(V(t) - V_{reset}\right) + I_{syn}(t) \\ &s(t) = \begin{cases} 1 & if(V(t) \geq V^{Tr}) \\ 0 & if(V(t) < V^{Tr}) \end{cases} \end{aligned} \tag{26}$$

where τ_m is membrane time constant, I_{syn} is the presynaptic current. The membrane potential $V(t)$ will be reset when crossing threshold V^{Tr} and clamped to the resting potential V_{rest}.

A.7 Static data encoding

We use a commonly used coding method: direct coding Wu et al. (2019); Rathi & Roy (2021) to encode the static images. Direct coding treats the first layer of the network as the coding layer. This approach significantly reduces the simulation length while maintaining accuracy. The direct encoding operation is divided into two steps for a normalized image $x \in [0,1]^{W\times H}$. First, the first layer of the network receives external stimuli and transforms them into a constant input current. Subsequently, this constant current is transformed into a spike sequence $\{0,1\}^{T\times(W\times H)}$ by LIF neurons, as described in Eq. 26.

A.8 Network output and loss functions

In the spiking neural network, the output of the network is decoded using a rate decoder. This involves calculating the average firing rate of the neurons in the output layer to determine the final output of the network. The loss function employed is the Mean Squared Error (MSE). In the LOCO algorithm, the feedback component is represented by a scalar Temporal Difference error(TD). This scalar quantifies the change in the evaluation metric across twice forward propagation.

The loss of neural network l is calculated as follows

$$\begin{aligned} l &= \tfrac{1}{2N} \sum_i \left(\tfrac{1}{T} \sum_{t=1}^{T} I_{spikes,i}(t) - y_i \right)^2 \\ \tilde{l} &= \tfrac{1}{2N} \sum_i \left(\tfrac{1}{T} \sum_{t=1}^{T} \tilde{I}_{spikes,i}(t) - y_i \right)^2 \end{aligned} \tag{27}$$

Two forward propagations yield two distinct losses: one is the loss from precise propagation, denoted as l, and another is the loss following the introduction of perturbation, denoted as $\tilde{l}$. $\sum\limits_{t=1}^{T} I_{spikes,i}(t)$ represents the output value of the i-th neuron in the output layer of the neural network, while $\sum\limits_{t=1}^{T} \tilde{I}_{spikes,i}(t)$ represents the output value of the same neuron after the introduction of the perturbation during the second propagation. y_i symbolizes the target output of the i-th neuron in the output layer.

Temporal Difference (TD) error is utilized to guide the learning process in the LOCO algorithm. The computation formula for the TD error is as follows

$$TD = \tilde{l} - l \tag{28}$$

The Temporal Difference (TD) error characterizes the change in the network's performance following the introduction of a perturbation, relative to its performance without the perturbation. A positive TD indicates that the perturbation has improved the network's performance, whereas a negative TD suggests a decrease in performance.

A.9 Network models of ANNs

The architecture of an Artificial Neural Network (ANN) is characterized by a multi-layered, fully connected structure. The number of neurons in the input layer corresponds to the dimensionality of the input data. Information is propagated to the subsequent layer of neurons after being weighted by the synaptic. The neurons in the next layer aggregate this postsynaptic membrane information and use an activation function to determine the value transmitted to the subsequent layers of the neural network. In the algorithm proposed in this paper, the ANN undergoes two propagation passes. The propagation equation is defined by the following formula

$$\begin{aligned} \mathbf{x}_l &= f(W_l^T \mathbf{x}_{l-1}), l = 1, 2, ..., L \\ \tilde{\mathbf{x}}_l &= f(W_l^T \tilde{\mathbf{x}}_{l-1}) + \sigma \xi_l, l = 1, 2, ..., L \end{aligned} \tag{29}$$

where $\mathbf{x}_{l-1} \in R^{n\times 1}$ represents the input data for layer l. $W_l \in R^{n\times n}$ denotes the weights of layer l, which correspond to synaptic strengths. The function $f(\cdot)$ signifies the activation function, with the ReLU (Rectified Linear Unit) function being utilized in this paper. $\mathbf{x}_l \in R^{n\times 1}$ indicates the output values of the neural network at layer l during the first precise propagation. $\tilde{\mathbf{x}}_l \in R^{n\times 1}$ represents the output values of the neural network at layer l during the second propagation, after the introduction of a perturbation.

