OPT2025: 17th Annual Workshop on Optimization for Machine Learning

Feature Learning as a Virtual Covariance Learning

Taehun Cha CTH127@KOREA.AC.KR
Donghun Lee HOLY@KOREA.AC.KR
Department of Mathematics, Korea University, Republic of Korea

Abstract

Feature learning is central to the success of neural networks but remains poorly understood. Recent work proposed the Neural Feature Ansatz, which highlights alignment between learned features and $\nabla_x f$, but does not explicitly explain why and how feature learning dynamics occur. To address this, we introduce a novel concept, **virtual update**, a stochastic gradient descent (SGD) step applied to inputs and hidden states rather than parameters, i.e., $x - \gamma \nabla_x \mathcal{L}$ and $h - \gamma \nabla_h \mathcal{L}$. We theoretically show that SGD aligns network weights with the covariance structure of the virtual update. This does not result in disagreement with an actual update, as the actually updated input does not deviate far from the virtually updated input. Building on this insight, we propose the **virtual covariance learning** algorithm, which directly obtains the weight matrix that achieves the desired covariance structure. This algorithm efficiently learns effective weights within one or two epochs–whereas SGD requires 10–20 epochs–with low variance and no overfitting. Our results provide both a new theoretical perspective on feature learning and an alternative to standard training.

1. Introduction

Though feature learning is a key factor behind the success of neural networks, it has been less extensively studied from a theoretical perspective compared to optimization or generalization. For example, one of the most notable deep learning theories, the neural tangent kernel [6], simply linearizes the network around initialization in parameter space, effectively freezing features in the infinite-width limit. Although this provides a great simplification for understanding neural network learning dynamics, it does not directly address *why feature learning happens and what neural networks actually learn.*

Recently, Radhakrishnan et al. [11] proposed the Neural Feature Ansatz, which states a proportional relationship between the Gram of the weight matrix, $W^\top W$, and the outer product of input gradients, $\nabla_x f \cdot \nabla_x f^\top$. Unlike previous works, which mainly utilize the weight gradients, $\nabla_w f$, they highlighted the relationship between the *learned* features and $\nabla_x f$. However, their work did not directly address the detailed mechanisms or the dynamics involved in the process of feature *learning*, leaving open important questions about how and why neural networks develop meaningful internal representations during training.

We present *virtual update* and its corollary concepts that facilitate feature learning dynamics in neural networks. The virtual update, $x - \gamma \nabla_x \mathcal{L}$ and $h - \gamma \nabla_h \mathcal{L}$, assumes hypothetical updates in input and a hidden state with a stochastic gradient descent (SGD) step. It is *virtual* in the sense that we never actually update the input itself. Surprisingly, however, we theoretically verify that SGD induces a remarkable alignment of the weight matrix with the virtual update. We show that (1) an update in the Gram matrix of the weights is approximately equivalent to the update in covariance of

the virtual update, and (2) it does not conflict with an actual update, as both share the same sign and bounded magnitude. This provides insight into what weight matrices actually learn across multiple layers and may identify the core mechanism of feature learning.

Building on this insight, we propose the *virtual covariance learning* algorithm. By solving a constrained quadratic optimization problem, we directly obtain a weight matrix that aligns with the desired covariance structure. As a result, our algorithm efficiently reaches the best-performing weights within one or two epochs on real-world image datasets, whereas SGD requires 10–20 epochs. Our work not only suggests a powerful training algorithm for neural networks but may also offer a novel framework to understand the core of feature learning.

2. Virtual Covariance Learning

2.1. Mathematical Construction

Define an L-layer neural network $f(x) = w^\top h_L$, where $h_l \in \mathbb{R}^{d_l}$ denotes the l-th hidden state, recursively defined as $h_l = \sigma(W_{l-1}h_{l-1})$ with $h_0 = x \in \mathbb{R}^{d_0}$, and σ is a nonlinear activation function. The vector $w \in \mathbb{R}^{d_L}$ serves as the final predictor, and $W_l \in \mathbb{R}^{d_{l+1}\times d_l}$ are the weight matrices. For a general loss function $\mathcal{L}$, the gradient with respect to W_l is represented as $\nabla_{W_l}\mathcal{L} \in \mathbb{R}^{d_{l+1}} \times \mathbb{R}^{d_l}$ defined as the element-wise derivative matrix. By applying the chain rule, we obtain

