Understanding Self-supervised Learning with Dual Deep Networks

Abstract

We propose a novel theoretical framework to understand self-supervised learning methods that employ dual pairs of deep ReLU networks (e.g., SimCLR, BYOL). First, we prove that in each SGD update of SimCLR, the weights at each layer are updated by a *covariance operator* that specifically amplifies initial random selectivities that vary across data samples but survive averages over data augmentations, which we show leads to the emergence of hierarchical features, if the input data are generated from a hierarchical latent tree model. With the same framework, we also show analytically that BYOL works due to an implicit contrastive term, acting as an approximate covariance operator. The term is formed by the interplay between the zero-mean operation of BatchNorm and the extra predictor in the online network. Extensive ablation studies justify our theoretical findings.

1 Introduction

While self-supervised learning (SSL) has achieved great empirical success across multiple domains, including computer vision [17, 15, 8, 16, 25, 7], natural language processing [12], and speech recognition [34, 4, 5], its theoretical understanding remains elusive, especially when multi-layer nonlinear deep networks are involved. Unlike supervised learning (SL) that deals with labeled data, SSL learns meaningful structures from randomly initialized networks without human-provided labels.

In this paper, we propose a systematic theoretical analysis of SSL with deep ReLU networks. Our analysis imposes no parametric assumptions on the input data distribution and is applicable to state-of-the-art SSL methods that typically involve two parallel (or *dual*) deep ReLU networks during training (e.g., SimCLR [8], BYOL [16], etc). We do so by developing an analogy between SSL and a theoretical framework for analyzing supervised learning, namely the student-teacher setting [31, 1, 21, 28], which also employs a pair of dual networks. Our results indicate that SimCLR weight updates at every layer are amplified by a fundamental positive semi definite (PSD) *covariance operator* that only captures feature variability across data points that *survive* averages over data augmentation procedures designed in practice to scramble semantically unimportant features (e.g. random image crops, blurring or color distortions [13, 20, 25, 27]). This covariance operator provides a principled framework to study how SimCLR amplifies initial random selectivity to obtain distinctive features that vary *across* samples after surviving averages over data-augmentations.

Based on the covariance operator, we further show that (1) in a two-layer setting, a top-level covariance operator helps accelerate the learning of low-level features, and (2) when the data are generated by a hierarchical latent tree model, training deep ReLU networks leads to an emergence of the latent variables in its intermediate layers. We also explain why BYOL works *without negative pairs*, by showing analytically that an interplay between the zero-mean operation in BatchNorm and the extra predictor in the online network creates an implicit contrastive term, thereby explaining empirical observations in the recent blog [14]. We also discover that reinitializing the predictor every a few epochs doesn't hurt the performance measured by linear evaluation protocol, questioning the hypothesis of "optimal predictor" in a recent version (v3) of BYOL [16].

To the best of our knowledge, we are the first to provide a systematic theoretical analysis of modern SSL methods with deep ReLU networks that also incorporates data augmentation, and show how the gradient-based SSL work *inside* the neural network. See Appendix I for more related works.

2 Overall framework

Notation. Consider an L-layer ReLU network obeying $\boldsymbol{f}_l = \psi(\tilde{\boldsymbol{f}}_l)$ and $\tilde{\boldsymbol{f}}_l = W_l \boldsymbol{f}_{l-1}$ for $l = 1, \dots L$. Here $\tilde{\boldsymbol{f}}_l$ and $\boldsymbol{f}_l$ are n_l dimensional pre-activation and activation vectors in layer l, with

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

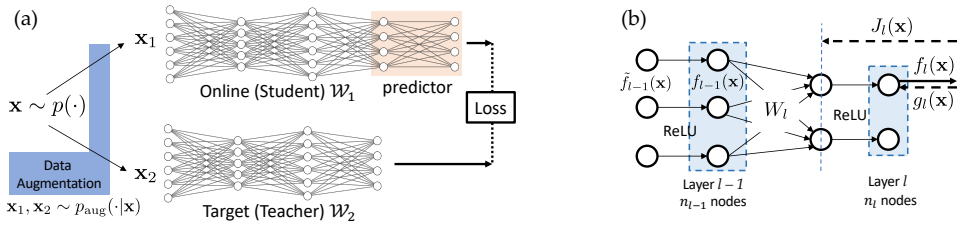


Figure 1: **(a)** Overview of the two SSL algorithms we study in this paper: SimCLR ($\mathcal{W}_1 = \mathcal{W}_2 = \mathcal{W}$, no predictor, NCE Loss) and BYOL ($\mathcal{W}_1$ has an extra predictor, $\mathcal{W}_2$ is a moving average), **(b)** Detailed Notation.

$\boldsymbol{f}_0 = \boldsymbol{x}$ being the input and $\boldsymbol{f}_L = \tilde{\boldsymbol{f}}_L$ the output (no ReLU at the top layer). $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ are the weight matrices, and $\psi(\boldsymbol{u}) := \max(\boldsymbol{u}, 0)$ is the element-wise ReLU nonlinearity. We let $\mathcal{W} := \{W_l\}_{l=1}^L$ be all network weights. We also denote the gradient of any loss function with respect to $\boldsymbol{f}_l$ by $\boldsymbol{g}_l \in \mathbb{R}^{n_l}$, and the derivative of the output $\boldsymbol{f}_L$ with respect to an earlier pre-activation $\tilde{\boldsymbol{f}}_l$ by the Jacobian matrix $J_l(\boldsymbol{x}; \mathcal{W}) \in \mathbb{R}^{n_L \times n_l}$, as both play key roles in backpropagation (Fig. 1(b)).

An analogy between self-supervised and supervised learning: the dual network scenario. Many recent successful approaches to self-supervised learning (SSL), including SimCLR [8], BYOL [16] and MoCo [17], employ a dual "Siamese-like" pair [19] of such networks (Fig. 1(b)). Each network has its own set of weights $\mathcal{W}_1$ and $\mathcal{W}_2$, receives respective inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and generates outputs $\boldsymbol{f}_{1,L}(\boldsymbol{x}_1; \mathcal{W}_1)$ and $\boldsymbol{f}_{2,L}(\boldsymbol{x}_2; \mathcal{W}_2)$. The pair of inputs $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ can be either positive or negative, depending on how they are sampled. For a positive pair, a *single* data point $\boldsymbol{x}$ is drawn from the data distribution $p(\cdot)$, and then two augmented views $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are drawn from a conditional augmentation distribution $p_{\text{aug}}(\cdot|\boldsymbol{x})$. Possible image augmentations include random crops, blurs or color distortions, that ideally preserve semantic content useful for downstream tasks. In contrast, for a negative pair, two *different* data points $\boldsymbol{x}, \boldsymbol{x}' \sim p(\cdot)$ are sampled, and then each are augmented independently to generate $\boldsymbol{x}_1 \sim p_{\text{aug}}(\cdot|\boldsymbol{x})$ and $\boldsymbol{x}_2 \sim p_{\text{aug}}(\cdot|\boldsymbol{x}')$. For SimCLR, the dual networks have tied weights with $\mathcal{W}_1 = \mathcal{W}_2$, and a loss function is chosen to encourage the representation of positive (negative) pairs to become similar (dissimilar). In BYOL, only positive pairs are used, and the first network $\mathcal{W}_1$, called the online network, is trained to match the output of the second network $\mathcal{W}_2$ (the target), using an additional layer named *predictor*. The target network ideally provides training targets that can improve the online network's representation and does not contribute gradient. The improved online network is gradually incorporated into the target network, yielding a bootstrapping procedure.

Our fundamental goal is to analyze the mechanisms governing how SSL methods like SimCLR and BYOL lead to the emergence of meaningful intermediate features, starting from random initializations, and how these features depend on the data distribution $p(\boldsymbol{x})$ and augmentation procedure $p_{\text{aug}}(\cdot|\boldsymbol{x})$. Interestingly, the analysis of *supervised* learning (SL) often employs a similar dual network scenario, called *teacher-student setting* [31, 1, 21, 28], where $\mathcal{W}_2$ are the ground truth weights of a *fixed* teacher network, which generates outputs in response to random inputs. These input-output pairs constitute training data for the first network, which is a student network. Only the student network's weights $\mathcal{W}_1$ are trained to match the target outputs provided by the teacher. This yields an interesting mathematical parallel between SL, in which the teacher is fixed and only the student evolves, and SSL, in which both the teacher and student evolve with potentially different dynamics. This mathematical parallel opens the door to using techniques from SL (e.g., [31]) to analyze SSL.

Gradient of ℓ_2 loss for dual deep ReLU networks. As seen above, the (dis)similarity of representations between a pair of dual networks plays a key role in both SSL and SL. We thus consider minimizing a simple measure of dissimilarity, the squared ℓ_2 distance $r := \frac{1}{2}\|\boldsymbol{f}_{1,L} - \boldsymbol{f}_{2,L}\|^2$ between the final outputs $\boldsymbol{f}_{1,L}$ and $\boldsymbol{f}_{2,L}$ of two multi-layer ReLU networks with weights $\mathcal{W}_1$ and $\mathcal{W}_2$ and inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Without loss of generality, we only analyze the gradient w.r.t $\mathcal{W}_1$. For each layer l, we first define *connection* $K_l(\boldsymbol{x})$, a quantity that connects the bottom-up feature vector $\boldsymbol{f}_{l-1}$ with the top-down Jacobian J_l, which both contribute to the gradient at weight layer l.

Definition 1 (The connection $K_l(\boldsymbol{x})$)**.** *The* connection $K_l(\boldsymbol{x}; \mathcal{W}) := \boldsymbol{f}_{l-1}(\boldsymbol{x}; \mathcal{W}) \otimes J_l^\intercal(\boldsymbol{x}; \mathcal{W}) \in \mathbb{R}^{n_l n_{l-1} \times n_L}$*. Here* $\otimes$ *is the Kronecker product.*

Theorem 1 (Squared ℓ_2 Gradient for dual deep ReLU networks)**.** *The gradient* g_{W_l} *of* r *w.r.t.* $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ *for a single input pair* $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ *is (here* $K_{1,l} := K_l(\boldsymbol{x}_1; \mathcal{W}_1)$ *and* $K_{2,l} := K_l(\boldsymbol{x}_2; \mathcal{W}_2)$*):*

$$g_{W_l} = \text{vec}\left(\partial r / \partial W_{1,l}\right) = K_{1,l}\left[K_{1,l}^\intercal \text{vec}(W_{1,l}) - K_{2,l}^\intercal \text{vec}(W_{2,l})\right]. \tag{1}$$

See Appendix for proofs of all theorems in main text.

3 Analysis of SimCLR

As discussed above, SimCLR [8] employs both positive and negative input pairs, and a symmetric network structure with $\mathcal{W}_1 = \mathcal{W}_2 = \mathcal{W}$. Let $\{\boldsymbol{x}_1, \boldsymbol{x}_+\}$ be a positive input pair, and let $\{\boldsymbol{x}_1, \boldsymbol{x}_{k-}\}$ for $k = 1, \ldots, H$ be H negative pairs. These input pairs induce corresponding squared ℓ_2 distances in output space, $r_+ := \frac{1}{2}\|\boldsymbol{f}_{1,L} - \boldsymbol{f}_{+,L}\|_2^2$, and $r_{k-} := \frac{1}{2}\|\boldsymbol{f}_{1,L} - \boldsymbol{f}_{k-,L}\|_2^2$. The InfoNCE loss [26] with temperature τ then minimizes (maximizes) the positive (negative) pair distances:

$$L(r_+, r_{1-}, r_{2-}, \ldots, r_{H-}) \;:=\; -\log \frac{e^{-r_+/\tau}}{e^{-r_+/\tau} + \sum_{k=1}^{H} e^{-r_{k-}/\tau}} \tag{2}$$

When $\|\boldsymbol{u}\|_2 = \|\boldsymbol{v}\|_2 = 1$, we have $-\frac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 = \mathrm{sim}(\boldsymbol{u}, \boldsymbol{v}) - 1$ where $\mathrm{sim}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u}^\intercal \boldsymbol{v}}{\|\boldsymbol{u}\|_2 \|\boldsymbol{v}\|_2}$, and Eqn. 2 reduces to what the original SimCLR uses (the term $e^{-1/\tau}$ cancels out). We leave the topmost ℓ_2-normalization to future work. One property of InfoNCE is important for our analysis:

Theorem 2 (Property of Contrastive Loss)**.** *For InfoNCE loss (Eqn. 2), we have* $\frac{\partial L}{\partial r_+} > 0$ *and* $\frac{\partial L}{\partial r_{k-}} < 0$ *for* $k = 1, \ldots, H$ *and* $\frac{\partial L}{\partial r_+} + \sum_{k=1}^{H} \frac{\partial L}{\partial r_{k-}} = 0$.

Now, under an approximation in which we neglect variations in $\frac{\partial L}{\partial r_{k-}}$ across different k in a large minibatch, Theorem 1 and 2 imply the SimCLR gradient is governed by positive semi-definite (PSD) covariance operator at any layer (some simple contrastive loss (e.g., $r_+ - r_-$) satisfies it trivially):

Theorem 3 (Covariance Operator from Contrastive Loss)**.** *Assume that for any* $\{\boldsymbol{x}_1, \boldsymbol{x}_+, \boldsymbol{x}_{k-}\}$, $\frac{\partial L}{\partial r_{k-}} = -\beta/H$ *is a constant, then in a large batch limit, the update for* W_l *is*

$$W_l(t+1) = W_l(t) + \alpha \Delta W_l(t), \qquad \textit{where } \mathrm{vec}(\Delta W_l(t)) = \beta \mathbb{V}_{\boldsymbol{x}}[\bar{K}_l(\boldsymbol{x})]\mathrm{vec}(W_l(t)). \tag{3}$$

where α *is the learning rate and* $\bar{K}_l(\boldsymbol{x}; \mathcal{W}) := \mathbb{E}_{\boldsymbol{x}' \sim p_{\mathrm{aug}}(\cdot|\boldsymbol{x})}\left[K_l(\boldsymbol{x}'; \mathcal{W})\right]$ *is the expected connection under the augmentation distribution, conditional on the datapoint* $\boldsymbol{x}$.

Above, we use the notation $\mathrm{Cov}[X, Y] := \mathbb{E}\left[XY^\intercal\right] - \mathbb{E}\left[X\right]\mathbb{E}\left[Y\right]^\intercal$ and $\mathbb{V}[X] := \mathrm{Cov}[X, X]$. The *covariance operator* $\mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right] \in \mathbb{R}^{n_l n_{l-1} \times n_l n_{l-1}}$ is a time-varying PSD matrix over the *entire* training procedure. Therefore, all its eigenvalues are non-negative and at any time t, W_l is most amplified along its largest eigenmodes. Intuitively, this covariance operator ignores different views of the same sample $\boldsymbol{x}$ by averaging over the augmentation distribution to compute $\bar{K}_l(\boldsymbol{x})$, and then computes the expected covariance of this *augmentation averaged* connection with respect to the data distribution $p(\boldsymbol{x})$. Thus, at all layers, any variability in the connection across different data points, that survives augmentation averages, leads to weight amplification. This amplification of weights by the PSD data covariance of an augmentation averaged connection constitutes a fundamental description of SimCLR learning dynamics for *arbitrary* data and augmentation distributions.

To understand the intuitions of covariance operator, please check Appendix A for details.

4 Analysis of ingredients underlying the success of BYOL

In BYOL, the two networks are no-longer identical and, interestingly, only positive pairs are used for training. The first network with weights $\mathcal{W}_1 = \mathcal{W} := \{\mathcal{W}_{\mathrm{base}}, \mathcal{W}_{\mathrm{pred}}\}$ is an *online* network that is trained to predict the output of the second *target* network with weights $\mathcal{W}_2 = \mathcal{W}'$, using a learnable *predictor* $\mathcal{W}_{\mathrm{pred}}$ to map the online to target outputs (Fig. 1(a) and Fig. 1 in [16]). In contrast, the target network has $\mathcal{W}' := \{\mathcal{W}'_{\mathrm{base}}, \mathcal{W}'_{\mathrm{pred}}\}$, where $\mathcal{W}'_{\mathrm{pred}}$ is an identity and $\mathcal{W}'_{\mathrm{base}}$ is exponential moving average (EMA) of $\mathcal{W}_{\mathrm{base}}$: $\mathcal{W}'_{\mathrm{base}}(t+1) = \gamma_{\mathrm{ema}}\mathcal{W}'_{\mathrm{base}}(t) + (1 - \gamma_{\mathrm{ema}})\mathcal{W}_{\mathrm{base}}(t)$.

