BARRIERS FOR LEARNING IN AN EVOLVING WORLD: MATHEMATICAL UNDERSTANDING OF LOSS OF PLASTICITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep learning models excel in stationary data but struggle in non-stationary environments due to a phenomenon known as loss of plasticity (LoP), the degradation of their ability to learn in the future. This work presents a first-principles investigation of LoP in gradient-based learning. Grounded in dynamical systems theory, we formally define LoP by identifying stable manifolds in the parameter space that trap gradient trajectories. Our analysis reveals two primary mechanisms that create these traps: frozen units from activation saturation and cloned-unit manifolds from representational redundancy. Our framework uncovers a fundamental tension: properties that promote generalization in static settings, such as low-rank representations and simplicity biases, directly contribute to LoP in continual learning scenarios. We validate our theoretical analysis with numerical simulations and explore architectural choices or targeted perturbations as potential mitigation strategies.

1 INTRODUCTION

The extraordinary success of back-propagation in training deep neural networks often relies on two implicit assumptions. First, *stationarity* is assumed. This means that the data distribution encountered during training is similar to the distribution faced during deployment. As a result, post-training adaptation is minimal or absent. Second, a *single random initialization* of network parameters is the main source of diversity and exploration potential, a resource that is progressively consumed by optimization and not replenished. These assumptions falter when an artificial agent must operate and learn continuously within an environment characterized by changing dynamics or evolving task distributions. This scenario, commonly referred to as continual or lifelong learning, presents the stability-plasticity dilemma (Abraham and Robins, 2005; Chaudhry et al., 2018). This dilemma demands that the system must be stable enough to retain previously acquired knowledge, yet plastic enough to effectively integrate new information.

Empirically, standard deep networks subjected to long sequences of tasks or slowly drifting data streams often exhibit a decline in their learning capability (Dohare et al., 2024; Berariu et al., 2021; Dohare et al., 2021; Nikishin et al., 2022; Lyle et al., 2023). This phenomenon, termed loss of plasticity (LoP), is distinct from catastrophic forgetting (McCloskey and Cohen, 1989; Ratcliff, 1990; French, 1999), where new learning overwrites old knowledge. LoP specifically refers to the diminished ability to learn new information effectively over time. Common symptoms include exploding weight magnitudes (Nikishin et al., 2022), activation saturation, the emergence of "dead" ReLU units (whose upstream parameters cease to update) (Nair and Hinton, 2010; Sokar et al., 2023; Dohare et al., 2021; Lyle et al., 2022), a collapse in the effective rank of hidden layer representations indicating reduced feature diversity (Papyan et al., 2020; Huh et al., 2023; Kumar et al., 2020; Gulcehre et al., 2022), and redundancy or diminishing contributions from network components like attention heads or filters (Lyle et al., 2023). Many of these issues have been highlighted in recent studies focusing on LoP (Dohare et al., 2023; Kumar et al., 2024; Ash and Adams, 2020).

Dohare et al. (2024) argue that such failures are intrinsically linked to the back-propagation algorithm itself. They posit that gradient descent optimized for transient single-task learning relies heavily on the initial random state for exploration, a resource that is consumed and not replenished during prolonged training. Their work demonstrates that standard deep learning methods can lose plasticity until they perform comparably to linear networks and suggests that maintaining plasticity requires mechanisms beyond pure gradient descent such as continually injecting diversity via methods like their proposed continual backpropagation.

Goal of this paper. Motivated by observations about LoP, our paper revisits the dynamics of gradient descent and back-propagation through the lens of dynamical systems theory. We seek to answer the question:

> *What structural features inherent in gradient flow dynamics inevitably lead to LoP, and how might we design algorithms or architectures capable of perpetual adaptation?*

The central theorem in this work is that the tendency of gradient-based optimization to favor low-rank or "simple" representations lies at the heart of plasticity loss. While properties like low effective rank and simplicity bias are often associated with improved generalization in the standard two-phase learning paradigm (Huh et al., 2023; Papyan et al., 2020; Zhang et al., 2017), we argue that these very properties become detrimental in continual learning settings. By reducing the effective dimensionality of the network's feature space, they limit its capacity to adapt to novel information, thus contributing to the LoP observed by (Dohare et al., 2024) and others. While loss of plasticity (LoP) has been previously linked to hallmarks of low-rank representations in the literature, our work introduces a novel perspective and develops a formal framework that systematically unifies several previously disparate observations. In addition, our theory is the first to establish explicit connections between LoP and the geometries induced by learning dynamics and cloning, two active research domains that we argue hold considerable potential for advancing the study of LoP.