$$\nabla_{W_l}\mathcal{L} = (\nabla_{h_{l+1}}\mathcal{L} \odot \sigma'(W_l h_l)) \cdot h_l^\top \quad \text{and}$$
$$\nabla_{h_l}\mathcal{L} = W_l^\top(\nabla_{h_{l+1}}\mathcal{L} \odot \sigma'(W_l h_l)),$$

where $\odot$ is an element-wise product, so we have,

$$W_l^\top \nabla_{W_l}\mathcal{L} = \nabla_{h_l}\mathcal{L} \cdot h_l^\top. \tag{1}$$

This equation suggests a relationship between the gradient with respect to the weight matrix and the gradient with respect to the hidden states. Based on this observation, we define the notions of virtual update and virtual covariance.

Definition 1 *We define the **virtual update** of a hidden state h_l (or of the input $h_0 = x$) with learning rate γ as $h_l^+ = h_l - \gamma\nabla_{h_l}\mathcal{L}$. Correspondingly, we define the **virtual covariance** as $\tilde{cov}(h_l^+) = h_l^+ \cdot (h_l^+)^\top = (h_l - \gamma\nabla_{h_l}\mathcal{L}) \cdot (h_l - \gamma\nabla_{h_l}\mathcal{L})^\top$. The **virtual covariance shift** is then given by $\tilde{cov}(h_l^+) - \tilde{cov}(h_l)$.*

The virtual update intuitively represents how the input to the l-th layer would change under gradient descent in order to minimize the loss. It is termed *virtual* because we never directly update the hidden states themselves, but rather the weight matrices. Since the virtual covariance corresponds to the covariance structure of virtually updated hidden states, the virtual covariance shift characterizes how the input covariance structure would evolve to reduce the loss.

We note, however, that $\tilde{\text{cov}}$ is not necessarily a true covariance matrix, as h_l^+ and h_l are not guaranteed to be centered. It coincides with a genuine covariance matrix when suitable normalization techniques are applied to the hidden states.

2.2. Rethinking the SGD as a Virtual Covariance Learning

Recall that stochastic gradient descent (SGD) optimizes the weight matrices by subtracting the gradients scaled by the learning rate, i.e., $W_l^+ = W_l - \gamma \nabla_W \mathcal{L}$. By computing the covariance matrix of the updated weight vector, we obtain the following theorem.

Theorem 2 *The shift in the covariance matrix of the SGD-updated weight matrix is approximately equal to the virtual covariance shift, up to a residual term of order γ^2, i.e.,*

$$(W_l^+)^\top W_l^+ - W_l^\top W_l = \tilde{cov}(h_l^+) - \tilde{cov}(h_l) + \gamma^2 \cdot \textit{residual}.$$

The proof is given in Appendix A.1. Theorem 2 states that after an SGD **actual update**, the weight covariance matrix is updated to reflect the covariance of the **virtual update**. This result can be naturally extended to the mini-batch training setting by defining $W_l^+ = W_l - \gamma \sum_i \nabla_W \mathcal{L}_i$ where $\mathcal{L}_i$ denotes the loss for the i-th data point. In this case, we obtain $(W_l^+)^\top W_l^+ - W_l^\top W_l \approx \sum_i \tilde{\text{cov}}(h_{l,i}^+) - \tilde{\text{cov}}(h_{l,i})$. By defining W_l^τ as the weight matrix obtained after τ steps of SGD updates, we have $(W_l^t)^\top W_l^t - (W_l^0)^\top W_l^0 \approx \sum_{\tau=0}^t \tilde{\text{cov}}(h_{l,\tau}^+) - \tilde{\text{cov}}(h_{l,\tau})$, where W_l^0 and h_l^0 denote the initialized weight matrix and its corresponding hidden state, respectively.

Compared to the Neural Feature Ansatz proposed by Radhakrishnan et al. [11], which states $W^\top W \propto \frac{1}{N}\sum_{i=1}^N \nabla_x f \cdot (\nabla_{x_i} f)^\top$, where $\frac{1}{N}\sum_{i=1}^N \nabla_x f \cdot (\nabla_{x_i} f)^\top$ is referred to as the Average Gradient Outer Product (AGOP), Theorem 2 also involves an AGOP-like term, but with gradients taken with respect to the loss function rather than the network output. Moreover, Theorem 2 treats the AGOP term as a negligible residual of order γ^2. We will empirically examine the effect of this AGOP term in Section 3.