We now theoretically analyze BYOL to explain why the predictor and an additional ingredient BatchNorm (BN) [18] are simultaneously required for BYOL's success. Without EMA ($\gamma_{\mathrm{ema}} = 0$) *and* the predictor, we obtain $\mathcal{W}' = \mathcal{W}$ and the BYOL update without BN for $\mathcal{W}$ is (derivation in Appendix F.1):

$$\mathrm{vec}\left(\Delta W_l\right)_{\mathrm{sym}} = -\mathbb{E}_{\boldsymbol{x}}\left[\mathbb{V}_{\boldsymbol{x}' \sim p_{\mathrm{aug}}(\cdot|\boldsymbol{x})}\left[K_l(\boldsymbol{x}')\right]\right]\mathrm{vec}(W_l). \tag{4}$$

This update only promotes variance minimization in the representations of different augmented views of the same data samples and therefore would yield model collapse. Motivated by a recent blogpost [14], we now consider BN in addition to the predictor. In particular, we analyze a simplified version of BN in which the mini-batch mean only is subtracted. When adding predictor, Theorem 1 can still be applied by adding identity layers on top of the target network $\mathcal{W}'$ so that the

online and the target networks have the same depth. Theorem 5 in [30] demonstrates this version of BN shifts the downward gradients so their mini-batch mean is $\mathbf{0}$:

$$\tilde{\boldsymbol{g}}_l^i := \boldsymbol{g}_l^i - \frac{1}{|B|}\sum_{i\in B}\boldsymbol{g}_l^i = \boldsymbol{g}_l^i - \bar{\boldsymbol{g}}_l \tag{5}$$

Here $\boldsymbol{g}_l^i$ is the i-th sample in a batch and $\bar{\boldsymbol{g}}_l$ is the batch average (same for $\bar{\boldsymbol{f}}_l$). Backpropagating through this BN (vs. just subtracting the mean only in the forward pass[1]), leads to a correction term:

Theorem 4. *If **(1)** the network is linear from layer l to the topmost and **(2)** the downward gradient $\boldsymbol{g}_l$ undergoes Eqn. 5, then with large batch limits, the correction of the update is[2] (for brevity, dependency on $\mathcal{W}$ is omitted, while dependency on $\mathcal{W}'$ is made explicit):*

$$\mathrm{vec}(\delta W_l^{\mathrm{BN}}) = \mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\left\{\mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l^\intercal(\boldsymbol{x})\right]\mathrm{vec}(W_l) - \mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l^\intercal(\boldsymbol{x};\mathcal{W}')\right]\mathrm{vec}(W_l')\right\} \tag{6}$$

and the corrected weight update is $\widetilde{\Delta W_l} := \Delta W_l + \delta W_l^{\mathrm{BN}}$. Using Eqn. 4, we have:

$$\mathrm{vec}(\widetilde{\Delta W_l}) = \mathrm{vec}(\Delta W_l)_{\mathrm{sym}} - \mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\mathrm{vec}(W_l) + \mathrm{Cov}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x}), \bar{K}_l(\boldsymbol{x};\mathcal{W}')\right]\mathrm{vec}(W_l') \tag{7}$$

Theorem 4 makes several predictions.

SimCLR. In this case, both networks use the same weight and there is no predictor. This means $\mathcal{W}' = \mathcal{W}$. From the analysis above, we have $\delta W_l^{\mathrm{BN}} = 0$ and BN should not matter. This is justified in the recent blogpost [14].

BYOL. When the predictor is present, $\mathcal{W}' \neq \mathcal{W}$ and BN is present, from the analysis above we know that $\delta W_l^{\mathrm{BN}} \neq 0$, which provides an implicit contrastive term. Note that $\mathcal{W}' \neq \mathcal{W}$ means there is a predictor, the online network uses EMA, or both.

The Predictor. We first discuss the predictor without EMA. To see why $\mathcal{W}_{\mathrm{pred}}$ plays a critical role, consider the last two terms (denoted as $\widehat{\Delta W_l}$) in Eqn. 7:

$$\widehat{\Delta W_l} = -\mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x};\mathcal{W})\right]\mathrm{vec}(W_l) + \mathrm{Cov}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x};\mathcal{W}), \bar{K}_l(\boldsymbol{x};\mathcal{W}')\right]\mathrm{vec}(W_l') \tag{8}$$

If there is no EMA (i.e., $\mathcal{W}'_{\mathrm{base}} = \mathcal{W}_{\mathrm{base}}$) and the weights of the predictor are all small positive numbers (e.g., $\approx \beta > 0$), then $+\mathrm{Cov}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x}), \bar{K}_l(\boldsymbol{x};\mathcal{W}')\right] \approx \beta\mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x};\mathcal{W}_{\mathrm{base}})\right]$ and for all layer l in $\mathcal{W}_{\mathrm{base}}$, Eqn. 8 becomes:

$$\widehat{\Delta W_l} \approx \beta(1-\beta)\mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x};\mathcal{W}_{\mathrm{base}})\right]\mathrm{vec}(W_l) \tag{9}$$

Intuitively, in Eqn. 8, the first term $-\mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x};\mathcal{W})\right]$ is second order in the Jacobian of $\mathcal{W}_{\mathrm{pred}}$, while the second term $+\mathrm{Cov}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x};\mathcal{W}), \bar{K}_l(\boldsymbol{x};\mathcal{W}')\right]$ is first-order with respect to $\mathcal{W}_{\mathrm{pred}}$. So if the predictor weight $\mathcal{W}_{\mathrm{pred}}$ is "small" in magnitude (e.g., $\beta \ll 1$), then the latter term dominates and $\widehat{\Delta W_l}$ becomes the covariance operator. In this regime, BYOL with predictor+BN sensibly amplifies variance across data samples through the covariance operator, and minimizes variance across different augmented views of the same data sample through $-\mathbb{E}_{\boldsymbol{x}}\left[\mathbb{V}_{\boldsymbol{x}'\sim p_{\mathrm{aug}}(\cdot|\boldsymbol{x})}\left[K_l(\boldsymbol{x}';\mathcal{W})\right]\right]$, which is the first term in Eqn. 7. Interestingly, SimCLR doesn't have such a term.

The Exponential Moving Average (EMA). On the other hand, the EMA part might play a different role. Consider the following linear dynamic system, which is a simplified version of Eqn. 7:

$$\boldsymbol{w}(t+1) - \boldsymbol{w}(t) = \Delta\boldsymbol{w}(t) = \alpha\left[-\boldsymbol{w}(t) + (1-\lambda)\boldsymbol{w}_{\mathrm{ema}}(t)\right] \tag{10}$$

Using z-transform, we could compute its two roots (See Appendix F.3 for the derivation):

$$z_{\min,\max} = 1 - \frac{1}{2}\left[(1-\gamma_{\mathrm{ema}}+\alpha) \pm \sqrt{(1-\gamma_{\mathrm{ema}}+\alpha)^2 - 4\alpha(1-\gamma_{\mathrm{ema}})\lambda}\right] \tag{11}$$

As analyzed above, the magnification factor of $\boldsymbol{w}_{\mathrm{ema}}(t)$ is larger than that of $\boldsymbol{w}(t)$, and thus $\lambda < 0$, and $z_{\max} > 1 > z_{\min}$. As a result, $\boldsymbol{w}(t) \propto z_{\max}^t$ will have exponential growth and learning happens. Compared to no EMA case (i.e., $\gamma_{\mathrm{ema}} = 0$), with a $\gamma_{\mathrm{ema}} < 1$ but close to 1, $z_{\max}$ becomes smaller (but still > 1) and the exponential growth is less aggressive, which stabilizes the training.

Indeed, empirical findings in a recent blogpost [14] as well as our own experiments (Tbl. 4) suggests that standard BYOL without BN fails. In addition, we also initialize the predictor with small positive weights (See Appendix H) , as well as reinitialize the predictor weight once in a while (Tbl. 6), and BYOL still works well. These empirical evidences are all consistent with the theoretical prediction.

[1] In PyTorch, the former is `x-x.mean()` and the latter is `x-x.mean().detach()`.

[2] A formal treatment requires Jacobian J to incorporate BatchNorm's contribution and is left for future work.

References

[1] Zeyuan Allen-Zhu and Yuanzhi Li. Backward feature correction: How deep learning performs deep learning. *arXiv preprint arXiv:2001.04413*, 2020.

[2] Humam Alwassel, Dhruv Mahajan, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. *arXiv preprint arXiv:1911.12667*, 2019.

[3] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. February 2019.

[4] Alexei Baevski and Abdelrahman Mohamed. Effectiveness of self-supervised pre-training for asr. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7694–7698. IEEE, 2020.

[5] Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. *arXiv preprint arXiv:1910.05453*, 2019.

[6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[9] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[10] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] William Falcon and Kyunghyun Cho. A framework for contrastive self-supervised learning and designing a new approach, 2020.

[14] Abe Fetterman and Josh Albrecht. Understanding self-supervised and contrastive learning with "bootstrap your own latent" (byol), 2020.

[15] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6391–6400, 2019.

[16] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

[19] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[20] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1920–1929, 2019.

[21] Andrew K Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[22] Jason D Lee, Qi Lei, Nikunj Saunshi, and Jiacheng Zhuo. Predicting what you already know helps: Provable self-supervised learning. *arXiv preprint arXiv:2008.01064*, 2020.

[23] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020.

[24] JG Liao and Arthur Berg. Sharpening jensen's inequality. *The American Statistician*, 2018.

[25] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[26] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[27] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv preprint arXiv:2007.13916*, 2020.

[28] David Saad and Sara A Solla. Dynamics of on-line gradient descent learning for multilayer neural networks. In *Advances in neural information processing systems*, pages 302–308, 1996.

[29] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[30] Yuandong Tian. A theoretical framework for deep locally connected relu network. *arXiv preprint arXiv:1809.10829*, 2018.

[31] Yuandong Tian. Student specialization in deep relu networks with finite width and input dimension. *ICML*, 2020.

[32] Christopher Tosh, Akshay Krishnamurthy, and Daniel Hsu. Contrastive learning, multi-view redundancy, and linear models. *arXiv preprint arXiv:2008.10150*, 2020.

[33] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *arXiv preprint arXiv:2005.10242*, 2020.

[34] Anne Wu, Changhan Wang, Juan Pino, and Jiatao Gu. Self-supervised representations improve end-to-end speech translation. *arXiv preprint arXiv:2006.12124*, 2020.

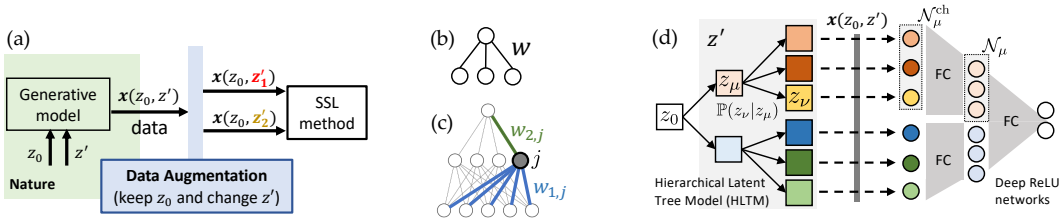


Figure 2: Overview of Sec. A. **(a)** To analyze the functionality of the *covariance operator* $\mathbb{V}_{z_0}\left[\bar{K}_l(z_0)\right]$ (Eqn. 3), we assume that Nature generates the data from a certain generative model with latent variable z_0 and z', while data augmentation takes $\boldsymbol{x}(z_0, z')$, changes z' but keeps z_0 intact. **(b)** *Sec. A.1*: one layer one neuron example. **(c)** *Sec. A.2*: two-layer case where $\mathbb{V}[\bar{K}_1]$ and $\mathbb{V}[\bar{K}_2]$ interplay. **(d)** *Sec. A.3*: Hierarchical Latent Tree Models and deep ReLU networks trained with SimCLR. A latent variable z_μ, and its corresponding nodes $\mathcal{N}_\mu$ in multi-layer ReLU side, covers a subset of input $\boldsymbol{x}$, resembling local receptive fields in ConvNet.

A How the covariance operator drives the emergence of features

To concretely illustrate how the fundamental covariance operator derived in Theorem 3 drives feature emergence in SimCLR, we setup the following paradigm for analysis. We assume the input $\boldsymbol{x} = \boldsymbol{x}(z_0, z')$ is generated by two groups of latent variables, *class/sample-specific* latent z_0 and *nuisance* latent z'. After data augmentation, z_0 remains the same while z' changes (Fig. 2(a)). In this case, the covariance operator is simply $\mathbb{V}_{z_0}[\bar{K}_l(z_0)]$ since all nuisance latents z' are integrated out in $\bar{K}_l(z_0)$.

In this setting, we first show that a linear neuron performs PCA within an augmentation preserved subspace. We then consider how nonlinear neurons with local receptive fields (RFs) can learn to detect simple objects. Finally, we extend our analysis to deep ReLU networks exposed to data generated by a hierarchical latent tree model (HLTM), proving that, with sufficient over-parameterization, there exist lucky nodes at initialization whose activation is correlated with latent variables underlying the data, and that SimCLR amplifies these initial lucky patterns during learning.

A.1 Self-supervised learning and the single neuron: illustrative examples

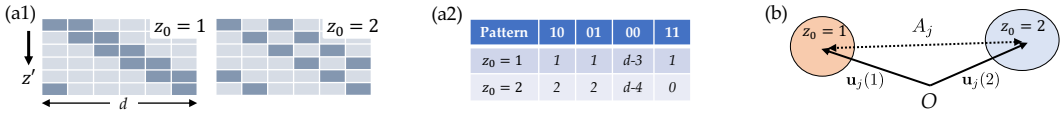
A single linear neuron performs PCA in a preserved subspace. For a single linear neuron ($L = 1, n_L = 1$), the connection in definition 1 is simply $K_1(\boldsymbol{x}) = \boldsymbol{x}$. Now imagine the input space $\boldsymbol{x}$ can be decomposed into the direct sum of a semantically relevant subspace, and its orthogonal complement, which corresponds to a subspace of nuisance features. Furthermore, suppose the augmentation distribution $p_{\text{aug}}(\cdot|\boldsymbol{x})$ is obtained by multiplying $\boldsymbol{x}$ by a random Gaussian matrix that acts *only* in the nuisance subspace, thereby identically preserving the semantic subspace. Then the augmentation averaged connection $\bar{K}_1(\boldsymbol{x}) = Q^s\boldsymbol{x}$ where Q^s is a projection operator onto the semantic subspace. In essence, only the projection of data onto the semantic subspace survives augmentation averaging, as the nuisance subspace is scrambled. Then the covariance operator in Theorem 3 is $\mathbb{V}_{\boldsymbol{x}}[\bar{K}_1(\boldsymbol{x})] = Q^s\mathbb{V}_{\boldsymbol{x}}[\boldsymbol{x}]Q^{s\intercal}$. Thus the covariance of the data distribution, projected onto the semantic subspace, governs the growth of the weight vector W_1, demonstrating SimCLR on a single linear neuron performs PCA within a semantic subspace preserved by data augmentation.

A single linear neuron cannot detect localized objects. We now consider a generative model in which data vectors can be thought of as images of objects of the form $\boldsymbol{x}(z_0, z')$ where z_0 is an important latent semantic variable denoting object identity, while z' is an unimportant latent variable denoting nuisance features, like object pose or location. The augmentation procedure scrambles pose/position while preserving object identity. Consider a simple concrete example (Fig. 3(a)):

$$\boldsymbol{x}(z_0, z') = \begin{cases} \boldsymbol{e}_{z'} + \boldsymbol{e}_{(z'+1) \bmod d} & z_0 = 1 \\ \boldsymbol{e}_{z'} + \boldsymbol{e}_{(z'+2) \bmod d} & z_0 = 2, \end{cases} \tag{12}$$

Here $0 \le z' \le d-1$ denotes d discrete translational object positions on a periodic ring and $z_0 \in \{1, 2\}$ denotes two possible objects `11` and `101`. The distribution is uniform both over objects and positions: $p(z_0, z') = \frac{1}{2d}$. Augmentation shifts the object to a uniformly random position via $p_{\text{aug}}(z'|z_0) = 1/d$. For a single linear neuron $K_1(\boldsymbol{x}) = \boldsymbol{x}$, and the augmentation-averaged connection is $\bar{K}_1(z_0) = \frac{2}{d}\mathbf{1}$, and is actually independent of object identity z_0 (both objects activate two pixels at any location). Thus $\mathbb{V}_{z_0}\left[\bar{K}_1(z_0)\right] = 0$ and no learning happens.