1.1 Background and Related Work

Loss of Plasticity (LoP). A network is said to suffer a loss of plasticity when, after some period of training, it can no longer acquire new information as effectively as a freshly-initialised model of the same architecture. LoP has been documented in a variety of continual-learning and reinforcement-learning settings (Dohare et al., 2024; Nikishin et al., 2022). Crucially, LoP is distinct from catastrophic forgetting: performance on past tasks may remain intact while the ability to learn future tasks degrades (Lyle et al., 2023). Typical symptoms include exploding weight norms, growing numbers of dead (saturated) units, and a collapse of the effective rank of hidden representations (Dohare et al., 2021; Papyan et al., 2020).

Previous explanations. Early accounts linked LoP to individual pathologies, e.g., weight-norm growth or activation sparsity. However, these factors alone failed to consistently explain the phenomenon (Lyle et al., 2022). A more recent view connects LoP to a degeneration of the network's neural tangent kernel (NTK) that is once the NTK becomes low-rank, many directions in function space receive negligible gradient and can no longer be learned (Lyle et al., 2023). This perspective suggests that LoP is multi-faceted with diverse surface-level defects (e.g., dead units, duplicated features) sharing the common consequence of reducing the network's effective degrees of freedom.

Geometric Singularities and Learning Dynamics. The connection between overparameterization, reduced dimensionality, and learning difficulties has deep roots in the analysis of neural network geometry. Hierarchical models exhibit singularities that are regions in parameter space where the mapping from parameters to function is not unique (e.g., due to unit duplication or vanishing). Foundational work by Fukumizu and Amari (2000) and Amari et al. (2006) demonstrated that these singularities cause the Fisher Information Matrix to degenerate, leading to slow learning dynamics (plateaus) as gradient descent is attracted to these regions.

Implicit Bias and Stochastic Dynamics. Recent work highlighted how the optimization algorithm itself contributes to the collapse towards simpler representations. Chen et al. (2023) analyzed the implicit bias

of Stochastic Gradient Descent (SGD), showing that gradient noise induces an attractive force towards these singular regions (termed "Invariant Sets"). This "Stochastic Collapse" suggests that the tendency towards LoP states is exacerbated by the stochastic nature of the optimization process, even if the regions are unstable under deterministic gradient descent. Furthermore, Wang et al. (2024) empirically demonstrated that maintaining trainability (ability to fit new data) does not guarantee generalizability (performance on unseen data), emphasizing the need for methods that restore genuine plasticity.

2 LoP Manifolds: Traps for Gradient Descent

In this section, we lay the groundwork for our analysis by defining Loss of Plasticity (LoP) within a dynamical systems framework and recalling the standard definitions for feed-forward neural networks and back-propagation. The stability of LoP manifolds, a crucial concept for understanding their persistence, will be discussed later in Sec. 2. Let $\theta \in \Theta \subseteq \mathbb{R}^p$ represent the parameters of a neural network. We consider training on a stream of data $\{(x_i, y_i)\}_{i=1}^N$ using gradient descent or its stochastic variants. The objective is typically to minimize a loss $\sum_{i=1}^N \mathcal{L}(\hat{y}_\theta(x_i), y_i)$, which we can succinctly refer to as $\mathcal{L}(\theta)$. This allows us to define LoP based on the the trajectory of parameters $\theta(t)$ in the parameter space Θ as driven by the negative gradient in the loss landscape.

Definition 2.1 (LoP Manifold)**.** *A manifold $\mathcal{M} \subset \Theta$ induces LoP if the gradient of the loss function is tangent to the manifold at every point on the manifold. That is, $\nabla_\theta \mathcal{L}(\theta) \in T_\theta \mathcal{M}$ for all $\theta \in \mathcal{M}$, where $T_\theta \mathcal{M}$ denotes the tangent space of $\mathcal{M}$ at θ. This tangency condition ensures that once the gradient flow enters $\mathcal{M}$, it remains within $\mathcal{M}$ under the dynamics of gradient flow $\frac{d\theta(t)}{dt} = -\nabla_\theta \mathcal{L}\big(\theta(t)\big)$.*

Remark 2.1. *If the conditions in Definition 2.1 hold irrespective of the specific data distribution generating the loss $\mathcal{L}$, which we can think of as functional LoP, and is our primary area of interest. Such LoP arises from the network architecture and gradient descent dynamics alone and is particularly relevant as it persists even if the task or data distribution evolves.*

Given these definitions, we can formalize existence of these LoP manifolds, restricting subsequent learning. We present a central theorem that jointly addresses LoP arising from frozen and duplicate units. The intuition is that once units become unresponsive (frozen) or perfectly redundant (cloned), they tend to remain so under standard gradient-based optimization.