It is important to note that the $\mathbf{x}_{l-1}$ mentioned here directly corresponds to $\mathbf{x}_{l-1}$ in Eq. 2. The weight update formula is the same as that in Eq. 2. Additionally, the proofs for convergence and the demonstration of convergence speed follow the same rationale as previously outlined.

A.10 Definition of convergence efficiency during training

The convergence efficiency ($Effi_i$) of algorithm i during training is determined by multiplying the mean number of epochs required to reach a specified accuracy level (Fig. S1A) with the algorithmic complexity per epoch, denoted as $O(n)_i$ (Fig. S1B). For the purpose of comparing two algorithms (where $i = 1, 2$), the convergence efficiency is evaluated using the following formula:

$$Effi_i = \frac{1}{N} \sum_{l=1}^{N} Argmin\left(f_i(x) = Acc_l\right) \times O(n)_i \tag{30}$$

where $Argmin(\cdot)$ is the argument of the minimum, $f_i(x)$ is the accuracy curve with input epoch x, $O(n)_i$ is the algorithmic complexity with n depicting the number of parameters, and N is the number of predefined accuracy levels ($N = 5$). Acc_l is selected out from a range of accuracy, with a upper bound of $Min(Max(f_1), Max(f_2))$, defined as the relatively lower maximal accuracy of $f_1(x)$ and $f_2(x)$, and also with an lower bound of $Max(Min(f_1), Min(f_2), f_0)$, defined as the relatively higher minimal accuracy among $f_1(x)$, $f_2(x)$, and an additionally predefined accuracy $f_0 = 0.8$ (the minimally acceptable accuracy).

A.11 MNIST and NETtalk datasets

MNIST dataset LeCun (1998) comprises 60,000 training and 10,000 test samples, each with a size of 28×28 pixels. This dataset covers 10 classes of handwritten digits ranging from 0 to 9. NETtalk dataset Sejnowski & Rosenberg (1987) consists of 5,033 training and 500 test samples. Each input sample represents an aligned English word through a 189-dimension vector, with individual letters encoded as one-hot vectors of 27 dimensions. Phonetic outputs are represented by multi-hot 26-dimension vectors, encompassing 21 sequential pronunciation features (e.g., ”Labial”, ”Dental”, ”Alveolar”, etc.) and five stress features (e.g., ”¡”, ”¿”, ”0”, ”1”, ”2”). Excluding punctuation, the total number of phonetic representation classes with stresses is 116.

A.12 Accuracy definition

In our experiments, the accuracy of MNIST is defined as the number of correctly identifying samples dividing by the number of all samples. Different from it, the accuracy of NETtalk is defined as the cosine similarity distance of identified phonemes and real phonemes for the consideration of the multiphonemes in the same sample.

A.13 Definition of r for measuring orthogonality at the activity level

We definite the average cosine similarities between different categories as r to measure orthogonality at the activity level. Small r indicating that different tasks is orthogonal on the activity level. The definition is,

$$r = \frac{1}{10^2} \sum_{i,j}^{10} \frac{x_i \cdot x_j}{\|x_i\| \cdot \|x_j\|}$$

x_i is the activity of hidden neurons when recognising i category.

A.14 Relationship between neural activity and weight modification

In the LOCO algorithm, the effect of low-dimensional and orthogonal activity dynamics in the brain can be explained as restricting synaptic modifications to occur within a low-dimensional and orthogonal space. This enhances the efficiency of perturbation-based optimization and improves the stability of the brain. We note that for the NP algorithm, ΔW_l^{NP} is always pointing to the same direction as the input vector x (Eq. 9). Therefore, if x are low-dimensional and orthogonal to each other, the ΔW_l^{NP} would hold the same property. Consequently, low-dimensionality and orthogonality of neural activity will lead to the same property of weight modification. Further more, with the analysis of Convergence speed for LOCO above, low-dimensionality will lead to higher convergence efficiency.