A local receptive field (RF) does not help. In the same generative model, now consider a linear neuron with a local RF of width 2. Within the RF only four patterns can arise: `00`, `01`, `10`, `11`. Taking the expectation over z' given z_0 (Fig. 3(a2)) yields $\bar{K}_1(z_0{=}1) = \frac{1}{d}\left[\boldsymbol{x}_{11} + \boldsymbol{x}_{01} + \boldsymbol{x}_{10} + (d-3)\boldsymbol{x}_{00}\right]$



Pattern	10	01	00	11
$z_0 = 1$	1	1	d-3	1
$z_0 = 2$	2	2	d-4	0

Figure 3: **(a)** Two 1D objects under translation: *(a1)* Two different objects `11` ($z_0 = 1$) and `101` ($z_0 = 2$) located at different locations specified by z'. *(a2)* The frequency table for a neuron with local receptive field of size 2. **(b)** In two-layer case (Fig. 2(c)), $\mathbb{V}[\bar{K}_1]$ and $\mathbb{V}[\bar{K}_2]$ interplay in two-cluster data distribution.

and $\bar{K}_1(z_0{=}2) = \frac{1}{d}\left[2\boldsymbol{x}_{01} + 2\boldsymbol{x}_{10} + (d-4)\boldsymbol{x}_{00}\right]$. Here, $\boldsymbol{x}_{11} \in \mathbb{R}^2$ denotes pattern `11`. This yields

$$\mathbb{V}_{z_0}\left[\bar{K}_1(z_0)\right] = \frac{1}{4d^2}\boldsymbol{u}\boldsymbol{u}^\intercal \qquad \text{where } \boldsymbol{u} := \boldsymbol{x}_{11} + \boldsymbol{x}_{00} - \boldsymbol{x}_{01} - \boldsymbol{x}_{10}. \tag{13}$$

and $\mathbb{V}_{z_0}\left[\bar{K}_1(z_0)\right] \in \mathbb{R}^{2\times 2}$ since the RF has width 2. Note that the signed sum of the four pattern vectors in $\boldsymbol{u}$ actually cancel, so that $\boldsymbol{u} = \boldsymbol{0}$, implying $\mathbb{V}_{z_0}\left[\bar{K}_1(z_0)\right] = 0$ and no learning happens. Interestingly, although the conditional distribution of the 4 input patterns depends on the object identity z_0 (Fig. 3(a2)), a linear neuron cannot learn to discriminate the objects.

A nonlinear neuron with local RF can learn to detect object selective features. With a ReLU neuron with weight vector $\boldsymbol{w}$, from Def. 1, the connection is now $K_1(\boldsymbol{x}, \boldsymbol{w}) = \psi'(\boldsymbol{w}^\intercal\boldsymbol{x})\boldsymbol{x}$. Suppose $\boldsymbol{w}(t)$ happens to be selective for a *single* pattern $\boldsymbol{x}_p$ (where $p \in \{\texttt{00}, \texttt{01}, \texttt{10}, \texttt{11}\}$), i.e., $\boldsymbol{w}(t)^\intercal\boldsymbol{x}_p > 0$ and $\boldsymbol{w}(t)^\intercal\boldsymbol{x}_{p'} < 0$ for $p' \neq p$. The augmentation averaged connection is then $\bar{K}_1(z_0) \propto \boldsymbol{x}_p$ where the proportionality constant depends on object identity z_0 and can be read off (Fig. 3(a2)). Since this averaged connection varies with object identity z_0 for all p, the covariance operator does not vanish and is given by $\mathbb{V}_{z_0}\left[\bar{K}_1(z_0)\right] = c_p\boldsymbol{x}_p\boldsymbol{x}_p^\intercal$ where the constant $c_p > 0$ depends on the selective pattern p and can be computed from Fig. 3(a2). By Theorem 3, the dot product $\boldsymbol{x}_p^\intercal\boldsymbol{w}(t)$ grows over time:

$$\boldsymbol{x}_p^\intercal\boldsymbol{w}(t+1) = \boldsymbol{x}_p^\intercal\left(I_{2\times 2} + \alpha\beta c_p\boldsymbol{x}_p\boldsymbol{x}_p^\intercal\right)\boldsymbol{w}(t) = \left(1 + \alpha\beta c_p\|\boldsymbol{x}_p\|^2\right)\boldsymbol{x}_p^\intercal\boldsymbol{w}_j(t) > \boldsymbol{x}_p^\intercal\boldsymbol{w}_j(t) > 0. \tag{14}$$

Thus the learning dynamics amplifies the initial selectivity to the object selective feature vector $\boldsymbol{x}_p$ in a way that cannot be done with a linear neuron. Note this argument also holds with bias terms and initial selectivity for more than one patterns. Moreover, with a local RF, the probability of weak initial selectivity to local object sensitive features is high, and we may expect amplification of such weak selectivity in real neural network training.

A.2 Two-layer case with multiple hidden neurons

Now consider a two-layer network ($L = 2$) with 1 output ($n_L = 1$) and 1-hidden layer with n_1 ReLU neurons (Fig. 2(c)). In this case, the augmentation-averaged connection $\bar{K}_1(z_0)$ at the lowest layer $l = 1$ can be written as

$$\bar{K}_1(z_0) = [w_{2,1}\boldsymbol{u}_1^\intercal(z_0), w_{2,2}\boldsymbol{u}_2^\intercal(z_0), \ldots, w_{2,n_1}\boldsymbol{u}_{n_1}^\intercal(z_0)]^\intercal \in \mathbb{R}^{n_1 d} \tag{15}$$

where $\boldsymbol{u}_j(z_0) := \mathbb{E}_{z'|z_0}\left[\boldsymbol{x}(z_0, z')\mathbb{I}(\boldsymbol{w}_{1,j}^\intercal\boldsymbol{x}(z_0, z') \geq 0)\right]$ is the expected activation of the hidden layer. Note that each $\boldsymbol{u}_j(z_0) \in \mathbb{R}^d$ and $w_{2,j}$ is the scalar weight of the second weight matrix that connects node j to the output. Then we have the following theorem:

Theorem 5. *If* $\mathrm{Cov}_{z_0}[\boldsymbol{u}_j, \boldsymbol{u}_k] = 0$ *for* $j \neq k$*, then the time derivative of* $w_{2,j}$ *and* $\boldsymbol{w}_{1,j}$ *satisfies:*

$$\dot{w}_{2,j} = w_{2,j}\boldsymbol{w}_{1,j}^\intercal A_j\boldsymbol{w}_{1,j}, \quad \dot{\boldsymbol{w}}_{1,j} = w_{2,j}^2 A_j\boldsymbol{w}_{1,j}, \quad \textit{where } A_j := \mathbb{V}_{z_0}[\boldsymbol{u}_j(z_0)]. \tag{16}$$

It is easy to see that $\mathrm{d}|w_{2,j}|^2/\mathrm{d}t = \mathrm{d}\|\boldsymbol{w}_{1,j}\|^2/\mathrm{d}t$ and thus $w_{2,j}^2 = \|\boldsymbol{w}_{1,j}\|^2 + c$ where c is some constant which doesn't change over time. Since A_j is PSD, $\boldsymbol{w}_{1,j}^\intercal A_j\boldsymbol{w}_{1,j} \geq 0$ and $w_{2,j}$ always grows, which in turn accelerates the convergence of $\boldsymbol{w}_{1,j}$ to the largest eigenvector of A_j. This shows *top-down modulation*: the existence of top-layer weights accelerate the training of the lower layer. Previous works [1] also mention a similar concept in supervised learning, called "backward feature corrections", for deep polynomial networks with quadratic activations. Here we provide a rigorous formulation showing similar behaviors happen in SSL with SGD training.

Note that if we consider ReLU neurons, A_j changes with $\boldsymbol{w}_{1,j}$; while for linear nodes, A_j is a constant, since the gating $\mathbb{I}(\boldsymbol{w}_{1,j}^\intercal\boldsymbol{x} > 0)$ is always 1. If the data $\boldsymbol{x}$ forms a mixture of two Gaussians:

Symbol	Definition	Size	Description
$\mathcal{N}_l$, $\mathcal{Z}_l$			The set of all nodes and all latent variables at layer l.
$\mathcal{N}_\mu, \mathcal{N}_\mu^{\text{ch}}$			Nodes corresponding to latent variable z_μ. $\mathcal{N}_\mu^{\text{ch}}$ are children under $\mathcal{N}_\mu$.
$P_{\mu\nu}$	$[\mathbb{P}(z_\nu \vert z_\mu)]$	2×2	The top-down transition probability from z_μ to z_ν.
$v_j(z_\mu)$, $\boldsymbol{v}_j$	$\mathbb{E}_z[f_j \vert z_\mu]$, $[v_j(z_\mu)]$	scalar, 2	Expected activation f_j given z_μ (z_μ's descendants are marginalized).
$\boldsymbol{f}_\mu$, $\boldsymbol{f}_{\mathcal{N}_\mu^{\text{ch}}}$	$[f_j]_{j\in\mathcal{N}_\mu}$, $[f_k]_{k\in\mathcal{N}_\mu^{\text{ch}}}$	$\vert\mathcal{N}_\mu\vert$, $\vert\mathcal{N}_\mu^{\text{ch}}\vert$	Activations for all nodes $j \in \mathcal{N}_\mu$ and for the children of $\mathcal{N}_\mu$
$\rho_{\mu\nu}$	$2\mathbb{P}(z_\nu{=}1 \vert z_\mu{=}1) - 1$	scalar in $[-1, 1]$	Polarity of the transitional probability.
ρ_0	$\mathbb{P}(z_0 = 1) - \mathbb{P}(z_0 = 0)$	scalar	Polarity of probability of root latent z_0.
s_k	$\frac{1}{2}(v_k(1) - v_k(0))$	scalar	Discrepancy of node k w.r.t its latent variable $z_{\nu(k)}$.
$\boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}}$	$[\rho_{\mu\nu(k)} s_k]_{k\in\mathcal{N}_\mu^{\text{ch}}}$	$\vert\mathcal{N}_\mu^{\text{ch}}\vert$	Child selectivity vector.

Table 1: Notation for Sec. A.3 (binary symmetric HLTM).

$\boldsymbol{x} \sim \frac{1}{2}\mathbb{I}(z_0{=}1)N(\boldsymbol{w}_1^*, \sigma^2) + \frac{1}{2}\mathbb{I}(z_0{=}2)N(\boldsymbol{w}_2^*, \sigma^2)$ and let $\Delta\boldsymbol{w}^* := \boldsymbol{w}_1^* - \boldsymbol{w}_2^*$, then in the linear case, $A_j \sim \Delta\boldsymbol{w}^*\Delta\boldsymbol{w}^{*\intercal}$ and $\boldsymbol{w}_{1,j}$ converges to $\pm\Delta\boldsymbol{w}^*$ (Fig. 3(b)). In the nonlinear case with multiple Gaussians, if one of the Gaussians sits at the origin (e.g., background noise), then dependent on initialization, A_j evolves into $\boldsymbol{w}_k^*\boldsymbol{w}_k^{*\intercal}$ for some center k, and $\boldsymbol{w}_{1,j} \to \boldsymbol{w}_k^*$. Note this dynamics is insensitive to specific parametric forms of the input data.

A.3 Deep ReLU SSL training with Hierarchical Latent Tree Models (HLTM)

For multi-layer case, we mainly study a hierarchical latent tree model, where each leaf variable is sampled via a Markov Chain from the root latent variable z_0. The model is motivated by the structure of ConvNet: each latent variable at the intermediate layer generates a *subset* of the observable data $\boldsymbol{x}$ that is sent to the deep ReLU networks, in which the nodes also have local receptive fields (Fig. 2(d)).

We define symbols in Tbl. 1. At layer l, we have categorical latent variables $\{z_\mu\}$, where $\mu \in \mathcal{Z}_l$ is the Greek letter index for latent variables. Each z_μ can take discrete values. The topmost latent variable is z_0. Following the tree structure, for $\mu \in \mathcal{Z}_l$ and $\nu_1, \nu_2 \in \mathcal{Z}_{l-1}$, conditional independence holds: $\mathbb{P}(z_{\nu_1}, z_{\nu_2} \vert z_\mu) = \mathbb{P}(z_{\nu_1} \vert z_\mu)\mathbb{P}(z_{\nu_2} \vert z_\mu)$. The final sample $\boldsymbol{x}$ is the instantiation of the leaf latents (Fig. 2(d)), and thus depends on all latent variables. Corresponding to the hierarchical tree model, each neural network node $j \in \mathcal{N}_l$ maps to a unique $\mu = \mu(j) \in \mathcal{Z}_l$. Let $\mathcal{N}_\mu$ be all nodes that map to μ. For $j \in \mathcal{N}_\mu$, its activation f_j only depends on the value of z_μ and its descendant latent variables. We also define $v_j(z_\mu) := \mathbb{E}_z[f_j \vert z_\mu]$ as the expected activation w.r.t z_μ. Given a sample $\boldsymbol{x}$, the data augmentation is done by *resampling* all z_μ (which are z' in Fig. 2), given the root z_0.

Symmetric Binary HLTM. Here we consider a symmetric binary case: each $z_\mu \in \{0, 1\}$ and for $\mu \in \mathcal{Z}_l, \nu \in \mathcal{Z}_{l-1}$, $\mathbb{P}(z_\nu{=}1 \vert z_\mu{=}1) = \mathbb{P}(z_\nu{=}0 \vert z_\mu{=}0) = (1 + \rho_{\mu\nu})/2$, where the *polarity* $\rho_{\mu\nu} \in [-1, 1]$ measures how informative $z_\mu = 1$ is. If $\rho_{\mu\nu} = \pm 1$ then there is no stochasticity in the top-down generation process; $\rho_{\mu\nu} = 0$ means no information in the downstream latents and the posterior of z_0 given the observation $\boldsymbol{x}$ can only be uniform. See Appendix for more general cases.

Now we compute covariance operator $\mathbb{V}_{z_0}[\bar{K}_\mu(z_0)]$ at different layers, where $\bar{K}_\mu(z_0) = \mathbb{E}_{z'}\left[\boldsymbol{f}_{\mathcal{N}_\mu^{\text{ch}}} \otimes J_\mu^\intercal \vert z_0\right]$, we first check the term $\mathbb{E}_{z'}\left[\boldsymbol{f}_{\mathcal{N}_\mu^{\text{ch}}} \vert z_0\right]$ and incorporate the Jacobian later.

Theorem 6 (Covariance operator in binary HLTM)**.** $\mathbb{V}_{z_0}[\mathbb{E}_{z'}\left[\boldsymbol{f}_{\mathcal{N}_\mu^{\text{ch}}} \vert z_0\right]] = \rho_{0\mu}^2(1 - \rho_0^2)\boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}}\boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}}^\intercal$. *Here* $\boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}} := [\rho_{\mu\nu(k)} s_k]_{k\in\mathcal{N}_\mu^{\text{ch}}}$. *If* $\max_{\alpha\beta} \vert\rho_{\alpha\beta}\vert \le \gamma < 1$, *then* $\lim_{L\to+\infty} \rho_{0\mu} \to 0$ *for* $\mu \in \mathcal{N}_l$.