Theorem 2.1. *Let $G = (V, E)$ be the network's computational DAG and let $\theta = \{\theta_{uv} : (u \to v) \in E\} \in \Theta$ denote the edge parameters.*

1. ***Frozen-unit manifold** $\mathcal{M}_F$. Assume there exists $F \subset V$ such that, for all finite inputs encountered, each $v \in F$ is persistently saturated ($f'(z_v) = 0$). Then the gradients w.r.t. all incoming parameters to v vanish on any mini-batch, so those coordinates remain fixed; writing the linear constraints as $\theta_{\text{in}}(v) = \text{const}$ for all $v \in F$, the affine subspace $\mathcal{M}_F := \{\theta : \theta_{\text{in}}(v) = \text{const}\ \forall v \in F\}$ satisfies $\nabla\mathcal{L}(\theta) \in T_\theta \mathcal{M}_F$ and GD/SGD updates initialized in $\mathcal{M}_F$ remain in $\mathcal{M}_F$.*

2. ***Cloning manifold** $\mathcal{M}_C$. Assume a partitioning of nodes into disjoint blocks $\{S_1, \ldots, S_k\}$ exists with following properties. For every ordered block pair (S_i, S_j), we have the linear equalities $\sum_{v \in S_j} \theta_{uv} = \sum_{v \in S_j} \theta_{u'v}$ for all $u, u' \in S_i$ (equal row-sums) and $\sum_{u \in S_i} \theta_{uv} = \sum_{u \in S_i} \theta_{uv'}$ for all $v, v' \in S_j$ (equal column-sums). Let $\mathcal{M}_C$ be the affine subspace of Θ consisting of all θ satisfying these constraints. If $\theta \in \mathcal{M}_C$, then (i) all units within any block share the same forward values on any input, (ii) all units within any block share the same backpropagated errors on any input, and therefore (iii) the per-edge gradients are constant across edges connecting the same block pair, i.e., for any (u, v) and (u', v') with $u, u' \in S_i$ and $v, v' \in S_j$, $\partial\mathcal{L}/\partial\theta_{uv} = \partial\mathcal{L}/\partial\theta_{u'v'}$. Hence $\nabla\mathcal{L}(\theta) \in T_\theta \mathcal{M}_C$ and GD/SGD updates initialized in $\mathcal{M}_C$ remain in $\mathcal{M}_C$.*

Note that both LoP manifolds $\mathcal{M}_F$ and $\mathcal{M}_C$ are defined as linear LoP manifolds in the sense of Definition 2.1. Formal proofs and further details are provided in Appx. A.2.

Proof idea. *Frozen units.* If a unit stays in a regime with $f'(z_v) = 0$ for all finite inputs (e.g., $\tanh$ with very large $\|\theta_{\text{in}}(v)\|$ or ReLU with a large negative bias), then $\partial\mathcal{L}/\partial\theta_{\text{in}}(v) \approx 0$; its incoming parameters are fixed, so updates are tangent to $\mathcal{M}_F$. *Cloning via redistribution.* The key idea the row/column-sum equalities mean total incoming/outgoing weight from/to any block is redistributed within each block pair (S_i, S_j). Thus, the total contribution to the forward and backward of each unit within a block remains identical, implying the forward and backward cloning (properties (i) and (ii)) within blocks. Thus, per-edge gradients $d\mathcal{L}/d\theta_{uv} = h(u)\,\delta(v)$, are therefore by forward and backward symmetries across the blocks the gradients will be constant for any two units in these blocks $(u, v) \in S_i \times S_j$. These block-wise constant gradients trivially satisfy the row-sum and column-sum equalities, and hence are tangent to $\mathcal{M}_C$, and first-order updates remain on both manifolds.