As we have established the relationship between the covariance of the actually updated weight matrix and that of the virtual update, it is natural to ask about the corresponding relationship for expectations; specifically, between the virtual input h_l^+ and the actual input $\sigma(W_{l-1}^+ h_{l-1})$. The following theorem provides a formal characterization of this relationship.

Theorem 3 *Assume the non-linear activation function σ is L-Lipshitz and $\|h_{l-1}\| = 1$. By fixing the other layers, we obtain $|\sigma(W_{l-1}^+ h_{l-1}) - \sigma(W_{l-1} h_{l-1})| < L^2 |h_l^+ - h_l|$, elementwisely. Also, if we assume that σ is an increasing function, then sgn $\big(\sigma(W_{l-1}^+ h_{l-1}) - \sigma(W_{l-1} h_{l-1})\big) = sgn(h_l^+ - h_l)$.*

The proof is given in Appendix A.2. Theorem 3 states that the change in the actually updated hidden state, $\sigma(W_{l-1}^+ h_{l-1}) - \sigma(W_{l-1} h_{l-1})$, has the same sign as the change in the virtual update, $h_l^+ - h_l$. Moreover, its absolute magnitude is bounded by $|h_l^+ - h_l|$ factor. For activation functions such as ReLU or GELU, which satisfy the 1-Lipschitz condition, this bound becomes $|h_l^+ - h_l|$ itself.

Recall that Theorem 2 states that W_l^+ learns the covariance structure of the virtual update h_l^+. However, if the virtually updated hidden state h_l^+ deviates significantly from the actually updated hidden state $\sigma(W_{l-1}^+ h_{l-1})$, then a discrepancy may arise between what W_l^+ is implicitly trained to expect and what it actually receives as input. Fortunately, by Theorem 3, the actually updated hidden state does not deviate far from the virtual update. As a result, in practice, W_l^+ already carries information about $\sigma(W_{l-1}^+ h_{l-1})$ before it explicitly observes it.

2.3. Virtual Covariance Learning

We have shown that SGD captures the virtual covariance shift as an unintended artifact. If this phenomenon lies at the core of feature learning, it is natural to ask: *can we directly learn the virtual covariance structure?*

To address this question, we propose a novel algorithm called **virtual covariance learning**. Our approach solves the following optimization problem:

$$\min_{W^+} \|W^+ - W\|_F^2 \quad \text{subject to} \quad (W^+)^\top W^+ - W^\top W = \tilde{\text{cov}}(h^+) - \tilde{\text{cov}}(h), \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

Note that there exist many solutions satisfying $(W_l^+)^\top W_l^+ - W_l^\top W_l = \tilde{\text{cov}}(h_l^+) - \tilde{\text{cov}}(h_l)$ (e.g., if one W_l^+ satisfies the condition, then for any orthogonal matrix Q, the product QW_l^+ is also a valid solution). Since the condition only constrains the input covariance structure, we enforce similarity in the output by requiring the minimum-norm solution. We can exactly solve the problem as follows:

Proposition 4 *Let $\tilde{W}_l^+ \in \mathbb{R}^{d_{l+1} \times d_l}$, which satisfies $(\tilde{W}_l^+)^\top (\tilde{W}_l^+) = W_l^\top W_l + \tilde{\text{cov}}(h_l^+) - \tilde{\text{cov}}(h_l)$. Let $P\Sigma Q^\top$ be a singular value decomposition of $W_l(\tilde{W}_l^+)^\top$. Then the solution for Equation (2) is $PQ^\top \tilde{W}_l^+$.*

The proof is given in Appendix A.3. We summarize our algorithm in Appendix B.

3. Experiments

3.1. Comparison with AGOP

Radhakrishnan et al. [11] proposed the Neural Feature Ansatz, which takes the form $W^\top W \propto \frac{1}{N}\sum_{i=1}^N \nabla_x f \, (\nabla_{x_i} f)^\top$. They reported a surprising agreement between the two factors. Subsequently, several follow-up works have shown that empirical phenomena observed in neural networks also arise in AGOP-based feature learning algorithms—for example, spurious correlations [11], grokking [9], and neural collapse [2].