A.15 Time complexity of training process

Training process only requires weight adjustments as Eq. 2, if the computation of P is finished in the forward propagation. $\xi_l(P_l\tilde{\mathbf{x}}_{l-1})^T$has been calculated, the weight modification is just the multiply of TD error $(\tilde{\ell}(\tilde{s}^L, s^0) - \ell(s^L, s^0))$ with $\xi_l^i(P_l\tilde{\mathbf{x}}_{l-1})^{jT}$. Considering each weight as a computational unit, and based on Amdahl's Law, the maximum speedup rate S is Ln^2. The time complexity of the optimization process is only $O(1)$ (independent of the number of neural network parameters), meaning that all weights can be modified simultaneously. Consequently, the training time does not increase with the scale of the network. This method is suitable for distributed training of large models and can achieve a high degree of parallel efficiency.

$$S = 1/\left((1-a) + a/m\right)$$

where a represent the proportion of the computation that can be parallelized, and m the number of parallel processing nodes. During the training process, all weights can simultaneously compute the weight updates and execute the modifications, thus the proportion of parallel computation is 1, i.e., $a = 1$. Considering each weight as a computational unit, and taking a fully connected network as an example, the number of weights is Ln^2, where L is the number of layers and n is the number of neurons in each hidden layer. Therefore, $m = Ln^2$. Hence, the maximum speedup of the training process through parallelization is Ln^2.

A.16 The upper bond of learning rate in LOCO and NP

Lemma 1 Let B be a random minibatch of size B. Then, the gradient norm of LOCO is

$$E\left[\left\|g^{LOCO}\left(\theta,\mathrm{B}\right)\right\|^2\right]=(r-o)\cdot E\left[\left\|\nabla\ell\left(\theta,\mathrm{B}\right)\right\|^2\right]$$

the gradient norm of NP is

$$E\left[\left\|g^{NP}\left(\theta,\mathrm{B}\right)\right\|^2\right]=r\cdot E\left[\left\|\nabla\ell\left(\theta,\mathrm{B}\right)\right\|^2\right]$$

θ is a unit set of parameters mentioned in Eq. 3. r is r-effective rank. o denotes the number of overlapped dimensionality between the complement of CO space and r-dimensional space.

Proof of Lemma 1. Similar with the proof in Malladi et al. (2024), We first note that in the $\sigma\to 0$ limit, we have

$$g\left(\theta,\mathrm{B}\right)=\frac{1}{B}\sum_{(x,y)\in\mathrm{B}}zz^T\nabla\ell\left(\theta,\{(x,y)\}\right)$$

Taking expectation over the batch B and the z, we have $E\left[g\left(\theta,\mathrm{B}\right)\right]=\nabla\ell\left(\theta\right)$, so $\left[g\left(\theta,\mathrm{B}\right)\right]$ is an unbiased estimator of the gradient.

Computing the second moment, we get

$$E\left[g\left(\theta,\mathrm{B}\right)g(\theta,\mathrm{B})^T\right]=\frac{1}{B^2}\sum_{(x_1,y_1),(x_2,y_2)\in\mathrm{B}}E\left[\left(zz^T\nabla\ell\left(\theta,\{(x_1,y_1)\}\right)\right)\left(zz^T\nabla\ell\left(\theta,\{(x_2,y_2)\}\right)\right)^T\right]$$

Let u,v be two arbitrary vectors. We have that

$$E_z\left[zz^Tuv^Tzz^T\right]=uv^T$$

then

$$\begin{aligned}&E_z\left[zz^Tuv^Tzz^T\right]=E_z\left[z^{\otimes 4}\right](u,v)\\&=\tfrac{3d}{d+2}Sym\left(I^{\otimes 2}\right)(u,v)\\&=\tfrac{d}{d+2}\cdot u^Tv\cdot I+\tfrac{2d}{d+2}\cdot uv^T\end{aligned}$$

Therefore

$$\begin{aligned}&E\left[g\left(\theta,\mathrm{B}\right)g(\theta,\mathrm{B})^T\right]=\tfrac{1}{B^2}\sum_{(x_1,y_1),(x_2,y_2)\in\mathrm{B}}\tfrac{2d}{d+2}\cdot E\left[\nabla\ell\left(\theta,\{(x_1,y_1)\}\right)\nabla\ell(\theta,\{(x_2,y_2)\})^T\right]\\&+\tfrac{d}{d+2}\cdot E\left[\nabla\ell(\theta,\{(x_1,y_1)\})^T\nabla\ell\left(\theta,\{(x_2,y_2)\}\right)\right]I\end{aligned}$$