Remark 2.2. *It is important to note that the Duplicate Manifold $\mathcal{M}_D$ (defined formally via Incoming and Outgoing Equitable partitions in Appx. A.2) represents a significant generalization of the cloning concepts typically discussed in literature. Prior analyses of singularities (Fukumizu and Amari, 2000) or invariant sets (Chen et al., 2023) generally define cloned units by requiring their associated weights to be strictly identical (the block-wise constant condition in our terminology). Our framework proves that invariance under gradient descent holds even under the relaxed condition of equitability, where individual weights may differ as long as specific incoming and outgoing sums are maintained. This significantly broadens the class of structures identified as LoP manifolds.*

Remark 2.3. *The cloning LoP manifold naturally lends itself to gradient descent and stochastic gradient descent, regardless of the order which we process the samples, will remain strictly within the manifold. This extends to virtually all variations of gradient descent based optimizations, namely Stochastic Gradient Descent (SGD), SGD with momentum, and Adam, as long as the optimizer is initialized at the onset of cloning. The only exception to this is weigh decay which could break some symmetries. This fact can be empirically observed across our cloning experiments, showing that across a wide range of optimization schemes the model remains trapped onto to the LoP manifold.*

Remarkably, the theorem admits a *modular* version, which allows us to create practical cloning certificate for modern architectures (see Appx. A.3).

Theorem 2.2 (Modular Cloning (informal))**.** *This cloning property can be decomposed modularly. If a network is composed of individual modules (e.g., layers or blocks), and each module locally satisfies the cloning invariance properties—namely, (1) cloned inputs produce cloned outputs (Forward Invariance), (2) cloned backward signals at the outputs produce cloned backward signals at the inputs (Backward Invariance), and (3) gradient updates preserve these invariances (Persistence)—then the entire network resides on a cloning manifold, provided the cloning profiles (partitions) are consistent at the interfaces between modules.*

To empirically test the validity of the cloning manifold and their potential escape mechanisms we conduct *cloning experiments*. First, a base model (e.g., an MLP) is trained on a specific task. Subsequently, a larger model is constructed by expanding the base model. This expansion involves increasing the width of the model for MLPs, the number of channels for CNNs and ResNets, and the feature dimension for ViTs. The weights of the cloned model are initialized in such a way that its activations are identical to those of the base model. This effectively creates blocks of units that have identical activations. Next, we train both the base and cloned models on the same task and monitor their training progress through the loss curve, the effective rank of representations, and the cloning R^2 score. Figure 2.1 presents the results of such experiments on MLPs, shedding light on the dynamics within and escapes from these LoP manifolds. The empirical validation of these claims, such as demonstrating perfect cloning under specific initializations or the persistence of dead units, can be found in Fig. 2.1 and Appx. B. Notably, despite Adam violating the symmetry conditions required for Theorem 2.1, the empirical evidence suggests that it frequently fails to escape the manifold. This observation implies the existence of a stronger theory capable of explaining this phenomenon.

Escaping the LoP manifolds with perturbations. While a comprehensive theoretical analysis of the stability of empirically observed LoP manifolds is beyond the scope of this work, our empirical investigations indicate that these manifolds are frequently unstable or resemble saddle-like shapes. Certain types of noise or symmetry-breaking operations can help models escape these manifolds. We highlight two common perturbations: (1) *Noisy SGD* is a modification of SGD that adds Gaussian noise to the computed gradients before parameter updates. The magnitude of this injected noise is usually proportional to the norm of the gradient, with its initial relative strength gradually decreasing over successive steps. By applying this noise after cloning, we can determine whether the model can escape the LoP manifold or if it will fall back. (2) *Dropout* introduces stochasticity in the forward and backward passes by randomly zeroing activations. For cloned units, this breaks the symmetry because different clones might be active in different dropout masks, leading to divergent gradient updates. This is supported by experiments where dropout helps a model escape an artificially induced cloning manifold (see Fig. 2.1).