In this section, we compare weight covariance ($W^\top W$) with AGOP ($\frac{1}{N}\sum_{i=1}^N \nabla_x f \cdot (\nabla_{x_i} f)^\top$) and the sum of our proposed virtual covariance shift ($\sum_{\tau=1}^t \tilde{\text{cov}}(h_{l,\tau}^+) - \tilde{\text{cov}}(h_{l,\tau})$). We employ a two-layer neural network with a hidden dimension of 256 and train it on the CIFAR-10 dataset [7] using SGD with a learning rate of 0.05. We use the GELU function [5] as a σ. The model achieves 46% accuracy on the evaluation dataset. We then plot the diagonal components of $W^\top W$, AGOP, and the sum of the virtual covariance shift in Figure 1. We also compute the Pearson correlation ρ between them.

We observe that both virtual covariance and AGOP exhibit a high correlation with the weight covariance. However, virtual covariance achieves a correlation of $\rho \approx 0.98$, which is substantially higher than that of AGOP ($\rho \approx 0.71$). This tendency is also evident in the visualizations: while both methods highlight the centered region, AGOP produces a relatively vague pattern, whereas virtual covariance closely resembles the weight covariance. Through this, we empirically verify that Theorem 2 holds, and the γ^2 factor does not significantly affect the resulting weight covariance.

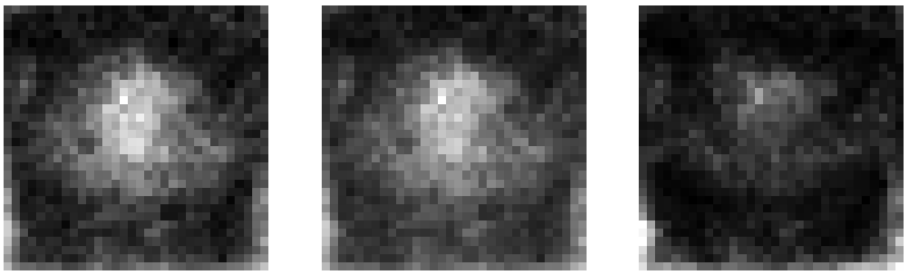

Figure 1: Visualization of the diagonal components of the weight covariance, AGOP, and the sum of our proposed virtual covariance shift. To quantify similarity, we compute the Pearson correlation coefficient ρ with respect to the weight covariance.

3.2. Empirical Performance

In this section, we empirically evaluate our virtual covariance update (**VC**) algorithm. As in the previous section, we employ a two-layer neural network with a hidden dimension of 256 and train it on the MNIST [3], SVHN [10], CIFAR-10, and CIFAR-100 datasets [7]. For our algorithm, we set $\alpha = 0.1$, where α denotes the matrix power factor, as $\alpha = 0.5$ leads to numerical instability. We perform hyper-parameter search for a learning rate $\gamma \in [5.0, 1.0, 0.5, 0.1, 0.05]$ and choose one which achieves the best test accuracy. Interestingly, a large learning rate (≥ 1.0), which is rarely used in neural network training, leads to good performance for some datasets. Before applying virtual covariance learning, we train the last layer as a warm-up stage.

For comparison, we train the same model using **SGD**, selecting the learning rate from $\gamma \in 0.1, 0.05, 0.01, 0.005$ and choosing the value that yields the highest test accuracy. Each experiment is repeated five times, and we report the mean and standard deviation of the results. The outcomes are shown in Figure 2.

For all datasets, our algorithm achieves the best performance of SGD in only one or two epochs. This is surprising, as SGD requires nearly 13 epochs on CIFAR-100 and 18 epochs on SVHN to reach comparable performance. More interestingly, on CIFAR-10, VC attains a level of performance that is not achievable by SGD. We verify that overfitting readily occurs with SGD on CIFAR-10, resulting in low mean accuracy and high variance. In contrast, our algorithm does not exhibit overfitting and consequently maintains consistently low variance.