Next, note that when $(x_1,y_1)\neq(x_2,y_2)$, we have

$$E\left[\nabla\ell\left(\theta,\{(x_1,y_1)\}\right)\nabla\ell(\theta,\{(x_2,y_2)\})^T\right]=\nabla\ell\left(\theta\right)\nabla\ell(\theta)^T$$

and when $(x_1,y_1)=(x_2,y_2)$ we have

$$E\left[\nabla\ell\left(\theta,\{(x_1,y_1)\}\right)\nabla\ell(\theta,\{(x_2,y_2)\})^T\right]=\nabla\ell\left(\theta\right)\nabla\ell(\theta)^T+\Sigma\left(\theta\right)$$

Therefore

$$\frac{1}{B^2}\sum_{(x_1,y_1),(x_2,y_2)\in \mathrm{B}} \frac{2d}{d+2}\cdot E\left[\nabla\ell\left(\theta,\{(x_1,y_1)\}\right)\nabla\ell(\theta,\{(x_2,y_2)\})^T\right] = \nabla\ell\left(\theta\right)\nabla\ell(\theta)^T + \frac{1}{B}\Sigma\left(\theta\right)$$

and plugging this yields

$$\begin{aligned}&E\left[g\left(\theta,\mathrm{B}\right)g(\theta,\mathrm{B})^T\right] = \tfrac{2d}{d+2}\cdot\left(\nabla\ell\left(\theta\right)\nabla\ell(\theta)^T + \tfrac{1}{B}\Sigma\left(\theta\right)\right)\\ &+\tfrac{d}{d+2}I\cdot\left(\left\|\nabla\ell\left(\theta\right)\right\|^2 + \tfrac{1}{B}tr\left(\Sigma\left(\theta\right)\right)\right)\end{aligned}$$

we have

$$\begin{aligned}&E\left[\left\|g\left(\theta,\mathrm{B}\right)\right\|^2\right] = d\cdot\left(\left\|\nabla\ell\left(\theta\right)\right\|^2 + \tfrac{1}{B}tr\left(\Sigma\left(\theta\right)\right)\right)\\ &= d\cdot E\left[\left\|\nabla\ell\left(\theta,\mathrm{B}\right)\right\|^2\right]\end{aligned}$$

For NP, $z = vec\left(\xi\mathbf{x}^T\right)$. Given that the directions of inputs from each category mainly resides within a subspace, which is a low-rank r-dimensional subspace. As a result, $d = N\cdot r$. Similarly, we have

$$d = \left\{\begin{array}{ll} N\cdot r & , z^{NP} = vec\left(\xi\mathbf{x}^T\right)\\ N\cdot(r-o) & , z^{LOCO} = vec\left(\xi(P\mathbf{x})^T\right)\end{array}\right.$$

As mentioned in Eq. 3, when we just consider a unit set of parameters, we have

$$d = \left\{\begin{array}{ll} r & z{=}^{NP} vec\left(\xi_j\mathbf{x}^T\right)\\ (r-o) & z = vec\left(\xi_j(P\mathbf{x})^T\right)\end{array}\right.$$

Finally,

$$\begin{aligned}&E\left[\left\|g^{LOCO}\left(\theta,\mathrm{B}\right)\right\|^2\right] = (r-o)\cdot E\left[\left\|\nabla\ell\left(\theta,\mathrm{B}\right)\right\|^2\right]\\ &E\left[\left\|g^{NP}\left(\theta,\mathrm{B}\right)\right\|^2\right] = r\cdot E\left[\left\|\nabla\ell\left(\theta,\mathrm{B}\right)\right\|^2\right]\end{aligned} \tag{31}$$

$$\begin{aligned}&\eta_{NP} \le \tfrac{2\|\nabla\ell(\theta_t)\|^2}{\ell\cdot\mathrm{E}[\|\mathbf{g}^{\mathbf{NP}}(\theta,\mathrm{B})\|^2]} = \tfrac{1}{r}\tfrac{2}{\ell}\\ &\eta_{LOCO} \le \tfrac{2\|\nabla\ell(\theta_t)\|^2}{\ell\cdot\mathrm{E}[\|\mathbf{g}^{\mathbf{LOCO}}(\theta,\mathrm{B})\|^2]} = \tfrac{1}{r-o}\tfrac{2}{\ell}\end{aligned} \tag{32}$$

Definition 1 (Gradient Covariance). The SGD gradient estimate on a batch B has covariance $\sum\left(\theta\right) = B\left(E\left[g\left(\theta;B\right)g(\theta;B)^T\right] - g\left(\theta\right)g(\theta)^T\right)$.