Theorem 6 suggests when $\rho_{0\mu}$ and $\|\boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}}\|$ is large, the covariance operator is large and training is faster. For deep HLTM and deep networks, at lower layers, $\rho_{0\mu} \to 0$ and $P_{0\mu}$ becomes uniform due to mixing from the progression of the Markov Chain, making $\mathbb{V}_{z_0}[\bar{K}_l(z_0)]$ small. Therefore, training in SSL is faster at the top layers where the covariance operators have large magnitude. On the other hand, large $\|\boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}}\|$ means that $s_k := (v_k(1) - v_k(0))/2$ is large, or the expected activation $v_k(z_\nu)$ is *selective* among different values of z_ν for $\nu \in \text{ch}(\mu)$. Interestingly, this can be achieved by *over-parameterization* (i.e., $\vert\mathcal{N}_\mu\vert > 1$):

Theorem 7 (Lucky nodes in deep ReLU networks regarding to binary HLTM)**.** *Suppose each element of the weights* W_l *between layer* $l + 1$ *and* l *are initialized with* $\text{Uniform}\left[-\sigma_w\sqrt{3/\vert\mathcal{N}_\mu^{\text{ch}}\vert}, \sigma_w\sqrt{3/\vert\mathcal{N}_\mu^{\text{ch}}\vert}\right]$. *There exists* σ_l^2 *so that* $\mathbb{V}[f_k \vert z_\nu] \le \sigma_l^2$ *for any* $k \in \mathcal{N}_l$. *For any* $\mu \in \mathcal{Z}_{l+1}$, *if* $\vert\mathcal{N}_\mu\vert = O(\exp(c))$, *then with high probability, there exists at least one node* $j \in \mathcal{N}_\mu$ *so that their* pre-activation gap $\vert\tilde{v}_j(1) - \tilde{v}_j(0)\vert = 2\boldsymbol{w}_j^\intercal\boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}} > 0$ *and the activations satisfy:*

$$\left|v_j^2(1) - v_j^2(0)\right| \geq 3\sigma_w^2 \left[\frac{1}{4|\mathcal{N}_\mu^{\rm ch}|}\sum_{k\in\mathcal{N}_\mu^{\rm ch}} |v_k(1)-v_k(0)|^2 \left(\frac{c+6}{6}\rho_{\mu\nu}^2 - 1\right) - \sigma_l^2\right]. \tag{17}$$

Intuitively, this means that with large polarity $\rho_{\mu\nu}$ (or strong signals sending top-down), over-parameterized ReLU networks with random initialization yields selective neurons, if the lower layer also contains selective ones. For example, when $\sigma_l = 0$, $c = 9$, if $\rho_{\mu\nu} \geq 63.3\%$ then there is a gap between expected activations $v_j(1)$ and $v_j(0)$, and the gap is larger when the selectivity in the lower layer l is higher. Note that at the lowest layer, $\{v_k\}$ are themselves observable leaf latent variables and are selective by definition. So a bottom-up mathematical induction will happen.

After the gradient update, if we assume the top-down Jacobian is just 1, then for the "lucky" node j we will have (for brevity, let $c := \alpha\beta\rho_{0\mu}^2(1-\rho_0^2)$):

$$\boldsymbol{a}_{\mathcal{N}_\mu^{\rm ch}}^\intercal \boldsymbol{w}_j(t+1) = \boldsymbol{a}_{\mathcal{N}_\mu^{\rm ch}}^\intercal \left[I + c\boldsymbol{a}_{\mathcal{N}_\mu^{\rm ch}}\boldsymbol{a}_{\mathcal{N}_\mu^{\rm ch}}^\intercal\right]\boldsymbol{w}_j(t) = (1+c\|\boldsymbol{a}_{\mathcal{N}_\mu^{\rm ch}}\|_2^2)\boldsymbol{a}_{\mathcal{N}_\mu^{\rm ch}}^\intercal \boldsymbol{w}_j(t) > \boldsymbol{a}_{\mathcal{N}_\mu^{\rm ch}}^\intercal \boldsymbol{w}_j(t) > 0$$

which means that the pre-activation gap $\tilde{v}_j(1) - \tilde{v}_j(0) = 2\boldsymbol{w}_j^\intercal \boldsymbol{a}_{\mathcal{N}_\mu^{\rm ch}}$ grows over time and the latent variable z_μ is *learned* (instantiated as f_j) during training, even if the network is never supervised with its true value. Similar to Sec. A.2, once the top layer starts to have large weights, $\bar{K}_l(z_0)$ for lower layer becomes larger due to large Jacobian and training is accelerated.

We implement the `HLTM` and confirm, as predicted by our theory, that the intermediate layers of deep ReLU networks do indeed learn the latent variables of the `HLTM` (see Tbl. 2 below).

B Proofs: Background and Basic Setting (Section 2)

B.1 Theorem 1

Definition 2 (reversibility)**.** *A layer l is* reversible *if there is a $G_l(\boldsymbol{x};\mathcal{W}) \in \mathbb{R}^{n_l\times n_{l-1}}$ so that $\boldsymbol{f}_l(\boldsymbol{x};\mathcal{W}) = G_l(\boldsymbol{x};\mathcal{W})\boldsymbol{f}_{l-1}(\boldsymbol{x};\mathcal{W})$ and $\boldsymbol{g}_{l-1} = G_l^\intercal(\boldsymbol{x};\mathcal{W})Q_l\boldsymbol{g}_l$ for some constant PSD matrix $\mathbb{R}^{n_l\times n_l} \ni Q_l \succeq 0$. A network is* reversible *if all layers are.*

Note that many different kinds of layers have this reversible property, including linear layers (MLP and Conv) and (leaky) ReLU nonlinearity. For multi-layer ReLU network, for each layer l, we have:

$$G_l(\boldsymbol{x};\mathcal{W}) = D_l(\boldsymbol{x};\mathcal{W})W_l, \qquad Q_l \equiv I_{n_l\times n_l} \tag{18}$$

where $D_l \in \mathbb{R}^{n_l\times n_l}$ is a binary diagonal matrix that encodes the gating of each neuron at layer l. The gating $D_l(\boldsymbol{x};\mathcal{W})$ depends on the current input $\boldsymbol{x}$ and current weight $\mathcal{W}$.

In addition to ReLU, other activation function also satisfies this condition, including linear, LeakyReLU and monomial activations. For example, for power activation $\psi(x) = x^p$ where $p > 1$, we have (where $\tilde{\boldsymbol{f}}_l$ is the pre-activation at layer l):

$$G_l(\boldsymbol{x};\mathcal{W}) = \mathrm{diag}^{p-1}(\tilde{\boldsymbol{f}}_l)W_l, \qquad Q_l \equiv pI_{n_l\times n_l} \tag{19}$$

Remark. Note that the reversibility is not the same as invertible. Specifically, reversibility only requires the transfer function of a backpropagation gradient is a transpose of the forward function.

Lemma 1 (Recursive Gradient Update (Extension to Lemma 1 in [31])**.** *Let (pseudo)-Jacobian matrix $J_L(\boldsymbol{x}) = I_{n_L\times n_L}$, and recursively define $J_{l-1}(\boldsymbol{x}) := J_l(\boldsymbol{x})\sqrt{Q_l}G_l(\boldsymbol{x}) \in \mathbb{R}^{n_l\times n_{l-1}}$. Here $\sqrt{Q_l}$ is the constant PSD matrix so that $\sqrt{Q_l}\sqrt{Q_l} = Q_l \succeq 0$.*

*If **(1)** the network is reversible (Def. 2) and **(2)** $\sqrt{Q_l}$ commutes with $J_l(\boldsymbol{x}_1)^\intercal J_l(\boldsymbol{x}_1)$ and $J_l(\boldsymbol{x}_1)^\intercal J_l(\boldsymbol{x}_2)$, then minimizing the ℓ_2 objective*

$$r(\mathcal{W}_1) := \frac{1}{2}\|\boldsymbol{f}_L(\boldsymbol{x}_1;\mathcal{W}_1) - \boldsymbol{f}_L(\boldsymbol{x}_2;\mathcal{W}_2)\|_2^2 \tag{20}$$

with respect to weight matrix W_l at layer l yields the following gradient at layer l:

$$\boldsymbol{g}_l = J_l^\intercal(\boldsymbol{x}_1;\mathcal{W}_1)\left[J_l(\boldsymbol{x}_1;\mathcal{W}_1)\boldsymbol{f}_l(\boldsymbol{x}_1;\mathcal{W}_1) - J_l(\boldsymbol{x}_2;\mathcal{W}_2)\boldsymbol{f}_l(\boldsymbol{x}_2;\mathcal{W}_2)\right] \tag{21}$$

Proof. We prove by induction. Note that our definition of W_l is the transpose of W_l defined in [31]. Also our $\boldsymbol{g}_l(\boldsymbol{x})$ is the gradient *before* nonlinearity, while [31] uses the same symbol for the gradient after nonlinearity.

For notation brievity, we let $\boldsymbol{f}_l(\boldsymbol{x}_1) := \boldsymbol{f}_l(\boldsymbol{x}_1; \mathcal{W}_l)$ and $G_l(\boldsymbol{x}_1) := G_l(\boldsymbol{x}_1; \mathcal{W}_l)$. Similar for $\boldsymbol{x}_2$ and W_2.

When $l = L$, by the property of ℓ_2-loss, we know that $\boldsymbol{g}_L = \boldsymbol{f}_L(\boldsymbol{x}_1; \mathcal{W}_1) - \boldsymbol{f}_L(\boldsymbol{x}_2; \mathcal{W}_2)$, by setting $J_L(\boldsymbol{x}_1) = J_L(\boldsymbol{x}_2) = I$, the condition holds. Now suppose for layer l, we have:

$$\boldsymbol{g}_l = J_l^\intercal(\boldsymbol{x}_1)\left[J_l(\boldsymbol{x}_1)\boldsymbol{f}_l(\boldsymbol{x}_1) - J_l(\boldsymbol{x}_2)\boldsymbol{f}_l(\boldsymbol{x}_2)\right] \tag{22}$$

Then:

$$\begin{aligned}
\boldsymbol{g}_{l-1} &= G_l^\intercal(\boldsymbol{x}_1) Q_l \boldsymbol{g}_l && (23)\\
&= G_l^\intercal(\boldsymbol{x}_1) Q_l J_l^\intercal(\boldsymbol{x}_1)\left[J_l(\boldsymbol{x}_1)\boldsymbol{f}_l(\boldsymbol{x}_1) - J_l(\boldsymbol{x}_2)\boldsymbol{f}_l(\boldsymbol{x}_2)\right] && (24)\\
&= \underbrace{G_l^\intercal(\boldsymbol{x}_1)\sqrt{Q_l} J_l^\intercal(\boldsymbol{x}_1)}_{J_{l-1}^\intercal(\boldsymbol{x}_1)} \cdot \left[J_l(\boldsymbol{x}_1)\sqrt{Q_l}\boldsymbol{f}_l(\boldsymbol{x}_1) - J_l(\boldsymbol{x}_2)\sqrt{Q_l}\boldsymbol{f}_l(\boldsymbol{x}_2)\right] && (25)\\
&= J_{l-1}^\intercal(\boldsymbol{x}_1)\left[\underbrace{J_l(\boldsymbol{x}_1)\sqrt{Q_l}G_l(\boldsymbol{x}_1)}_{J_{l-1}(\boldsymbol{x}_1)}\boldsymbol{f}_{l-1}(\boldsymbol{x}_1) - \underbrace{J_l(\boldsymbol{x}_2)\sqrt{Q_l}G_l(\boldsymbol{x}_2)}_{J_{l-1}(\boldsymbol{x}_2)}\boldsymbol{f}_{l-1}(\boldsymbol{x}_2)\right] && (26)\\
&= J_{l-1}^\intercal(\boldsymbol{x}_1)\left[J_{l-1}(\boldsymbol{x}_1)\boldsymbol{f}_{l-1}(\boldsymbol{x}_1) - J_{l-1}(\boldsymbol{x}_2)\boldsymbol{f}_{l-1}(\boldsymbol{x}_2)\right] && (27)
\end{aligned}$$

Note that for multi-layered ReLU network, $G_l(\boldsymbol{x}) = D_l(\boldsymbol{x})W_l$, $Q_l = I$ for each ReLU+Linear layer, if we set $\boldsymbol{x}_1 = \boldsymbol{x}_2 = \boldsymbol{x}$, $\mathcal{W}_1 = \mathcal{W}$, $\mathcal{W}_2 = \mathcal{W}^*$ (teacher weights), then we go back to the original Lemma 1 in [31]. □

Remark on the top-most ℓ_2-normalization layer. For ℓ_2-normalized layer $\boldsymbol{f}_l := \boldsymbol{f}_{l-1}/\|\boldsymbol{f}_{l-1}\|_2$, we have $G_l := 1/\|\boldsymbol{f}_{l-1}\|_2 \cdot I_{n_l \times n_l}$ and due to the following identity (here $\tilde{\boldsymbol{y}} := \boldsymbol{y}/\|\boldsymbol{y}\|_2$):

$$\frac{\partial \tilde{\boldsymbol{y}}}{\partial \boldsymbol{y}} = \frac{1}{\|\boldsymbol{y}\|_2}\left(I - \tilde{\boldsymbol{y}}\tilde{\boldsymbol{y}}^\intercal\right) \tag{28}$$

Therefore we have $\partial \boldsymbol{f}_l / \partial \boldsymbol{f}_{l-1} = (I_{n_l \times n_l} - \boldsymbol{f}_l \boldsymbol{f}_l^\intercal) G_l$ and we could set $Q_l := I - \boldsymbol{f}_l \boldsymbol{f}_l^\intercal$, which is a projection matrix and thus PSD. Furthermore, since the normalization layer is at the topmost, $J_l = I$ and Q_l trivially commutes with $J_l^\intercal J_l$.

The *only issue* is that Q_l is not a constant PSD matrix and can change over training. Therefore Lemma 1 doesn't apply exactly to such a layer but can be regarded as an approximate way to model.

Remark on ResNet. Note that the same structure holds for blocks of ResNet with ReLU activation.

Now we prove Theorem 1 in a more general setting where the network is reversible (note that deep ReLU networks are included and its Q_l is a simple identity matrix):

Lemma 2 (Squared ℓ_2 Gradient for dual deep reversible networks)**.** *The gradient g_{W_l} of the squared loss r with respect to $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$ for a single input pair $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ is:*

$$g_{W_l} = \mathrm{vec}\left(\partial r / \partial W_{1,l}\right) = K_{1,l}\left[K_{1,l}^\intercal \mathrm{vec}(W_{1,l}) - K_{2,l}^\intercal \mathrm{vec}(W_{2,l})\right]. \tag{29}$$

Here $K_l(\boldsymbol{x}; \mathcal{W}) := \boldsymbol{f}_{l-1}(\boldsymbol{x}; \mathcal{W}) \otimes J_l^\intercal(\boldsymbol{x}; \mathcal{W})$, $K_{1,l} := K_l(\boldsymbol{x}_1; \mathcal{W}_1)$ and $K_{2,l} := K_l(\boldsymbol{x}_2; \mathcal{W}_2)$.

Proof. We consider more general case where the two towers have different parameters, namely $\mathcal{W}_1$ and $\mathcal{W}_2$. Applying Lemma 1 for the branch with input $\boldsymbol{x}_1$ at the linear layer l, and we have:

$$\boldsymbol{g}_{1,l} = J_{1,l}^\intercal [J_{1,l} W_{1,l} \boldsymbol{f}_{1,l-1} - J_{2,l} W_{2,l} \boldsymbol{f}_{2,l-1}] \tag{30}$$

where $\boldsymbol{f}_{1,l-1} := \boldsymbol{f}_{l-1}(\boldsymbol{x}_1; \mathcal{W}_1)$ is the activation of layer $l-1$ just below the linear layer at tower 1 (similar for other symbols). The gradient (and the weight update, according to gradient descent) of the weight W_l between layer l and layer $l-1$ is:

$$\begin{aligned}
\frac{\partial r}{\partial W_{1,l}} &= \boldsymbol{g}_{1,l} \boldsymbol{f}_{1,l-1}^\intercal && (31)\\
&= J_{1,l}^\intercal J_{1,l} W_{1,l} \boldsymbol{f}_{1,l-1} \boldsymbol{f}_{1,l-1}^\intercal - J_{1,l}^\intercal J_{2,l} W_{2,l} \boldsymbol{f}_{2,l-1} \boldsymbol{f}_{1,l-1}^\intercal && (32)
\end{aligned}$$

Using $\mathrm{vec}(AXB) = (B^\intercal \otimes A)\mathrm{vec}(X)$ (where $\otimes$ is the Kronecker product), we have:

$$\mathrm{vec}\left(\frac{\partial r}{\partial W_{1,l}}\right) = \left(\boldsymbol{f}_{1,l-1}\boldsymbol{f}_{1,l-1}^\intercal \otimes J_{1,l}^\intercal J_{1,l}\right)\mathrm{vec}(W_{1,l}) - \left(\boldsymbol{f}_{1,l-1}\boldsymbol{f}_{2,l-1}^\intercal \otimes J_{1,l}^\intercal J_{2,l}\right)\mathrm{vec}(W_{2,l}) \tag{33}$$

Let

$$K_l(\boldsymbol{x}; W) := \boldsymbol{f}_{l-1}(\boldsymbol{x}; \mathcal{W}) \otimes J_l^\intercal(\boldsymbol{x}; \mathcal{W}) \in \mathbb{R}^{n_l n_{l-1} \times n_L} \tag{34}$$

Note that $K_l(\boldsymbol{x}; \mathcal{W})$ is a function of the current weight W, which includes weights at all layers. By the mixed-product property of Kronecker product $(A \otimes B)(C \otimes D) = AC \otimes BD$, we have:

$$\begin{aligned}
\mathrm{vec}\left(\frac{\partial r}{\partial W_{1,l}}\right) &= K_l(\boldsymbol{x}_1)K_l(\boldsymbol{x}_1)^\intercal \mathrm{vec}(W_{1,l}) - K_l(\boldsymbol{x}_1)K_l(\boldsymbol{x}_2)^\intercal \mathrm{vec}(W_{2,l}) &(35)\\
&= K_l(\boldsymbol{x}_1)\left[K_l(\boldsymbol{x}_1)^\intercal \mathrm{vec}(W_{1,l}) - K_l(\boldsymbol{x}_2)^\intercal \mathrm{vec}(W_{2,l})\right] &(36)
\end{aligned}$$

where $K_l(\boldsymbol{x}_1) = K_l(\boldsymbol{x}_1; \mathcal{W}_1)$ and $K_l(\boldsymbol{x}_2) = K_l(\boldsymbol{x}_2; \mathcal{W}_2)$.