In summary, our proposed virtual covariance learning achieves three properties:

- **Efficiency**: VC attains the best performance in only a few epochs.
- **Robustness**: VC exhibits very small performance variance.
- **Non-overfitting**: VC does not overfit with further training epochs, despite achieving the best performance early.

These results demonstrate not only the power of virtual covariance learning but may also provide insight into understanding complex feature learning dynamics in neural networks.

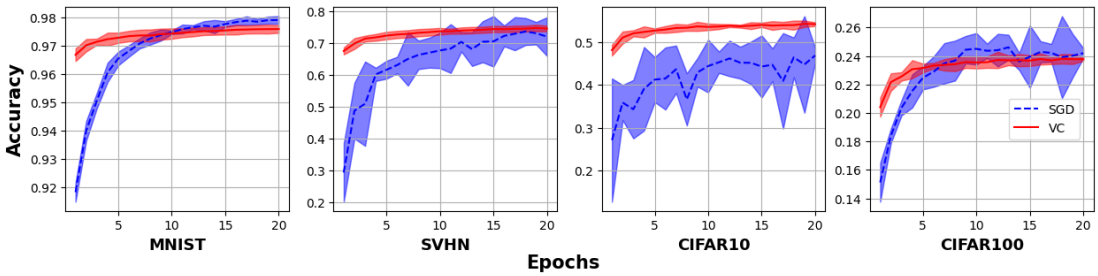

Figure 2: Test set accuracy for MNIST, SVHN, CIFAR-10, and CIFAR-100 datasets for each epoch. We run each experiment 5 times and report the mean and $2\times$ standard deviation. The blue dashed line represents routinely trained **SGD**, while the red solid line refers to our virtual covariance learning (**VC**).

4. Discussion

Note that the most time-consuming step for our algorithm is computing the square root of $(W_l^+)^\top W_l^+$, whose size is $d_l \times d_l$. This becomes particularly expensive when the input dimension is large; for example, in image data, $d_0 =$ (channels) $\times$ (height) $\times$ (width). However, since $W_l^\top W_l$ is already known and $\tilde{\mathrm{cov}}(h^+) - \tilde{\mathrm{cov}}(h)$ is low-rank, the problem can be solved recursively using the secular equation with improved efficiency. We leave a detailed investigation of this approach for future work.

We only investigated our algorithm in a two-layer feed-forward neural network. However, the difficulty of training *deep* neural networks is well-known both empirically [4] and theoretically [13]. Moreover, various neural network architectures have been introduced to incorporate inductive biases, such as convolutional neural networks [8] and self-attention structures [14]. We leave the extension of our virtual covariance learning to these deep and diverse neural network architectures as future work.

Another interesting phenomenon in neural networks is benign overfitting, i.e., the model generalizes well to unseen data while easily memorizing the training data. Bartlett et al. [1] showed that this phenomenon is deeply related to the eigenvalue structure of the input covariance, $\mathbb{E}[xx^\top]$. Since our virtual covariance learning also treats the input covariance as a direct training signal, we hope our framework sheds light on this previously unexplored phenomenon.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-24873052 and RS-2025-25433902).

References

[1] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

[2] Daniel Beaglehole, Peter Súkeník, Marco Mondelli, and Misha Belkin. Average gradient outer product as a mechanism for deep neural collapse. *Advances in Neural Information Processing Systems*, 37:130764–130796, 2024.

[3] 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. doi: 10.1109/MSP.2012.2211477.

[4] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[5] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[6] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[7] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[9] Neil Mallinar, Daniel Beaglehole, Libin Zhu, Adityanarayanan Radhakrishnan, Parthe Pandit, and Mikhail Belkin. Emergence in non-neural models: grokking modular arithmetic via average gradient outer product. *arXiv preprint arXiv:2407.20199*, 2024.

[10] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, 2011.

[11] Adityanarayanan Radhakrishnan, Daniel Beaglehole, Parthe Pandit, and Mikhail Belkin. Mechanism for feature learning in neural networks and backpropagation-free machine learning models. *Science*, 383(6690):1461–1467, 2024. doi: 10.1126/science.adi5639. URL https://www.science.org/doi/abs/10.1126/science.adi5639.