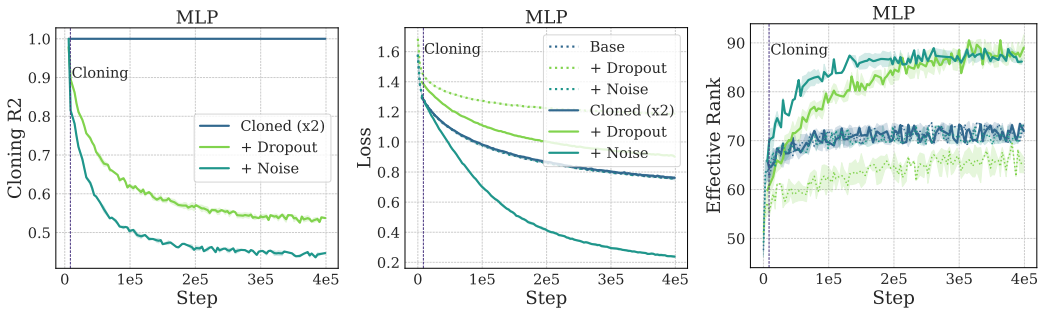

Figure 2.1: Cloning MLPs experiments. The empirical data validates Theorem 2.1 on duplicate manifold LoP. The cloned network dynamics remain confined in the base network manifold when using SGD, however using Noisy SGD or Dropout the dynamics can escape the manifold. **Left**: Cloning R^2 score quantifies the proportion of variance in individual unit activations within a cloned block that is explained by the mean activation of that block. An R^2 score of 1 indicates perfect cloning (units in a block are nearly identical), while a 0 score indicates no explained variance. See Appx. B.1.2 (Appendix B) for the precise formula and calculation details. **Middle**: Training loss comparison. *Cloned loss* refers to the loss of the cloned model during its training phase, while *base loss* refers to the loss of the original base model, which continues training for comparison. **Right**: Effective rank evolution showing representational diversity.

Both noisy SGD and dropout act as symmetry-breaking operations. In the case of dropout, both forward and backward passes are asymmetric for cloned units. For noisy SGD, the backward pass (gradient update) becomes asymmetric. This asymmetry causes the parameters of notionally cloned units to slowly diverge. Remarkably, in our MLP experiments, even a small amount of gradient noise, e.g., a single step with noise magnitude 0.01 relative to gradient norms, suffices to initiate escape from an LoP manifold, though stronger noise generally leads to faster escape. In contrast, in settings such as Vision Transformers, while the model could escape from the manifold with a small perturbation, it did not move very far from it. More experimental studies into this direction would be vital to better understand the stability of these LoP manifolds.

3 Emergence of Loss of Plasticity from Linear–Nonlinear Rank Dynamics

Having established the existence of LoP manifolds, we now discuss the mechanisms within standard training that drive their formation. The optimization process can be seen as a trajectory, beginning with an expansion of representational diversity as features propagate through nonlinear layers and become increasingly decorrelated

(Poole et al., 2016). However, this initial growth is followed by a compression phase, where the network simplifies its representation to retain only the most relevant features for the task. This low-dimensional structure is a key characteristic of neural collapse, where last-layer features for each class converge to their means in a highly organized geometric configuration (Papyan et al., 2020). This is also consistent with the information bottleneck principle, which describes training as a process of first fitting the data and then compressing the representation to discard irrelevant information (Shwartz-Ziv and Tishby, 2017). Here, we argue that these two principles derive the model towards the LoP manifolds we identified earlier. Thus, this provides a direct link between these fundamental compression dynamics inherent to deep network optimization, and emergence of LoP.

To diagnose whether features are diversifying or compressing during training, we track a smooth surrogate of the rank of the feature correlation matrix. Exact rank is numerically unstable, because it is not a continuous nor a differentiable map from the matrix space. Therefore, we use differentiable proxies to rank such as Rényi-2 rank, $\mathrm{er}_2(M) = (\mathrm{tr}\, M)^2/\|M\|_F^2$, or the Shannon effective rank, $\mathrm{er}(M) = \exp(H(\lambda(M)/\mathrm{tr}\, M))$. Both increase when the eigenmass is evenly distributed or dominated by a few values. Note that both of these surrogates are maximized when matrix M has equal eivenvalues and minimized when it a rank-1 matrix. The following theorem offers an insight into how nonlinear layers contribute to the formation of features.

Theorem 3.1 (rank gain across one linear–nonlinear step)**.** *Assume ϕ is nonlinear with a Hermite expansion. Let C be the pre-activation correlation matrix with unit diagonal. For an activation ϕ, define the correlation kernel $K_\phi(r) = \mathrm{Corr}(\phi(x), \phi(y))$ where (x, y) are jointly Gaussian with correlation r. The nonlinearity acts entrywise on correlations, producing $K_\phi(C)$. Its correlation kernel satisfies $K_\phi(0) = 0$, $K_\phi(1) = 1$, and $|K_\phi(r)| < |r|$ for all $|r| < 1$. For any correlation matrix C with unit diagonal the Rényi-2 effective rank obeys*

$$\frac{\mathrm{er}_2(K_\phi(C))}{\mathrm{er}_2(C)} = \frac{d + \sum_{i \neq j} C_{ij}^2}{d + \sum_{i \neq j} K_\phi(C_{ij})^2} \geq 1,$$

and the ratio equals 1 only when every off-diagonal magnitude C_{ij} is 0 or 1.