In SimCLR case, we have $\mathcal{W}_1 = \mathcal{W}_2 = \mathcal{W}$ so

$$\mathrm{vec}\left(\frac{\partial r}{\partial W_l}\right) = K_l(\boldsymbol{x}_1)\left[K_l(\boldsymbol{x}_1) - K_l(\boldsymbol{x}_2)\right]^\intercal \mathrm{vec}(W_l) \tag{37}$$

□

C Proofs: Analysis of SimCLR using Teacher-Student Setting (Section 3)

C.1 Theorem 2

Proof. we have:

$$\begin{aligned}
\frac{\partial L}{\partial r_+} &= \frac{1}{\tau}\left(1 - \frac{e^{-r_+/\tau}}{e^{-r_+/\tau} + \sum_{k'=1}^H e^{-r_{k'-}/\tau}}\right) > 0 &(38)\\
\frac{\partial L}{\partial r_{k-}} &= -\frac{1}{\tau}\left(\frac{e^{-r_{k-}/\tau}}{e^{-r_+/\tau} + \sum_{k'=1}^H e^{-r_{k'-}/\tau}}\right) < 0, \qquad k = 1, \ldots, H &(39)
\end{aligned}$$

and obviously we have:

$$\frac{\partial L}{\partial r_+} + \sum_{k=1}^H \frac{\partial L}{\partial r_{k-}} = 0 \tag{40}$$

□

C.2 Theorem 3

Proof. First we have:

$$\mathrm{vec}(g_{W_l}) = \frac{\partial L}{\partial W_l} = \frac{\partial L}{\partial r_+}\frac{\partial r_+}{\partial W_l} + \sum_{k=1}^H \frac{\partial L}{\partial r_{k-}}\frac{\partial r_{k-}}{\partial W_l} \tag{41}$$

Then we compute each terms. Using Theorem 1, we know that:

$$\begin{aligned}
\frac{\partial r_+}{\partial W_l} &= K_l(\boldsymbol{x}_1)(K_l(\boldsymbol{x}_1) - K_l(\boldsymbol{x}_+))^\intercal \mathrm{vec}(W_l) &(42)\\
\frac{\partial r_{k-}}{\partial W_l} &= K_l(\boldsymbol{x}_1)(K_l(\boldsymbol{x}_1) - K_l(\boldsymbol{x}_{k-}))^\intercal \mathrm{vec}(W_l), \quad k = 1, \ldots, n &(43)
\end{aligned}$$

Since Eqn. 40 holds, $K_l(\boldsymbol{x}_1)K_l^\intercal(\boldsymbol{x}_1)$ will be cancelled out and we have:

$$\mathrm{vec}(g_{W_l}) = K_l(\boldsymbol{x}_1)\sum_{k=1}^H \left[\frac{\partial L}{\partial r_{k-}}(K_l(\boldsymbol{x}_+) - K_l(\boldsymbol{x}_{k-}))^\intercal\right]\mathrm{vec}(W_l) \tag{44}$$

Then we use the assumption that $\frac{\partial L}{\partial r_{k-}} = -\beta/H$ with $\beta > 0$:

$$\mathrm{vec}(g_{W_l}) = -\beta K_l(\boldsymbol{x}_1)\left[K_l^\intercal(\boldsymbol{x}_+) - \frac{1}{n}\sum_{k=1}^{H} K_l^\intercal(\boldsymbol{x}_{k-})\right]\mathrm{vec}(W_l) \tag{45}$$

Taking large batch limits, we know that $\mathbb{E}\left[K_l(\boldsymbol{x}_1)K_l^\intercal(\boldsymbol{x}_+)\right] = \mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\bar{K}_l^\intercal(\boldsymbol{x})\right]$ since $\boldsymbol{x}_1, \boldsymbol{x}_+ \sim p_{\mathrm{aug}}(\cdot|\boldsymbol{x})$ are all augmented data points from a common sample $\boldsymbol{x}$. On the other hand, $\mathbb{E}\left[K_l(\boldsymbol{x}_1)K_l^\intercal(\boldsymbol{x}_{k-})\right] = \mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l^\intercal(\boldsymbol{x})\right]$ since $\boldsymbol{x}_1$ and $\boldsymbol{x}_{k-}$ are generated from independent samples and data augmentation. Therefore,

$$\begin{aligned}\mathrm{vec}(g_{W_l}) &= -\beta\Big\{\mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\bar{K}_l^\intercal(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l^\intercal(\boldsymbol{x})\right]\Big\}\mathrm{vec}(W_l) &(46)\\ &= -\beta\mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\mathrm{vec}(W_l) &(47)\end{aligned}$$

□

D Proofs: The dynamics of two-layer ReLU network and the interplays of covariance operators between nearby layers (Section A.2)

D.1 Theorem 5

Proof. For convenience, we define the centralized version of $\boldsymbol{u}_j(z_0)$: $\hat{\boldsymbol{u}}_j(z_0) = \boldsymbol{u}_j(z_0) - \mathbb{E}_{z_0}\left[\boldsymbol{u}_j(z_0)\right]$ and the matrices $A_{jk} := \mathrm{Cov}_{z_0}[\boldsymbol{u}_j(z_0), \boldsymbol{u}_k(z_0)] = \mathbb{E}_{z_0}\left[\hat{\boldsymbol{u}}_j(z_0)\hat{\boldsymbol{u}}_k^\intercal(z_0)\right]$.

At layer $l = 1$ the covariance operator is $\mathbb{V}_{z_0}[\bar{K}_1(z_0)] = [w_{2,j}w_{2,k}A_{jk}] \in \mathbb{R}^{n_1 d \times n_1 d}$.

On the other hand, if we check the second layer $l = 2$, we could compute $\bar{K}_2(z_0) \in \mathbb{R}^{n_1}$:

$$\bar{K}_2(z_0) = \mathbb{E}_{z'|z_0}\left[\boldsymbol{f}_1|z_0\right] = \mathbb{E}_{z'|z_0}\left[D_1 W_1 \boldsymbol{x}|z_0\right] = \mathbb{E}_{z'|z_0}\left[\boldsymbol{x}^\intercal \otimes D_1(\boldsymbol{x})|z_0\right]\mathrm{vec}(W_1) \tag{48}$$

For each component j, we have $[\bar{K}_2(z_0)]_j = \boldsymbol{w}_{1,j}^\intercal \boldsymbol{u}_j(z_0)$, or:

$$\bar{K}_2(z_0) = [\boldsymbol{w}_{1,1}^\intercal \boldsymbol{u}_1(z_0), \boldsymbol{w}_{1,2}^\intercal \boldsymbol{u}_2(z_0), \ldots, \boldsymbol{w}_{1,n_1}^\intercal \boldsymbol{u}_{n_1}(z_0)] \tag{49}$$

So at layer $l = 2$ we can compute the covariance operator $\mathbb{V}_{z_0}[\bar{K}_2(z_0)] = [\boldsymbol{w}_{1,j}^\intercal A_{jk} \boldsymbol{w}_{1,k}] \in \mathbb{R}^{n_1 \times n_1}$.

Using the assumption that $A_{jk} := \mathbb{E}_{z_0}[\hat{\boldsymbol{u}}_j(z_0)\hat{\boldsymbol{u}}_k^\intercal(z_0)] = \mathbb{E}_{z_0}\left[\hat{\boldsymbol{u}}_j(z_0)\right]\mathbb{E}_{z_0}\left[\hat{\boldsymbol{u}}_k^\intercal(z_0)\right] = 0$ for $j \neq k$. We arrive at the conclusion that $\boldsymbol{w}_j$ is only coupled with v_j, yielding the following dynamical system:

$$\dot{w}_{2,j} = w_{2,j}\boldsymbol{w}_{1,j}^\intercal A_{jj}\boldsymbol{w}_{1,j}, \qquad \dot{\boldsymbol{w}}_{1,j} = w_{2,j}^2 A_{jj}\boldsymbol{w}_{1,j} \tag{50}$$

□

E Proofs: Hierarchical Latent Tree Models (Section A.3)

E.1 Lemmas

Lemma 3 (Variance Squashing)**.** *Suppose a function $\phi : \mathbb{R} \mapsto \mathbb{R}$ is L-Lipschitz continuous: $|\phi(x) - \phi(y)| \leq L|x - y|$, then for $x \sim p(\cdot)$, we have:*

$$\mathbb{V}_p[\phi(x)] \leq L^2 \mathbb{V}_p[x] \tag{51}$$

Proof. Suppose $x, y \sim p(\cdot)$ are independent samples and $\mu_\phi := \mathbb{E}\left[\phi(x)\right]$. Note that $\mathbb{V}[\phi(x)]$ can be written as the following:

$$\begin{aligned}\mathbb{E}\left[|\phi(x) - \phi(y)|^2\right] &= \frac{1}{2}\mathbb{E}\left[|(\phi(x) - \mu_\phi) - (\phi(y) - \mu_\phi)|^2\right]\\ &= \mathbb{E}\left[|\phi(x) - \mu_\phi|^2\right] + \mathbb{E}\left[|\phi(y) - \mu_\phi|^2\right] - 2\mathbb{E}\left[(\phi(x) - \mu_\phi)(\phi(y) - \mu_\phi)\right]\\ &= 2\mathbb{V}_p[\phi(x)] &(52)\end{aligned}$$

Symbol	Definition	Size	Description
$\mathcal{N}_l$, $\mathcal{Z}_l$			The set of all nodes and all latent variables at layer l.
$\mathcal{N}_\mu, \mathcal{N}_\mu^{\text{ch}}$			Nodes corresponding to latent variable z_μ. $\mathcal{N}_\mu^{\text{ch}}$ are children under $\mathcal{N}_\mu$.
m_μ			Number of possible categorical values taken by z_μ. $0 \le z_\mu < m_\mu$.
$\mathbf{0}_\mu$, $\mathbf{1}_\mu$		m_μ	All-one and all-zero vectors.
$P_{\mu\nu}$	$[\mathbb{P}(z_\nu \vert z_\mu)]$	$m_\mu \times m_\nu$	The top-down transition probability from z_μ to z_ν.
$\rho_{\mu\nu}$	$2\mathbb{P}(z_\nu{=}1 \vert z_\mu{=}1) - 1$	scalar in $[-1, 1]$	Polarity of the transitional probability in the binary case.
P_0	$\text{diag}[\mathbb{P}(z_0)]$	$m_0 \times m_0$	The diagonal matrix of probability of z_0 taking different values.
$v_j(z_\mu)$	$\mathbb{E}_z[f_j \vert z_\mu]$	scalar	Expectation of activation f_j given z_μ (z_μ's descendants are marginalized).
$\boldsymbol{v}_j$	$[v_j(z_\mu)]$	m_μ	Vector form of $v_j(z_\mu)$.
$\boldsymbol{f}_\mu$, $\boldsymbol{f}_{\mathcal{N}_\mu^{\text{ch}}}$	$[f_j]_{j\in\mathcal{N}_\mu}$, $[f_k]_{k\in\mathcal{N}_\mu^{\text{ch}}}$	$\vert\mathcal{N}_\mu\vert$, $\vert\mathcal{N}_\mu^{\text{ch}}\vert$	Activations for all nodes $j \in \mathcal{N}_\mu$ and for the children of $\mathcal{N}_\mu$
$\boldsymbol{v}_{0k}$, $V_{0,\mathcal{N}_\mu^{\text{ch}}}$	$[\mathbb{E}_z[f_k \vert z_0]]$, $[\boldsymbol{v}_{0k}]_{k\in\mathcal{N}_\mu^{\text{ch}}}$	m_0, $m_0 \times \vert\mathcal{N}_\mu^{\text{ch}}\vert$	Expected activation conditioned on z_0
s_k	$\frac{1}{2}(v_k(1) - v_k(0))$	scalar	Discrepancy of node k w.r.t its latent variable $z_{\nu(k)}$.
$\boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}}$	$[\rho_{\mu\nu(k)} s_k]_{k\in\mathcal{N}_\mu^{\text{ch}}}$	$\vert\mathcal{N}_\mu^{\text{ch}}\vert$	Child selectivity vector in the binary case.

Table 2: Extended notation in HLTM.

Therefore we have:

$$\mathbb{V}_p[\phi(x)] = \frac{1}{2}\mathbb{E}\left[|\phi(x) - \phi(y)|^2\right] \le \frac{L^2}{2}\mathbb{E}\left[|x - y|^2\right] = L^2\mathbb{V}_p[x] \tag{53}$$

□

Lemma 4 (Sharpened Jensen's inequality [24])**.** *If function ϕ is twice differentiable, and $x \sim p(\cdot)$, then we have:*

$$\frac{1}{2}\mathbb{V}[x]\inf\phi'' \le \mathbb{E}[\phi(x)] - \phi(\mathbb{E}[x]) \le \frac{1}{2}\mathbb{V}[x]\sup\phi'' \tag{54}$$

Lemma 5 (Sharpened Jensen's inequality for ReLU activation)**.** *For ReLU activation $\psi(x) := \max(x, 0)$ and $x \sim p(\cdot)$, we have:*

$$0 \le \mathbb{E}[\psi(x)] - \psi(\mathbb{E}[x]) \le \sqrt{\mathbb{V}_p[x]} \tag{55}$$

Proof. Since ψ is a convex function, by Jensen's inequality we have $\mathbb{E}[\psi(x)] - \psi(\mathbb{E}[x]) \ge 0$. For the other side, let $\mu := \mathbb{E}_p[x]$ and we have (note that for ReLU, $\psi(x) - \psi(\mu) \le |x - \mu|$):

$$\begin{aligned}
\mathbb{E}[\psi(x)] - \psi(\mathbb{E}[x]) &= \int (\psi(x) - \psi(\mu))p(x)\mathrm{d}x && (56)\\
&\le \int |x - \mu|p(x)\mathrm{d}x && (57)\\
&\le \left(\int |x - \mu|^2 p(x)\mathrm{d}x\right)^{1/2}\left(\int p(x)\mathrm{d}x\right)^{1/2} && (58)\\
&= \sqrt{\mathbb{V}_p[x]} && (59)
\end{aligned}$$

where the last inequality is due to Cauchy-Schwarz. □

E.2 More general assumption of conditional distribution in HLTM

We consider a more general case of HLTM where each z_μ is categorical: $0 \le z_\mu < m_\mu$. For convenience, we define the following symbols for $k \in \mathcal{N}_\mu^{\text{ch}}$ (note that $|\mathcal{N}_\mu^{\text{ch}}| = \mathcal{N}_\mu^{\text{ch}}$ is the number of the children of the node set $\mathcal{N}_\mu$):

$$\begin{aligned}
\boldsymbol{v}_{\mu k} &:= \mathbb{E}_z[f_k \vert z_\mu] = P_{\mu\nu(k)}\boldsymbol{v}_k \in \mathbb{R}^{m_\mu} && (60)\\
V_{\mu,\mathcal{N}_\mu^{\text{ch}}} &:= [\boldsymbol{v}_{\mu k}]_{k\in\mathcal{N}_\mu^{\text{ch}}} && (61)\\
\tilde{\boldsymbol{v}}_j &:= \left[\mathbb{E}_z\left[\tilde{f}_j \vert z_\mu\right]\right] = V_{\mu,\mathcal{N}_\mu^{\text{ch}}}\boldsymbol{w}_j \in \mathbb{R}^{m_\mu} && (62)
\end{aligned}$$

As an extension of binary symmetric HLTM, we make an assumption for the transitional probability:

Assumption 1. *For $\mu \in \mathcal{Z}_l$ and $\nu \in \mathcal{Z}_{l-1}$, the transitional probability matrix $P_{\mu\nu} := [\mathbb{P}(z_\nu|z_\mu)]$ has decomposition $P_{\mu\nu} = \frac{1}{m_\nu}\mathbf{1}_\mu\mathbf{1}_\nu^\intercal + C_{\mu\nu}$ where $C_{\mu\nu}\mathbf{1}_\nu = \mathbf{0}_\mu$ and $\mathbf{1}_\mu^\intercal C_{\mu\nu} = \mathbf{0}_\nu$.*

Note that $C_{\mu\nu}\mathbf{1} = \mathbf{0}$ is obvious due to the property of conditional probability. The real condition is $\mathbf{1}_\mu^\intercal C_{\mu\nu} = \mathbf{0}_\nu$. If $m_\mu = m_\nu$, then $P_{\mu\nu}$ is a square matrix and Assumption 1 is equivalent to $P_{\mu\nu}$ is double-stochastic. Assumption 1 makes computation of $P_{\mu\nu}$ easy for any z_μ and z_ν.