[12] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[13] Hao Shen. Towards a mathematical understanding of the difficulty in learning with feedforward neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 811–820, 2018.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Appendix A. Proof

A.1. Proof of Theorem 2

Theorem 2 *The shift in the covariance matrix of the SGD-updated weight matrix is approximately equal to the virtual covariance shift, up to a residual term of order γ^2, i.e.,*

$$(W_l^+)^\top W_l^+ - W_l^\top W_l = \tilde{cov}(h_l^+) - \tilde{cov}(h_l) + \gamma^2 \cdot \textit{residual}.$$

Proof

$$\begin{aligned}
(W_l^+)^\top W_l^+ &= (W_l - \gamma \nabla_W \mathcal{L})^\top (W_l - \gamma \nabla_W \mathcal{L}) \\
&= W_l^\top W_l - W_l^\top \gamma \nabla_W \mathcal{L} - \gamma \nabla_W \mathcal{L}^\top W_l + \gamma^2 \cdot \text{residual} \\
&\approx W_l^\top W_l - \gamma (W_l^\top \nabla_W \mathcal{L} + \nabla_W \mathcal{L}^\top W_l) \\
&= W_l^\top W_l - \gamma (\nabla_{h_l} \mathcal{L} \cdot h_l^\top + h_l \cdot \nabla_{h_l} \mathcal{L}^\top) \quad \text{by Equation (1).}
\end{aligned} \tag{3}$$

As $(h - \gamma \nabla_h \mathcal{L}) \cdot (h - \gamma \nabla_h \mathcal{L})^\top = h \cdot h^\top - \gamma (\nabla_{h_l} \mathcal{L} \cdot h_l^\top + h_l \cdot \nabla_{h_l} \mathcal{L}^\top) + \gamma^2 \cdot \text{residual}$, we can rewrite Equation (3) as,

$$\begin{aligned}
(W_l^+)^\top W_l^+ - W_l^\top W_l &\approx (h_l - \gamma \nabla_{h_l} \mathcal{L}) \cdot (h_l - \gamma \nabla_{h_l} \mathcal{L})^\top - h_l \cdot h_l^\top \\
&= \tilde{\text{cov}}(h_l^+) - \tilde{\text{cov}}(h_l)
\end{aligned}$$

■

A.2. Proof of Theorem 3

Theorem 3 *Assume the non-linear activation function σ is L-Lipshitz and $\|h_{l-1}\| = 1$. By fixing the other layers, we obtain $|\sigma(W_{l-1}^+ h_{l-1}) - \sigma(W_{l-1} h_{l-1})| < L^2 |h_l^+ - h_l|$, elementwisely. Also, if we assume that σ is an increasing function, then sgn $\big(\sigma(W_{l-1}^+ h_{l-1}) - \sigma(W_{l-1} h_{l-1})\big) = sgn(h_l^+ - h_l)$.*

Proof Define the pre-activation hidden state $z_l = W_{l-1} h_{l-1}$ and its actual update $z_l^+ = W_{l-1}^+ h_{l-1}$. Then we can observe

$$\begin{aligned}
z_l^+ &= W_{l-1}^+ h_{l-1} \\
&= W_{l-1} h_{l-1} - \gamma (\nabla_{W_{l-1}} \mathcal{L}) h_{l-1} \\
&= z_l - \gamma \nabla_{z_l} \mathcal{L} \langle h_{l-1}, h_{l-1} \rangle \quad \text{as } \nabla_{W_{l-1}} \mathcal{L} = \nabla_{z_l} \mathcal{L} \cdot h_{l-1}^\top \\
&= z_l - \gamma \nabla_{z_l} \mathcal{L} \quad \text{as } \|h_{l-1}\| = 1.
\end{aligned}$$

By the L-Lipschit property,

$$\begin{aligned}
|\sigma(W_{l-1}^+ h_{l-1}) - \sigma(W_{l-1} h_{l-1})| &< L |W_{l-1}^+ h_{l-1} - W_{l-1} h_{l-1}| \\
&= L |z_l^+ - z_l| \\
&= L |\gamma \nabla_{z_l} \mathcal{L}| \\
&= L |\gamma \sigma'(z_l) \odot \nabla_{h_l} \mathcal{L}| \quad \text{as } \nabla_{z_l} \mathcal{L} = \sigma'(z_l) \odot \nabla_{h_l} \mathcal{L} \\
&< L^2 |\gamma \nabla_{h_l} \mathcal{L}| \quad \text{as } \sigma'(z_l)_i \leq L, \forall i \\
&= L^2 |h_l^+ - h_l|
\end{aligned}$$