Note that the theorem implies that any correlations $|C_{ij}| \in (0, 1)$ strictly increase Rényi-2 rank after the nonlinearity because K_ϕ is below the identity map everywhere other than those fixed points. Note that this gap also depends on the activation itself, and how nonlinear it is. We can summarize this gap via a scalar that summarizes how strongly the kernel pulls intermediate correlations toward the fixed points $\{0, 1\}$, which we term the decorrelation strength $\kappa_\phi = \max_{r \in [0,1]}(r^2 - K_\phi(r)^2)$. Larger κ_ϕ implies a larger rank gain for the same spectrum of C. For a formal statement and proof of this statement see Appx. A.1.

Implication for emergence of frozen units So far we implicitly treated pre-activations as standardized. In practice, their first and second moments drift during training. Allowing nonzero means and nonunit variances changes the operating regime of the activation and thus modifies the kernel K_ϕ and the decorrelation strength κ_ϕ. For ReLU, making the effective bias more negative increases nonlinearity and raises κ_ϕ. For $\tanh$, increasing effective gain does the same. These are precisely the regimes where the derivative is near zero on most inputs, which explains why training that enhances decorrelation also creates units that are nearly always inactive or saturated. One insight from this connection is that we can unify two empirically observed phenomena: "dead ReLU" and "frozen units" as both symptoms are caused by the same underlying force.

Implication for creation of duplicate or cloned units Neural collapse is a widely observed endpoint in which the penultimate features are low-rank, and the class means form a simplex or Equiangular Tight Frame (ETF) structure (Papyan et al., 2020). Under this geometry, the theorem makes a precise prediction. To preserve low rank through the nonlinearity, correlations must lie at kernel fixed points that do not increase effective rank, which means they should be close to 0 or 1. Maintaining rank C therefore requires roughly C orthogonal directions for class structure and $d - C$ directions that are near duplicates within classes. Linear layers and the loss encourage this compression and duplication, while the nonlinearity would otherwise expand diversity unless within-class correlations are driven close to 1 and others close to 0.

The two implications above provide a theoretical perspective on why duplicate features and frozen units frequently emerge at or near convergence, and why the resulting representation lies close to LoP manifolds, as proven in Theorem 2.1.

3.1 Experimental Validation

We validate our theory with experiments on MLP, CNN, ResNet, and ViT architectures, training them continually on a sequence of 40 5-class tasks derived from Tiny ImageNet. We track the emergence of LoP symptoms, including dead units, duplicate units, and effective rank degradation. Full experimental details are provided in Appx. B. The experimental evidence confirms our intuitions (see Appx. B and Figs. 3.1, 3.2, B.4 and B.6): depending on the architecture, we observe that a degradation in the model performance is concomitant with the emergence of duplicate or frozen units, and a corresponding decrese in representational diversity. Our inquiry so far highlights two key pathways to LoP common symptoms: (1)

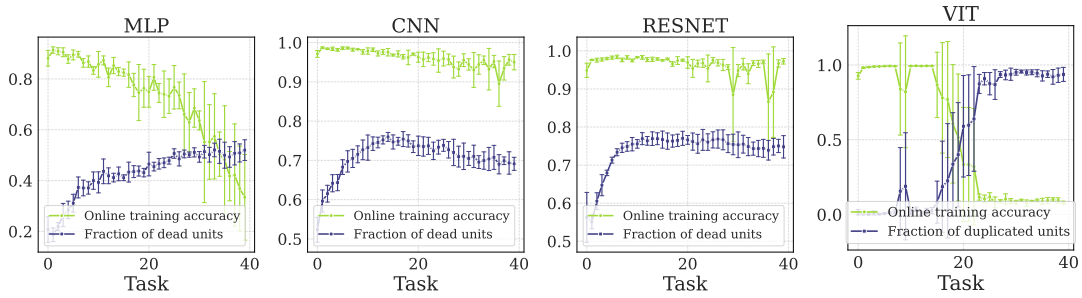

Figure 3.1: Causes and symptoms of Loss of Plasticity emerging during continual learning. The plots illustrate (across different architectures like MLP, CNN, ResNet, and ViT from left to right) an increase in the fraction of dead or duplicate units during training, coincidental with a decrease in training accuracy. These are key indicators of LoP. (Details of experimental setup in Appx. B).