Lemma 6 (Transition Probability)**.** *If Assumption 1 holds, then for $\mu \in \mathcal{Z}_l$, $\nu \in \mathcal{Z}_{1-1}$ and $\alpha \in \mathcal{Z}_{l-2}$, we have:*

$$P_{\mu\alpha} = P_{\mu\nu}P_{\nu\alpha} = \frac{1}{m_\alpha}\mathbf{1}_\mu\mathbf{1}_\alpha^\intercal + C_{\mu\nu}C_{\nu\alpha} \tag{63}$$

In general, for any $\mu \in \mathcal{N}_{l_1}$ and $\alpha \in \mathcal{N}_{l_2}$ with $l_1 > l_2$, we have:

$$P_{\mu\alpha} = \frac{1}{m_\alpha}\mathbf{1}_\mu\mathbf{1}_\alpha^\intercal + \prod_{\mu,\ldots,\xi,\zeta,\ldots,\alpha} C_{\xi\zeta} \tag{64}$$

Proof. Using Assumption 1, we have

$$\begin{aligned} P_{\mu\alpha} &= P_{\mu\nu}P_{\nu\alpha} && (65)\\ &= \left(\frac{1}{m_\nu}\mathbf{1}_\mu\mathbf{1}_\nu^\intercal + C_{\mu\nu}\right)\left(\frac{1}{m_\alpha}\mathbf{1}_\nu\mathbf{1}_\alpha^\intercal + C_{\nu\alpha}\right) && (66)\end{aligned}$$

since $\mathbf{1}_\nu^\intercal\mathbf{1}_\nu = m_\nu$, $C_{\mu\nu}\mathbf{1}_\nu = \mathbf{0}_\nu$ and $\mathbf{1}_\nu^\intercal C_{\nu\alpha} = \mathbf{0}_\alpha$, the conclusion follows. □

Remark. In the symmetric binary `HLTM` mentioned in the main text, all $C_{\mu\nu}$ can be parameterized as (here $\boldsymbol{q} := [-1, 1]^\intercal$):

$$C_{\mu\nu} = C_{\mu\nu}(\rho_{\mu\nu}) = \frac{1}{2}\begin{bmatrix} \rho_{\mu\nu} & -\rho_{\mu\nu} \\ -\rho_{\mu\nu} & \rho_{\mu\nu} \end{bmatrix} = \frac{1}{2}\rho_{\mu\nu}\boldsymbol{q}\boldsymbol{q}^\intercal \tag{67}$$

This is because $\mathbf{1}_2^\intercal C_{\mu\nu} = \mathbf{0}_2$ and $C_{\mu\nu}\mathbf{1}_2 = \mathbf{0}_2$ provides 4 linear constraints (1 redundant), leaving 1 free parameter, which is the polarity $\rho_{\mu\nu} \in [-1, 1]$ of latent variable z_ν given its parent z_μ. Moreover, since $\boldsymbol{q}^\intercal\boldsymbol{q} = 2$, the parameterization is close under multiplication:

$$C(\rho_{\mu\nu})C(\rho_{\nu\alpha}) = \frac{1}{4}\boldsymbol{q}\boldsymbol{q}^\intercal\boldsymbol{q}\boldsymbol{q}^\intercal\rho_{\mu\nu}\rho_{\nu\alpha} = \frac{1}{2}\boldsymbol{q}\boldsymbol{q}^\intercal\rho_{\mu\nu}\rho_{\nu\alpha} = C(\rho_{\mu\nu}\rho_{\nu\alpha}) \tag{68}$$

E.3 Theorem 6

Proof. First note that for each node $k \in \mathcal{N}_\mu^{\text{ch}}$:

$$\begin{aligned} \boldsymbol{v}_{0k} &:= [\mathbb{E}_z[f_k|z_0]] = \sum_{z_\nu}\mathbb{E}_z[f_k|z_\nu]\,\mathbb{P}(z_\nu|z_0) = P_{0\nu}\boldsymbol{v}_k && (69)\\ &= \left(\frac{1}{m_\nu}\mathbf{1}_0\mathbf{1}_\nu^\intercal + C_{0\nu}\right)\boldsymbol{v}_k && (70)\\ &= \frac{1}{m_\nu}\mathbf{1}_0\mathbf{1}_\nu^\intercal\boldsymbol{v}_k + C_{0\nu}\boldsymbol{v}_k && (71)\end{aligned}$$

Note that $\mathbf{1}_0\mathbf{1}_\nu^\intercal\boldsymbol{v}_k$ is a constant regarding to change of z_0. So we could remove it when computing covariance operator. On the other hand, for a categorical distribution

$$(\mathbb{P}(z_0 = 0), u(0)), \quad (\mathbb{P}(z_0 = 1), u(1)), \quad \ldots, \quad (\mathbb{P}(z_0 = m_0 - 1), u(m_0 - 1))$$

With $P_0 := \text{diag}[\mathbb{P}(z_0)]$, the mean is $\mathbb{E}_{z_0}[u] = \mathbf{1}^\intercal P_0\boldsymbol{u}$ and its covariance can be written as (here $\mathbf{1} = \mathbf{1}_0$):

$$\mathbb{V}_{z_0}[u] = (\boldsymbol{u} - \mathbf{1}\mathbf{1}^\intercal P_0\boldsymbol{u})^\intercal P_0(\boldsymbol{u} - \mathbf{1}\mathbf{1}^\intercal P_0\boldsymbol{u}) = \boldsymbol{u}^\intercal(P_0 - P_0\mathbf{1}\mathbf{1}^\intercal P_0)\boldsymbol{u} \tag{72}$$

Note that each column of $V_{0,\mathcal{N}_\mu^{\text{ch}}}$ is $\boldsymbol{v}_{0k}$. Setting $\boldsymbol{u} = \boldsymbol{v}_{0k}$ and we have:

$$\mathbb{V}_{z_0}[\mathbb{E}_z\left[\boldsymbol{f}_{\mathcal{N}_\mu^{\text{ch}}}|z_0\right]] = V_{0,\mathcal{N}_\mu^{\text{ch}}}^\intercal(P_0 - P_0^\intercal\mathbf{1}\mathbf{1}^\intercal P_0)V_{0,\mathcal{N}_\mu^{\text{ch}}} \tag{73}$$

Note that $\mathbb{E}_{z_0}[\boldsymbol{v}_{0k}] = \frac{1}{m_\nu}\mathbf{1}_\nu^\intercal\boldsymbol{v}_k + \mathbb{E}_{z_0}[C_{0\nu}\boldsymbol{v}_k]$, since $\mathbf{1}^\intercal P_0\mathbf{1} = 1$. With some computation, we could see $\text{Cov}_{z_0}[\boldsymbol{v}_{0k}, \boldsymbol{v}_{0k'}] = \text{Cov}_{z_0}[C_{0\nu(k)}\boldsymbol{v}_k, C_{0\nu(k')}\boldsymbol{v}_{k'}]$.

The equation above can be applied for any cardinality of latent variables. In the binary symmetric case, we have (note here we define $\rho_0 := \mathbb{P}(z_0 = 1) - \mathbb{P}(z_0 = 0)$, and $\boldsymbol{q} := [-1, 1]^\intercal$):

$$P_0 - P_0 \mathbf{1}\mathbf{1}^\intercal P_0 = \frac{1}{4}(1 - \rho_0^2)\boldsymbol{q}\boldsymbol{q}^\intercal \tag{74}$$

Note that in the binary symmetric case, according to remarks in Lemma 6, all $C_{\mu\nu} = \frac{1}{2}\rho_{\mu\nu}\boldsymbol{q}\boldsymbol{q}^\intercal$ and we could compute $C_{0\nu}\boldsymbol{v}_k$:

$$C_{0\nu}\boldsymbol{v}_k = \frac{1}{2}\rho_{0\nu}\boldsymbol{q}\boldsymbol{q}^\intercal\boldsymbol{v}_k = \rho_{0\nu}\frac{1}{2}(v_k(1) - v_k(0))\boldsymbol{q} = \rho_{0\nu}s_k\boldsymbol{q} \tag{75}$$

where according to Eqn. 68, we have:

$$\rho_{0\nu} := \prod_{0,\ldots,\alpha,\beta,\ldots,\nu} \rho_{\alpha\beta} \tag{76}$$

and the covariance between node k and k' can be computed as:

$$\begin{aligned}
\mathrm{Cov}_{z_0}[\boldsymbol{v}_{0k}, \boldsymbol{v}_{0k'}] &= \mathrm{Cov}_{z_0}[C_{0\nu(k)}\boldsymbol{v}_k, C_{0\nu(k')}\boldsymbol{v}_{k'}] && (77)\\
&= \rho_{0\nu(k)}\rho_{0\nu(k')}s_k s_{k'}\frac{1}{4}\boldsymbol{q}^\intercal\boldsymbol{q}\boldsymbol{q}^\intercal\boldsymbol{q}(1 - \rho_0^2) && (78)\\
&= \rho_{0\nu(k)}\rho_{0\nu(k')}s_k s_{k'}(1 - \rho_0^2) && (79)\\
&= \rho_{0\mu}^2\rho_{\mu\nu(k)}\rho_{\mu\nu(k')}s_k s_{k'}(1 - \rho_0^2) && (80)
\end{aligned}$$

The last equality is due to the fact that due to tree structure, the path from z_0 to all child nodes in $\mathcal{N}_\mu^{\mathrm{ch}}$ must pass z_μ.

Therefore we can compute the covariance operator:

$$\mathbb{V}_{z_0}[\mathbb{E}_z\left[\boldsymbol{f}_{\mathcal{N}_\mu^{\mathrm{ch}}}|z_0\right]] = \rho_{0\mu}^2(1 - \rho_0^2)\boldsymbol{a}_{\mathcal{N}_\mu^{\mathrm{ch}}}\boldsymbol{a}_{\mathcal{N}_\mu^{\mathrm{ch}}}^\intercal \tag{81}$$

When $L \to +\infty$, we have:

$$\rho_{0\nu} := \prod_{0,\ldots,\alpha,\beta,\ldots,\nu} \rho_{\alpha\beta} \to 0 \tag{82}$$

and thus the covariance becomes zero as well. □

E.4 Theorem 7

Proof. According to our setting, for each node $k \in \mathcal{N}_\mu$, there exists a unique latent variable z_ν with $\nu = \nu(k)$ that corresponds to it. In the following we omit its dependency on k for brevity.

Since we are dealing with binary case, we define the following for convenience:

$$\begin{aligned}
v_k^+ &:= v_k(1) && (83)\\
v_k^- &:= v_k(0) && (84)\\
\bar{v}_k &:= \frac{1}{2}(v_k^+ + v_k^-) = \frac{1}{2}\left(\mathbb{E}_z\left[f_k|z_\nu = 1\right] + \mathbb{E}_z\left[f_k|z_\nu = 0\right]\right) && (85)\\
\bar{\boldsymbol{v}}_{\mathcal{N}_\mu^{\mathrm{ch}}} &:= [\bar{v}_k]_{k\in\mathcal{N}_\mu^{\mathrm{ch}}} \in \mathbb{R}^{|\mathcal{N}_\mu^{\mathrm{ch}}|} && (86)\\
s_k &:= \frac{1}{2}(v_k^+ - v_k^-) = \frac{1}{2}\left(\mathbb{E}_z\left[f_k|z_\nu = 1\right] - \mathbb{E}_z\left[f_k|z_\nu = 0\right]\right) && (87)\\
\boldsymbol{s}_{\mathcal{N}_\mu^{\mathrm{ch}}} &:= [s_k]_{k\in\mathcal{N}_\mu^{\mathrm{ch}}} \in \mathbb{R}^{|\mathcal{N}_\mu^{\mathrm{ch}}|} && (88)
\end{aligned}$$

We also define the *sensitivity* of node k to be $\lambda_k := |(v_k^+)^2 - (v_k^-)^2|$. Intuitively, a large λ_k means that the node k is sensitive for changes of latent variable z_ν. If $\lambda_k = 0$, then the node k is invariant to latent variable z_ν.

We first consider pre-activation $\tilde{f}_j := \sum_k w_{jk}f_k$ and its expectation with respect to latent variable z:

$$\tilde{v}_j^+ := \mathbb{E}_z\left[\tilde{f}_j\middle|z_\mu = 1\right], \quad \tilde{v}_j^- := \mathbb{E}_z\left[\tilde{f}_j\middle|z_\mu = 0\right] \tag{89}$$

Note that for each node $k \in \mathcal{N}_\mu^{\text{ch}}$ we have:

$$v_{\mu k}^+ = \bar{v}_k + \rho_{\mu\nu} s_k, \quad v_{\mu k}^- = \bar{v}_k - \rho_{\mu\nu} s_k \tag{90}$$

Let $\boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}} := [a_k]_{k \in \mathcal{N}_\mu^{\text{ch}}} := [\rho_{\mu\nu} s_k]_{k \in \mathcal{N}_\mu^{\text{ch}}}$ and

$$\boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^+ := V_{\mu,\mathcal{N}_\mu^{\text{ch}}}^+ := \Big[\mathbb{E}\left[f_k | z_\mu = 1\right]\Big] = \bar{\boldsymbol{v}}_{\mathcal{N}_\mu^{\text{ch}}} + \boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}} \tag{91}$$

$$\boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^- := V_{\mu,\mathcal{N}_\mu^{\text{ch}}}^- := \Big[\mathbb{E}\left[f_k | z_\mu = 0\right]\Big] = \bar{\boldsymbol{v}}_{\mathcal{N}_\mu^{\text{ch}}} - \boldsymbol{a}_{\mathcal{N}_\mu^{\text{ch}}} \tag{92}$$

Then we have $\tilde{v}_j^+ = \boldsymbol{w}_j^\intercal \boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^+$ and $\tilde{v}_j^- = \boldsymbol{w}_j^\intercal \boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^-$.

Note that $\boldsymbol{w}_j$ is a random variable with each entry $w_{jk} \sim \text{Uniform}\left[-\sigma_w\sqrt{\frac{3}{|\mathcal{N}_\mu^{\text{ch}}|}}, \sigma_w\sqrt{\frac{3}{|\mathcal{N}_\mu^{\text{ch}}|}}\right]$. It is easy to verify that $\mathbb{E}[w_{jk}] = 0$ and $\mathbb{V}[w_{jk}] = \sigma_w^2/|\mathcal{N}_\mu^{\text{ch}}|$. Therefore, for two dimensional vector $\tilde{\boldsymbol{v}}_j = [\tilde{v}_j^+, \tilde{v}_j^-]^\intercal$, we can compute its first and second order moments: $\mathbb{E}_w[\tilde{\boldsymbol{v}}_j] = \boldsymbol{0}$ and $\mathbb{V}_w[\tilde{\boldsymbol{v}}_j] = \frac{\sigma_w^2}{|\mathcal{N}_\mu^{\text{ch}}|} V_{\mu,\mathcal{N}_\mu^{\text{ch}}} V_{\mu,\mathcal{N}_\mu^{\text{ch}}}^\intercal = \frac{\sigma_w^2}{|\mathcal{N}_\mu^{\text{ch}}|}[\boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^+, \boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^-]^\intercal [\boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^+, \boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^-]$.

Define the positive and negative set (note that $a_k := \rho_{\mu\nu} s_k$):

$$A_+ = \{k : a_k \geq 0\}, \quad A_- = \{k : a_k < 0\} \tag{93}$$

Without loss of generality, assume that $\sum_{k \in A_+} a_k^2 \geq \sum_{k \in A_-} a_k^2$. In the following, we show there exists j with λ_j is greater than some positive threshold. Otherwise the proof is symmetric and we can show λ_j is lower than some negative threshold.