For the sign case, as σ is a increasing function,

$$\begin{aligned}
\operatorname{sgn}(\sigma(W_{l-1}^{+}h_{l-1}) - \sigma(W_{l-1}h_{l-1})) &= \operatorname{sgn}(\sigma(z_l^{+}) - \sigma(z_l)) \\
&= \operatorname{sgn}(z_l^{+} - z_l) \quad \text{by increasing property} \\
&= \operatorname{sgn}(-\nabla_{z_l}\mathcal{L}) \\
&= \operatorname{sgn}(-\sigma'(z_l) \odot \nabla_{h_l}\mathcal{L}) \\
&= \operatorname{sgn}(-\nabla_{h_l}\mathcal{L}) \quad \text{by increasing property} \\
&= \operatorname{sgn}(h_l^{+} - h_l)
\end{aligned}$$

■

A.3. Proof of Proposition 4

Proposition 4 *Let $\tilde{W}_l^{+} \in \mathbb{R}^{d_{l+1} \times d_l}$, which satisfies $(\tilde{W}_l^{+})^{\top}(\tilde{W}_l^{+}) = W_l^{\top}W_l + \tilde{\operatorname{cov}}(h_l^{+}) - \tilde{\operatorname{cov}}(h_l)$. Let $P\Sigma Q^{\top}$ be a singular value decomposition of $W_l(\tilde{W}_l^{+})^{\top}$. Then the solution for Equation (2) is $PQ^{\top}\tilde{W}^{+}$.*

Proof Recall, our problem is

$$\min_{W^{+}} \|W^{+} - W\|_F^2 \quad \text{subject to} \quad (W^{+})^{\top}W^{+} - W^{\top}W = \tilde{\operatorname{cov}}(h^{+}) - \tilde{\operatorname{cov}}(h).$$

For any matrix $U \in \mathbb{R}^{d_{l+1}} \times \mathbb{R}^{d_{l+1}}$, satisfying $U^{\top}U = I$, UW^{+} satisfies the equation condition. So our goal becomes finding

$$\min_{U^{\top}U=I} \|U\tilde{W}^{+} - W\|_F^2.$$

for any $\tilde{W}^{+}$ satisfying the condition, and it is an orthogonal Procrustes problem [12]. By applying the singular value decomposition to $W(\tilde{W}^{+})^{\top}$, we obtain $P\Sigma Q^{\top}$, and our desired solution is $PQ^{\top}\tilde{W}^{+}$

■

Appendix B. Virtual Covariance Learning Algorithm

Here we present our virtual covariance learning algorithm. We introduce a hyperparameter α to control the matrix power, as we found that a small value (e.g., $\alpha \approx 0.1$) provides better numerical stability than $\alpha = 0.5$.

Algorithm 1: Virtual Covariance Learning

Input : Neural network f with weight matrices W_l, loss function $\mathcal{L}$, data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, hyperparameters γ and α

Output: Trained neural network $\hat{f}$

Randomly initialize f;
$f \leftarrow$ Train Last Layer (warm Up) ;
Set initial virtual covariance shift of each layer $\mathcal{V}_l = 0$;
for *Mini batch* $B \subset \mathcal{D}$ **do**
  Compute h_l and $\nabla_{h_l}\mathcal{L}$ for each layer l with forward pass;
  for $l \in \{1, \ldots, L\}$ **do**
    Compute virtual covariance shift $v_l = \tilde{\text{cov}}(h_l^+) - \tilde{\text{cov}}(h_l)$;
    $\mathcal{V}_l \leftarrow \mathcal{V}_l + v_l$;
    $Q\Lambda Q^\top \leftarrow$ EVD of $W_l^\top W_l + \mathcal{V}_l$;
    $\tilde{W_l}^+ \leftarrow \Lambda[: d_{l+1}, : d_{l+1}]^\alpha Q[:, : d_{l+1}]^\top$;
    $P\Sigma Q^\top \leftarrow$ SVD of $W_l(\tilde{W_l}^+)^\top$;
    $W_l \leftarrow PQ^\top \tilde{W_l}^+$;
  end
end