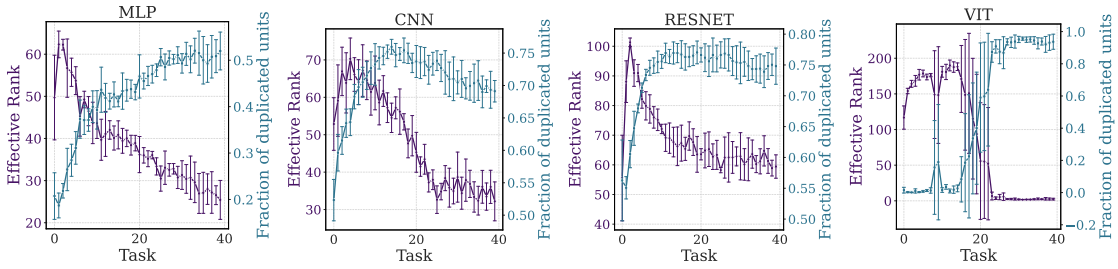

Figure 3.2: Co-evolution of Effective rank and LoP symptoms, such as dead or duplicate units in the network during continual training. (Experimental details in Appx. B).

Emergence of **duplicate features**, where distinct computational units, or groups of units, within a network layer effectively learn to become identical or highly correlated, as a potential consequence of attempting to lower representational rank, (2) Emergence of **frozen or dead features**, where weights and biases of a unit stop learning, as a result of attempting to maximize rank increase (leading to saturation) or to flatten the loss landscape around the current parameters.

4 MITIGATION AND RECOVERY STRATEGIES

Having discussed the emergence of LoP symtoms and the existence of LoP manifolds, we now turn to strategies for preventing their formation or recovering from them if they have already occurred.

Preventing LoP with Normalization. As established in Sec. 3, one primary cause for activations becoming frozen is their pre-activations drifting into saturated regions. It is therefore natural to expect that normalization layers like Batch Normalization (BN) or Layer Normalization (LN) can help prevent this. By standardizing pre-activation statistics, these layers can keep activations operating in their more dynamic, non-linear range. Even with learnable affine parameters (γ, β) after normalization, these parameters often act to maintain pre-activations within a "healthy" range, rather than pushing them into extreme values that cause saturation (e.g., consistently negative for ReLU). This is widely supported by empirical evidence (see Appx. B, Figures like Fig. 3.1 in Sec. 3, and Fig. B.3). BN and LN generally help maintain higher effective rank of representations throughout training (as seen in Fig. 3.2) and concurrently prevent frozen/dead features and excessive feature duplication from becoming dominant.

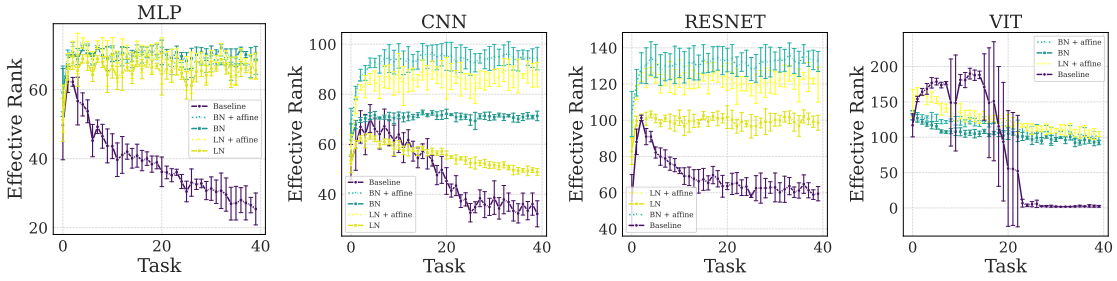

Figure 4.1: Evolution of the Effective rank during training for architectures with and without normalization layers. Dotted lines represent normalization with affine parameters. (Experimental details in Appx. B).