When $|\mathcal{N}_\mu^{\text{ch}}|$ is large, by Central Limit Theorem, $\boldsymbol{v}$ can be regarded as zero-mean 2D Gaussian distribution and we have for some $c > 0$:

$$\mathbb{P}\left(\tilde{v}_j^+ \geq \frac{\sqrt{c}\sigma_w}{\sqrt{|\mathcal{N}_\mu^{\text{ch}}|}} \|\boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^+\|\right) = \frac{1 - \text{erf}(\sqrt{c}/2)}{2} \tag{94}$$

Moreover, if $\boldsymbol{a}_l \neq \boldsymbol{0}$, then the following probability is also not small :

$$\mathbb{P}\left(\tilde{v}_j^+ \geq \frac{\sqrt{c}\sigma_w}{\sqrt{|\mathcal{N}_\mu^{\text{ch}}|}} \|\boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^+\| \quad \text{and} \quad \tilde{v}_j^- < 0\right) \tag{95}$$

Therefore, when $|\mathcal{N}_\mu| = O(\exp(c))$, with high probability, there exists $\boldsymbol{w}_j$ so that

$$\tilde{v}_j^+ = \boldsymbol{w}_j^\intercal \boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^+ \geq \frac{\sqrt{c}\sigma_w}{\sqrt{|\mathcal{N}_\mu^{\text{ch}}|}} \|\boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^+\|, \qquad \tilde{v}_j^- = \boldsymbol{w}_j^\intercal \boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^- < 0 \tag{96}$$

Since $\bar{\boldsymbol{v}}_{\mathcal{N}_\mu^{\text{ch}}} \geq 0$ (all f_k are after ReLU and non-negative), this leads to:

$$\tilde{v}_j^+ \geq \frac{\sqrt{c}\sigma_w}{\sqrt{|\mathcal{N}_\mu^{\text{ch}}|}} \|\boldsymbol{u}_{\mathcal{N}_\mu^{\text{ch}}}^+\| \geq \frac{\sqrt{c}\sigma_w}{\sqrt{|\mathcal{N}_\mu^{\text{ch}}|}} \sqrt{\sum_{k \in A_+} a_k^2} \geq \sigma_w \sqrt{\frac{c}{2|\mathcal{N}_\mu^{\text{ch}}|} \sum_{k \in \mathcal{N}_\mu^{\text{ch}}} \rho_{\mu\nu}^2 s_k^2} \tag{97}$$

By Jensen's inequality, we have (note that $\psi(x) := \max(x, 0)$ is the ReLU activation):

$$v_j^+ = \mathbb{E}_z\left[f_j | z_\mu = 1\right] = \mathbb{E}_z\left[\psi(\tilde{f}_j) | z_\mu = 1\right] \tag{98}$$

$$\geq \psi\left(\mathbb{E}_z\left[\tilde{f}_j \middle| z_\mu = 1\right]\right) = \psi(\tilde{v}_j^+) \geq \sigma_w \sqrt{\frac{c}{2|\mathcal{N}_\mu^{\text{ch}}|} \sum_{k \in \mathcal{N}_\mu^{\text{ch}}} \rho_{\mu\nu}^2 s_k^2} \tag{99}$$

On the other hand, we also want to compute $v_j^- := \mathbb{E}_z\left[f_j|z_\mu = 0\right]$ using sharpened Jensen's inequality (Lemma 5). For this we need to compute the conditional covariance $\mathbb{V}_z[\tilde{f}_j|z_\mu]$:

$$\mathbb{V}_z[\tilde{f}_j|z_\mu] \overset{②}{=} \sum_k w_{jk}^2 \mathbb{V}_z[f_k|z_\mu] \overset{③}{\leq} \frac{3\sigma_w^2}{|\mathcal{N}_\mu^{\text{ch}}|}\sum_k \mathbb{V}_z[f_k|z_\mu] \tag{100}$$

$$= \frac{3\sigma_w^2}{|\mathcal{N}_\mu^{\text{ch}}|}\sum_k \left(\mathbb{E}_{z_\nu|z_\mu}\left[\mathbb{V}[f_k|z_\nu]\right] + \mathbb{V}_{z_\nu|z_\mu}[\mathbb{E}_z\left[f_k|z_\nu\right]]\right) \tag{101}$$

$$\leq 3\sigma_w^2\left(\sigma_l^2 + \frac{1}{|\mathcal{N}_\mu^{\text{ch}}|}\sum_k \mathbb{V}_{z_\nu|z_\mu}[\mathbb{E}_z\left[f_k|z_\nu\right]]\right) \tag{102}$$

Note that ② is due to conditional independence: f_k as the computed activation, only depends on latent variable z_ν and its descendants. Given z_μ, all z_ν and their respective descendants are independent of each other and so does f_k. ③ is due to the fact that each w_{jk} are sampled from uniform distribution and $|w_{jk}| \leq \sigma_w\sqrt{\frac{3}{|\mathcal{N}_\mu^{\text{ch}}|}}$.

Here $\mathbb{V}_{z_\nu|z_\mu}[\mathbb{E}_z\left[f_k|z_\nu\right]] = s_k^2(1-\rho_{\mu\nu}^2)$ can be computed analytically. It is the variance of a binomial distribution: with probability $\frac{1}{2}(1+\rho_{\mu\nu})$ we get v_k^+ otherwise get v_k^-. Therefore, we finally have:

$$\mathbb{V}_z[\tilde{f}_j|z_\mu] \leq 3\sigma_w^2\left(\sigma_l^2 + \frac{1}{|\mathcal{N}_\mu^{\text{ch}}|}\sum_k s_k^2(1-\rho_{\mu\nu}^2)\right) \tag{103}$$

As a side note, using Lemma 3, since ReLU function ψ has Lipschitz constant ≤ 1 (empirically it is smaller), we know that:

$$\mathbb{V}_z[f_j|z_\mu] \leq 3\sigma_w^2\left(\sigma_l^2 + \frac{1}{|\mathcal{N}_\mu^{\text{ch}}|}\sum_k s_k^2(1-\rho_{\mu\nu}^2)\right) \tag{104}$$

Finally using Lemma 5 and $\tilde{v}_j^- < 0$, we have:

$$v_j^- = \mathbb{E}_z\left[f_j|z_\mu = 0\right] = \mathbb{E}_z\left[\psi(\tilde{f}_j)|z_\mu = 0\right] \tag{105}$$

$$\leq \psi\left(\mathbb{E}_z\left[\tilde{f}_j\middle|z_\mu = 0\right]\right) + \sqrt{\mathbb{V}_z[\tilde{f}_j|z_\mu = 0]} \tag{106}$$

$$= \sqrt{\mathbb{V}_z[\tilde{f}_j|z_\mu = 0]} \tag{107}$$

$$\leq \sigma_w\sqrt{3\sigma_l^2 + \frac{3}{|\mathcal{N}_\mu^{\text{ch}}|}\sum_k s_k^2(1-\rho_{\mu\nu}^2)} \tag{108}$$

Combining Eqn. 99 and Eqn. 108, we have a bound for λ_j:

$$\lambda_j = (v_j^+)^2 - (v_j^-)^2 \geq 3\sigma_w^2\left[\frac{1}{|\mathcal{N}_\mu^{\text{ch}}|}\sum_k s_k^2\left(\frac{c+6}{6}\rho_{\mu\nu}^2 - 1\right) - \sigma_l^2\right] \tag{109}$$

□

F Proofs: The Analysis of BYOL 4

F.1 Derivation of BYOL gradient

Note that for BYOL, we have:

$$\text{vec}\left(\frac{\partial r}{\partial W_l}\right) = K_l(\boldsymbol{x}_1;\mathcal{W})\left[K_l^\intercal(\boldsymbol{x}_1;\mathcal{W})\text{vec}(W_l) - K_l^\intercal(\boldsymbol{x}_2;\mathcal{W}')\text{vec}(W_l')\right] \tag{110}$$

under large batchsize, we have (note that we omit $\mathcal{W}$ for any term that depends on $\mathcal{W}$, but make dependence of $\mathcal{W}'$ explicit in the math expression):

$$\text{vec}\left(\frac{\partial r}{\partial W_l}\right) = \mathbb{E}_{\boldsymbol{x}\sim p(\cdot)}\left[\mathbb{E}_{\boldsymbol{x}'\sim p_{\text{aug}}(\cdot|\boldsymbol{x})}\left[K_l(\boldsymbol{x}')K_l^\intercal(\boldsymbol{x}')\right]\text{vec}(W_l) - \bar{K}_l(\boldsymbol{x})\bar{K}_l^\intercal(\boldsymbol{x};\mathcal{W}')\text{vec}(W_l')\right]$$

For brevity, we write $\mathbb{E}_{\boldsymbol{x}}\left[\cdot\right] := \mathbb{E}_{\boldsymbol{x}\sim p(\cdot)}\left[\cdot\right]$ and $\mathbb{E}_{\boldsymbol{x}'}\left[\cdot\right] := \mathbb{E}_{\boldsymbol{x}'\sim p_{\text{aug}}(\cdot|\boldsymbol{x})}\left[\cdot\right]$. Similar for $\mathbb{V}$. And the equation above can be written as:

$$\begin{aligned}\text{vec}\left(\frac{\partial r}{\partial W_l}\right) &= \mathbb{E}_{\boldsymbol{x}}\left\{\mathbb{V}_{\boldsymbol{x}'}[K_l(\boldsymbol{x}')]\right\}\text{vec}(W_l) &(111)\\ &+ \mathbb{E}_{\boldsymbol{x}}\left\{\bar{K}_l(\boldsymbol{x})\left[\bar{K}_l^{\intercal}(\boldsymbol{x})\text{vec}(W_l) - \bar{K}_l^{\intercal}(\boldsymbol{x};\mathcal{W}')\text{vec}(W_l')\right]\right\} &(112)\end{aligned}$$

In terms of weight update by gradient descent, since $\Delta W_l = -\frac{\partial r}{\partial W_l}$, we have:

$$\begin{aligned}\text{vec}\left(\Delta W_l\right) &= -\mathbb{E}_{\boldsymbol{x}}\left\{\mathbb{V}_{\boldsymbol{x}'}[K_l(\boldsymbol{x}')]\right\}\text{vec}(W_l) &(113)\\ &- \mathbb{E}_{\boldsymbol{x}}\left\{\bar{K}_l(\boldsymbol{x})\left[\bar{K}_l^{\intercal}(\boldsymbol{x})\text{vec}(W_l) - \bar{K}_l^{\intercal}(\boldsymbol{x};\mathcal{W}')\text{vec}(W_l')\right]\right\} &(114)\end{aligned}$$

If we consider the special case $\mathcal{W} = \mathcal{W}'$, then the last two terms cancelled out, yielding:

$$\text{vec}(\Delta W_l)_{\text{sym}} = -\mathbb{E}_{\boldsymbol{x}}\left\{\mathbb{V}_{\boldsymbol{x}'}[K_l(\boldsymbol{x}')]\right\}\text{vec}(W_l) \tag{115}$$

And the general update (Eqn. 117) can be written as:

$$\begin{aligned}\text{vec}\left(\Delta W_l\right) &= \text{vec}(\Delta W_l)_{\text{sym}} &(116)\\ &- \mathbb{E}_{\boldsymbol{x}}\left\{\bar{K}_l(\boldsymbol{x})\left[\bar{K}_l^{\intercal}(\boldsymbol{x})\text{vec}(W_l) - \bar{K}_l^{\intercal}(\boldsymbol{x};\mathcal{W}')\text{vec}(W_l')\right]\right\} &(117)\end{aligned}$$

F.2 Theorem 4

Proof. When BN is present, Eqn. 110 needs to be corrected with an additional term, $\widetilde{\frac{\partial r}{\partial W_l}} := \frac{\partial r}{\partial W_l} - \delta W_l^{\text{BN}}$, where δW_l^{BN} is defined as follows:

$$\delta W_l^{\text{BN}} := \frac{1}{|B|}\sum_{i\in B} D_l^i \bar{\boldsymbol{g}}_l \boldsymbol{f}_{l-1}^{i\intercal} \tag{118}$$

From the proof of Theorem 1, we know that for each sample $i \in B$:

$$D_l^i \boldsymbol{g}_l^i = J_l^{i\intercal}[J_l^i W_l \boldsymbol{f}_{l-1}^i - J_l^i(\mathcal{W}')W_l'\boldsymbol{f}_{l-1}^i(\mathcal{W}')] \tag{119}$$

Since the network is linear from layer l to the topmost layer L, we have $D_l^i = \bar{D}_l$. Since the only input dependent part in J_l^i is the gating function between the current layer l and the topmost layer L, for linear network the gating is always 1 and thus $\bar{J}_l = J_l^i$ and is independent of input data. We now have (note that we omit $\mathcal{W}$ for any terms that are dependent on $\mathcal{W}$, but will write $\mathcal{W}'$ explicitly for terms that are depend on $\mathcal{W}'$):

$$\begin{aligned}\delta W_l^{\text{BN}} &:= \frac{1}{|B|}\sum_{i\in B} D_l^i \bar{\boldsymbol{g}}_l \boldsymbol{f}_{l-1}^{i\intercal} = -\bar{D}_l \bar{\boldsymbol{g}}_l \bar{\boldsymbol{f}}_{l-1}^{\intercal} &(120)\\ &= \bar{J}_l^{\intercal}[\bar{J}_l W_l \bar{\boldsymbol{f}}_{l-1} - \bar{J}_l(\mathcal{W}')W_l'\bar{\boldsymbol{f}}_{l-1}(\mathcal{W}')]\bar{\boldsymbol{f}}_{l-1}^{\intercal} &(121)\end{aligned}$$

Therefore we have:

$$\text{vec}(\delta W_l^{\text{BN}}) = (\bar{\boldsymbol{f}}_{l-1}\otimes\bar{J}_l^{\intercal})\left[(\bar{\boldsymbol{f}}_{l-1}\otimes\bar{J}_l^{\intercal})\text{vec}(W_l) - (\bar{\boldsymbol{f}}_{l-1}(\mathcal{W}')\otimes\bar{J}_l^{\intercal}(\mathcal{W}'))\text{vec}(W_l')\right] \tag{122}$$

Note that by assumption, since $\bar{J}_l$ doesn't depend on the input data, we have

$$\bar{\boldsymbol{f}}_{l-1}\otimes\bar{J}_l^{\intercal} = \mathbb{E}_B\left[\boldsymbol{f}_{l-1}\right]\otimes\bar{J}_l^{\intercal} = \mathbb{E}_B\left[\boldsymbol{f}_{l-1}\otimes\bar{J}_l^{\intercal}\right] \tag{123}$$

Taking large batchsize limits and notice that the batch B could contain any augmented data generated from independent samples from $p(\cdot)$, we have:

$$\begin{aligned}\text{vec}(\delta W_l^{\text{BN}}) &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}\left[K_l(\boldsymbol{x}')\right]\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}\left[K_l^{\intercal}(\boldsymbol{x}')\right]\text{vec}(W_l) &(124)\\ &- \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}\left[K_l(\boldsymbol{x}')\right]\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}\left[K_l^{\intercal}(\boldsymbol{x}';\mathcal{W}')\right]\text{vec}(W_l') &(125)\end{aligned}$$

An important thing is that the expectation is taking over $\boldsymbol{x}\sim p(\boldsymbol{x})$ and $\boldsymbol{x}'\sim p_{\text{aug}}(\cdot|\boldsymbol{x})$. Intuitively, this is because $\bar{\boldsymbol{f}}_{l-1}$ and $\bar{\boldsymbol{g}}_l$ are averages over the entire batch, which has both intra-sample and inter-sample variation.