A.17 Orthogonality and low-rank structures in biological neural networks

In this work, inspired by Orthogonality and low-rank structures obtained through empirical and theoretical studies, we proposed LOCO introducing them in the process of training artificial neural networks. Our results confirmed that these brain-inspired constraints indeed significantly improved the learning capability and efficiency in deep neural networks, when the BP-calculated gradients are not available. Neuronal dynamics are organized at a low-dimensional manifolds have been actively studied in recent years. However, how such feature can facilitate learning in networks is not well understood. In the LOCO algorithm, the weights modifications are based on network activity patterns. As a result, the low-dimensional activity dynamics naturally leads to low-dimensional changes in connections (see Relationship between neural activity dynamics and weight modification in mathematics in Appendix), which in turn can significantly reduce the exploration in the parameter space and facilitate learning.

Recent empirical studies have revealed that neural activities in the brain are restricted to low-dimensional manifolds and activity patterns exhibit orthogonal characteristics across different tasks, leading to the suggestion that these features may be the key to the brain's efficient learning.

Flesch et al. Flesch et al. (2022) explored how neural networks effectively encode multiple tasks through orthogonality, which is manifested by projecting the representations of different tasks onto mutually orthogonal low-dimensional manifolds. Theoretically, such orthogonal manifold designs allow different tasks to be independent in neural representation, enhancing learning efficiency in multitasking and reducing interference in learning different tasks. Indeed, functional Magnetic Resonance Imaging (fMRI) results show that human brains employs similar orthogonal representation patterns when processing different tasks, suggesting that the brain optimizes information processing using orthogonality to minimize interference between tasks. In addition, Libby and Buschman Libby & Buschman (2021) discovered that the brain reduces interference between sensation and memory of the same auditory stimuli by rotating sensory representations into orthogonal memory representations. Recording of neural activities in the auditory cortex of mice showed that neural populations represent sensory input and memory along two orthogonal dimensions. This transformation process, facilitated by ”stable” neurons (maintaining selectivity) and ”switching” neurons (reversing selectivity over time), effectively converts sensory input into memory. Model simulation also confirmed that the rotation dynamic and orthogonal representations can protect memory from sensory interference. These studies suggest that orthogonal representation play a significant role in reducing interference between tasks.

In the study by Flesch et al. Flesch et al. (2022), another discovery is the existence of low-dimensional, task-specific representations in human brains, particularly in the prefrontal areas. Neural encoding along task-irrelevant dimensions is compressed, yet this compression still retains the original space of the inputs. It suggests that the brain can handle multitasking and complex scenarios by encodes tasks through low-dimensional representations. In addition, the low-dimensional dynamics of neural population activities have been well documented in the areas of motor learning and motor execution. For example, Sadtler et al. (2014) investigated neural activity patterns in the primary motor cortex of rhesus monkeys during learning, revealed intrinsic constraints in the form of low-dimensional manifolds. Another study Perich et al. (2018) found that the brain uses low-dimensional neural population activities for rapid behavioral adaptation. In the premotor (PMd) and primary motor (M1) cortices , despite high-dimensional complexity in neuronal activities, the core functional connectivity remains stable. As a result, the activity of hundreds of motor neurons is represented in a low-dimensional manifold that reflects the covariance across the neuronal population. Pandarinath et al. (2018) discovered co-activation patterns among neurons. This means that even without observing all neurons, the brain's computational processes can be understood by analyzing a few key latent factors. This led to the proposal of the Latent Factor Analysis Dynamic Systems (LFADS) method. Applying this method to M1 and PMd also led to the discovery that the activities of a large number of neurons can be described by low-dimensional dynamics. In addition, Goudar et al.'s study Goudar et al. (2023) explored the acceleration of learning speed in neural networks when learning similar problems. They found that network activity patterns in the learning process are formed within the low-dimensional subspace of neural activity, and efficiency in learning similar tasks can be enhanced by restricting parameter exploration within this low-dimensional subspace.