With augment-mean connection $\bar{K}_l(\boldsymbol{x})$ we could write:

$$\begin{aligned}\operatorname{vec}(\delta W_l^{\mathrm{BN}}) &= \mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l^{\intercal}(\boldsymbol{x})\right]\operatorname{vec}(W_l) - \mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l^{\intercal}(\boldsymbol{x};\mathcal{W}')\right]\operatorname{vec}(W_l') \\ &= \mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\left\{\mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l^{\intercal}(\boldsymbol{x})\right]\operatorname{vec}(W_l) - \mathbb{E}_{\boldsymbol{x}}\left[\bar{K}_l^{\intercal}(\boldsymbol{x};\mathcal{W}')\right]\operatorname{vec}(W_l')\right\} \quad (126)\end{aligned}$$

Plug in $\delta W_{l,BN}$ into Eqn. 117 and we have corrected gradient for BYOL:

$$\begin{aligned}\operatorname{vec}\widetilde{\left(\frac{\partial r}{\partial W_l}\right)} &= \operatorname{vec}\left(\frac{\partial r}{\partial W_l}\right) - \operatorname{vec}\left(\delta W_l^{\mathrm{BN}}\right) \quad (127)\\ &= \mathbb{E}_{\boldsymbol{x}}\left[\mathbb{V}_{\boldsymbol{x}'\sim p_{\mathrm{aug}}(\cdot|\boldsymbol{x})}\left[K_l(\boldsymbol{x}')\right]\right]\operatorname{vec}(W_l) + \mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\operatorname{vec}(W_l) \quad (128)\\ &- \operatorname{Cov}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x}), \bar{K}_l(\boldsymbol{x};\mathcal{W}')\right]\operatorname{vec}(W_l') \quad (129)\end{aligned}$$

And the weight update $\widetilde{\Delta W_l} = \Delta W_l + \delta W_l^{\mathrm{BN}}$ is:

$$\begin{aligned}\operatorname{vec}\left(\widetilde{\Delta W_l}\right) &= -\mathbb{E}_{\boldsymbol{x}}\left[\mathbb{V}_{\boldsymbol{x}'\sim p_{\mathrm{aug}}(\cdot|\boldsymbol{x})}\left[K_l(\boldsymbol{x}')\right]\right]\operatorname{vec}(W_l) - \mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\operatorname{vec}(W_l) \quad (130)\\ &+ \operatorname{Cov}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x}), \bar{K}_l(\boldsymbol{x};\mathcal{W}')\right]\operatorname{vec}(W_l') \quad (131)\end{aligned}$$

Using Eqn. 115, we have:

$$\begin{aligned}\operatorname{vec}\left(\widetilde{\Delta W_l}\right) &= \operatorname{vec}(\Delta W_l)_{\mathrm{sym}} \quad (132)\\ &- \mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right]\operatorname{vec}(W_l) + \operatorname{Cov}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x}), \bar{K}_l(\boldsymbol{x};\mathcal{W}')\right]\operatorname{vec}(W_l') \quad (133)\end{aligned}$$

□

F.3 Analysis on EMA

Consider the following discrete dynamics of a weight vector $\boldsymbol{w}(t)$:

$$\boldsymbol{w}(t+1) - \boldsymbol{w}(t) = \alpha\left[-\boldsymbol{w}(t) + (1-\lambda)\boldsymbol{w}_{\mathrm{ema}}(t)\right] \quad (134)$$

where α is the learning rate, $\boldsymbol{w}_{\mathrm{ema}}(t+1) = \gamma_{\mathrm{ema}}\boldsymbol{w}_{\mathrm{ema}}(t) + (1-\gamma_{\mathrm{ema}})\boldsymbol{w}(t)$ is the exponential moving average of $\boldsymbol{w}(t)$. For convenience, we use $\eta := 1 - \gamma_{\mathrm{ema}}$.

Since it is a recurrence equation, we apply *z-transform* on the temporal domain, where $\boldsymbol{w}(z) := \mathbb{Z}[\boldsymbol{w}(t)] = \sum_{t=0}^{+\infty}\boldsymbol{w}(t)z^{-t}$. This leads to:

$$z(\boldsymbol{w}(z) - \boldsymbol{w}(0)) = \boldsymbol{w}(z) - \alpha\left(\boldsymbol{w}(z) - \mathbb{Z}[\boldsymbol{w}_{\mathrm{ema}}(t)](1-\lambda)\right) \quad (135)$$

Note that for $w_{\mathrm{ema}}(t)$ we have:

$$z(\boldsymbol{w}_{\mathrm{ema}}(z) - \boldsymbol{w}_{\mathrm{ema}}(0)) = (1-\eta)\boldsymbol{w}_{\mathrm{ema}}(z) + \eta\boldsymbol{w}(z) \quad (136)$$

If we set $\boldsymbol{w}_{\mathrm{ema}}(0) = 0$, i.e., the target network is all zero at the beginning, then it gives $\boldsymbol{w}_{\mathrm{ema}}(z) = \frac{\eta}{z-1+\eta}\boldsymbol{w}(z)$. Plugging it back to Eqn. 135 and we have:

$$z(\boldsymbol{w}(z) - \boldsymbol{w}(0)) = \boldsymbol{w}(z) - \alpha\boldsymbol{w}(z)\left(1 - \frac{\eta}{z-1+\eta}(1-\lambda)\right) \quad (137)$$

And then we could solve $\boldsymbol{w}(z)$:

$$\boldsymbol{w}(z) = \frac{z(z-1+\eta)}{(z-1)^2 + (\eta+\alpha)(z-1) + \alpha\eta\lambda}\boldsymbol{w}(0) \quad (138)$$

Note that the denominator has two roots z_1 and z_2:

$$z_{1,2} = 1 - \frac{1}{2}\left(\eta + \alpha \pm \sqrt{(\eta+\alpha)^2 - 4\alpha\eta\lambda}\right) \quad (139)$$

and $\boldsymbol{w}(z)$ can be written as

$$\boldsymbol{w}(z) = \frac{z(z-1+\eta)}{(z-z_1)(z-z_2)}\boldsymbol{w}(0) \quad (140)$$

Without loss of generality, let $z_1 < z_2$. The larger root $z_2 > 1$ when $\lambda < 0$, so the zero ($z = 1 - \eta = \gamma_{\text{ema}}$) in the nominator won't cancel out the pole at z_2. And we have:

$$\frac{z}{(z-z_1)(z-z_2)} = \frac{z}{z_2 - z_1}\frac{(z-z_1)-(z-z_2)}{(z-z_1)(z-z_2)} \tag{141}$$

$$= \frac{z}{z_2 - z_1}\left(\frac{1}{z-z_2} - \frac{1}{z-z_1}\right) \tag{142}$$

$$= \frac{1}{z_2 - z_1}\left(\frac{1}{1-z_2 z^{-1}} - \frac{1}{1-z_1 z^{-1}}\right) \tag{143}$$

where $1/(1-z_2 z^{-1})$ corresponds to a power series z_2^t in the temporal domain. Therefore, we could see $\boldsymbol{w}(t)$ has exponential growth due to $z_2 > 1$.

Table 3: Normalized Correlation between the topmost latent variables in binary HLTM and topmost nodes in deep ReLU networks ($L = 5$) go up when training with SimCLR with NCE loss. We see higher correlations at both initialization and end of training, with more over-parameterization (**Left**: $|\mathcal{N}_\mu| = 2$, **Right**: $|\mathcal{N}_\mu| = 5$).

$\rho_{\mu\nu}$	Initial	1 epoch	20 epochs
$\sim$ Uniform$[0.7, 1]$	0.51	0.69	0.76
$\sim$ Uniform$[0.8, 1]$	0.65	0.76	0.79
$\sim$ Uniform$[0.9, 1]$	0.81	0.85	0.86

$\rho_{\mu\nu}$	Initial	1 epoch	20 epochs
$\sim$ Uniform$[0.7, 1]$	0.60	0.72	0.88
$\sim$ Uniform$[0.8, 1]$	0.73	0.80	0.87
$\sim$ Uniform$[0.9, 1]$	0.87	0.90	0.95

Table 4: Top-1 STL performance with different combination of predictor (**P**), EMA and BatchNorm using BYOL. When EMA is on, we use $\gamma_{\text{ema}} = 0.996$. Batch size is 128 and all experiments run on 5 seeds.

-	EMA	BN	EMA, BN	P	P, EMA	P, BN	P, EMA, BN
38.7 ± 0.6	39.3 ± 0.9	33.0 ± 0.3	32.8 ± 0.5	39.5 ± 3.1	44.4 ± 3.2	63.6 ± 1.06	**78.1** ± 0.3

Table 5: Top-1 STL performance using different BatchNorm components in the predictor and the projector of BYOL ($\gamma_{\text{ema}} = 0.996$, 100 epochs). There is no affine part. "μ" = zero-mean normalization only, "μ, σ" = BN without affine, "$\mu, \sigma^{\nexists}$" = normalization with mean and std but only backpropagating through mean. All variants with detached zero-mean normalization (in red) yield similar poor performance as no normalization.

-	μ	σ	μ, σ	$\mu^{\nexists}$	$\sigma^{\nexists}$	$\mu^{\nexists}, \sigma$	$\mu, \sigma^{\nexists}$	$\mu^{\nexists}, \sigma^{\nexists}$
43.9 ± 4.2	64.8 ± 0.6	72.2 ± 0.9	**78.1** ± 0.3	44.2 ± 7.0	54.2 ± 0.6	48.3 ± 2.7	76.3 ± 0.4	47.0 ± 8.1

Table 6: Top-1 performance of BYOL using reinitialization of the predictor every T epochs.

	Original BYOL	ReInit $T = 5$	ReInit $T = 10$	ReInit $T = 20$
STL-10 (100 epochs)	78.1	78.6	**79.1**	79.0
ImageNet (60 epochs)	60.9	61.9	**62.4**	**62.4**

G Experiments

We test our theoretical findings through experiments on STL-10 [10] and ImageNet [11]. We use a simplified linear evaluation protocol: the linear classifier is trained on frozen representations computed *without* data augmentation. This reuses pre-computed representations and accelerates evaluation by 10x.

Hierarchical Latent Tree Model (HLTM). We implement the HLTM and check whether the intermediate layers of deep ReLU networks learn the corresponding latent variables at the same layer. The degree of learning is measured by the normalized correlations between the ground truth latent variable z_μ and its best corresponding node $j \in \mathcal{N}_\mu$ at each layer. Tbl. 2 indicates this measure increases with over-parameterization and learning, consistent with the analysis in Sec. A.3.

Factors underlying BYOL performance. To test our theory, we perform an ablation study of BYOL on STL-10 by modifying three key components: predictor, EMA and BN. Tbl. 4 shows that BN and predictor are critical and EMA further improves the performance. First, without a predictor, neither BN nor EMA give good performance. A predictor without BN still doesn't work. A predictor with BN starts to show good performance (63.6%) and further adding EMA leads to the best performance (78.1%). This is consistent with our theoretical findings in Sec. 4, in which we show that using a predictor with BN yields $\delta W_l^{\text{BN}} \neq 0$ and leads to an implicit contrastive term.

To further test our understanding of the role played by BN, we fractionate BN into several sub-components: subtract by batch mean (`mean-norm`), divide by batch standard deviation (`std-norm`) and `affine`, and do ablation studies (Tbl. 5). Surprisingly, removing `affine` yields slightly better performance on STL-10 (from 78.1% to 78.7%). We also find that variants of `mean-norm` performs reasonably, while variants of *detached* `mean-norm` has similar poor performance as no normalization, supporting that centralizing backpropagated gradient leads to implicit contrastive terms (Sec. 4). Note that `std-norm` also helps, which we leave for future analysis.

We also check whether the online network requires an "optimal predictor" as suggested by recent version (v3) of BYOL. For this, we reinitialize the predictor (`ReInit`) every T epochs and compare the final performance under linear evaluation protocol. Interestingly, as shown in Tbl. 6, `ReInit` actually improves the performance a bit, compared to the original BYOL that keeps training the same predictor, which should therefore be closer to optimal. Moreover, if we shrink the initial weight range of the predictor to make $\mathrm{Cov}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x}), \bar{K}_l(\boldsymbol{x}; \mathcal{W}')\right]$ (third term in Eqn. 7) more dominant, and reduce the learning rate, the performance further improves (See Tbl. 10 in Appendix H), thereby corroborating our analysis.

H Further Experiments on Predictor

Based on our theoretical analysis, we try training the predictor in different ways and check whether it still works.

From the analysis in Sec. 4, we know that the reason why BYOL works is due to the dominance of $\mathrm{Cov}_{\boldsymbol{x}}[\bar{K}_l(\boldsymbol{x}), \bar{K}_l(\boldsymbol{x}; \mathcal{W})]$ and its resemblance of the covariance operator $\mathbb{V}_{\boldsymbol{x}}[\bar{K}_l(\boldsymbol{x})]$, which is a PSD matrix.

The dominance should be stronger if the predictor has smaller weights than normally initialized using Xavier/Kaiming initialization. Also, $\mathrm{Cov}_{\boldsymbol{x}}[\bar{K}_l(\boldsymbol{x}), \bar{K}_l(\boldsymbol{x}; \mathcal{W}')]$ should behave more like a PSD matrix, if the predictor's weights are all small positive numbers and no BN is used.

The following table justifies our theoretical findings. In particular, Tbl. 10 shows better performance in STL-10 with smaller learning rate and smaller sample range of the predictor weights.

Table 7: Training one-layer predictor with positive initial weights and no EMA ($\gamma_{\mathrm{ema}} = 0$). All experiments run for 3 seeds.

Sample range of predictor weight	$[0, 0.01]$	$[0, 0.02]$	$[0, 0.05]$
With BN in predictor	62.78 ± 1.40	62.94 ± 1.03	62.31 ± 1.80
Without BN in predictor	71.95 ± 0.27	72.06 ± 0.44	71.91 ± 0.59

Table 8: Training one-layer predictor with positive initial weights with EMA ($\gamma_{\mathrm{ema}} = 0.996$) and predictor resetting every $T = 10$ epochs. All experiments run for 3 seeds. Note that Xavier range is $\mathrm{Uniform}[-0.15, 0.15]$ and our initialization range is much smaller than that.

Sample range of predictor weight	$[0, 0.003]$	$[0, 0.005]$	$[0, 0.007]$
With BN in predictor	65.61 ± 1.34	70.56 ± 0.57	70.87 ± 1.51
Without BN in predictor	74.39 ± 0.67	74.52 ± 0.63	74.80 ± 0.57

Table 9: Same as Tbl. 8 but with different weight range. All experiments run for 3 seeds.

Sample range of predictor weight	$[0, 0.01]$	$[0, 0.02]$	$[0, 0.05]$
With BN in predictor	68.98 ± 2.34	66.56 ± 1.70	68.41 ± 1.19
Without BN in predictor	74.66 ± 0.81	73.60 ± 0.32	74.34 ± 0.77

I Related Works

In addition to SimCLR and BYOL, many concurrent SSL frameworks exist to learn good representations for computer vision tasks. MoCo [17, 9] keeps a large bank of past representations in a queue as the slow-progressing target to train from. DeepCluster [6] and SwAV [7] learn the representations by iteratively or implicitly clustering on the current representations and improving representations using the cluster label. [2] applies similar ideas to multi-modality tasks. Contrastive Predictive

Table 10: Top-1 Performance on STL-10 with a two-layer predictor with BN and EMA ($\gamma_{\text{ema}} = 0.996$). Learning rate is smaller (0.02) and predictor weight sampled from $\text{Uniform}[-range, range]$. Note that for this, Xavier range is $\text{Uniform}[-0.097, 0.097]$ and our range is smaller.

Weight range	0.01	0.02	0.03	0.05
$T = 3$	79.48 ± 0.40	79.70 ± 0.47	79.66 ± 0.37	78.63 ± 0.10
$T = 5$	78.97 ± 0.62	79.63 ± 0.23	79.65 ± 0.37	79.01 ± 0.27
$T = 10$	79.25 ± 0.20	79.63 ± 0.22	79.58 ± 0.25	79.18 ± 0.22
$T = 20$	79.15 ± 0.66	79.91 ± 0.10	79.78 ± 0.05	79.54 ± 0.25

Coding [26] learns the representation by predicting the future of a sequence in the latent space with autoregressive models and InfoNCE loss. Contrastive MultiView Coding [29] uses multiple sensory channels (called "view") of the same scene as the positive pairs and independently sampled views as the negative pairs to train the model. Recently, [23] moves beyond instance-wise pairs and proposes to use prototypes to construct training pairs that are more semantically meaningful.

In contrast, the literature on the (theoretical) analysis of SSL is sparse. [33] shows directly optimizing the alignment/uniformity of the positive/negative pairs leads to comparable performance against contrastive loss. [3] proposes an interesting analysis of how contrastive learning aids downstream classification tasks, given assumptions about data generation. [22] analyzes how learning pretext tasks could reduce the sample complexity of the downstream task and [32] analyzes contrastive loss with multi-view data in the semi-supervised setting, with different generative models. However, they either work on linear model or treat deep models as a black-box function approximator with sufficient capacity. In comparison, we incorporate self-supervision, deep models, contrastive loss, data augmentation and generative models together into the same theoretical framework, and makes an attempt to understand how the intermediate features are emerged in modern SSL architectures with deep models that achieve SoTA.