Recovery from LoP via Perturbations. What if LoP conditions, such as widespread frozen units or extensive feature cloning, have already set in? In such cases, mitigation strategies like normalization, which act proactively, may no longer be sufficient to reverse the state, as indicated by cloning experiments where normalization alone doesn't break perfect, established clones. However, similar to our discussion on manifold stability (Section 2), injecting noise into the training process can be a viable recovery strategy. The principle is that if the LoP manifold is unstable or saddle-like, perturbations can allow the optimizer to find an escape route. Noisy SGD and the more sophisticated Continual Backpropagation (Dohare et al., 2024) are examples of such mechanisms. We test recovery from LoP on the "bit-flipping" benchmark, an online regression task with a non-stationary target function designed to challenge a model's adaptability. A detailed description of the task is in Appx. B.1.1. In order to demonstrate the recovery potential of noise injection, we design an experiment where the first half of 5M samples is processed by plain Stochastic Gradient Descent (SGD) and in the second half we switch the learning rule to Continual Backpropagation (CBP). Figure B.1 clearly shows a reversal in trend when the switch happens: whereas SGD causes the representations' rank to drop and the online training loss to increase, CBP amplifies the features' rank and reduces the online training loss, effectively *recovering plasticity.* Additionally Fig. B.2 illustrates how aspects like rank and feature duplication are affected by the dimensionality of the model. The disparity between SGD and CBP is only increased by the model size, hinting that the model scale might aggravate the symptoms of LoP. For details see Appx. B.1.1.

An interesting distinction arises when comparing artificially induced LoP (like explicit cloning) with naturally emerging LoP symptoms in challenging scenarios like continual learning. In controlled cloning setups (e.g.,

as conceptualized in Fig. 2.1, dropout can be effective in breaking the artificially imposed symmetry and allowing units to diverge. In contrast, in our continual Bit Flipping experiments, the role of dropout can be mixed or even detrimental. While it might prevent some forms of LoP, it can also hinder the consolidation of new knowledge or exacerbate forgetting if it too aggressively discards learned information relevant to the new task. This suggests that the optimal strategy for maintaining or recovering plasticity might be context-dependent.

5 Conclusion

This work has presented a mathematical framework to understand Loss of Plasticity (LoP) in deep neural networks through the lens of dynamical systems. We formally defined LoP manifolds as regions in parameter space that trap gradient-based optimization. We identified two primary mechanisms for their formation: the saturation of activations leading to frozen units, and representational redundancy manifesting as cloned-unit manifolds. Our analysis reveals that these LoP states are frequently characterized by a reduction in the effective rank of representations. We investigated how architectural choices, such as normalization, can mitigate the emergence of LoP, and how perturbations, like noise injection, can facilitate escape from these restrictive manifolds, depending on their stability.

A key finding from our investigation is the inherent tension between learning objectives in static and dynamic environments. While properties conducive to good generalization on a fixed dataset, such as the emergence of low-rank features or simplicity biases, appear to be beneficial, they can also lead to a loss of adaptability when the learning process is extended over time or across changing tasks. This suggests that continual learning necessitates mechanisms that actively preserve or regenerate representational diversity.

This study raises several intriguing questions and suggests directions for future research. From a theoretical perspective, our analysis has primarily focused on linear or affine LoP manifolds. However, it remains an open question whether non-linear LoP manifolds exist and could potentially arise in practical network training scenarios. Additionally, a more comprehensive theoretical understanding of the stability conditions for various types of LoP manifolds is essential. Specifically, we need to determine the precise architectural or data conditions that lead to one type of stability over another.

Numerically, the curvature of the loss landscape in directions normal to an LoP manifold is critical. Even for an unstable manifold, if the negative curvatures are very slight (near flat), escaping might necessitates significant perturbations or many training steps. Characterizing these curvatures and their impact on escape dynamics would be valuable. While we have demonstrated that models can escape artificially cloned LoP manifolds with interventions like dropout or noise, the question remains: can a model, once recovered from such a state, explore the parameter space as effectively and find solutions as generalizable as a model trained from a fresh, random initialization? This question is of significant practical importance, as it explores whether we can fully restore exploratory capacity after falling into a highly restricted subspace.

One of the most intriguing outcomes of this work is the connection between unit cloning, a phenomenon often studied in model compression or network analysis, and LoP in continual learning. These have been largely treated as separate fields of inquiry. However, our theoretical framework, particularly the theorems regarding cloned units, reveals a deep link, suggesting that insights and tools can be transferred between these domains. This raises questions about whether techniques from continual learning, such as noisy backpropagation or methods like Continual Backpropagation (CBP) (Dohare et al., 2024), could be beneficial in the context of model expansion or escaping cloned states in other scenarios.

Ultimately, understanding and overcoming LoP is crucial for building AI systems that can learn continuously and adapt robustly in an ever-changing world. By providing a mathematical characterization of some fundamental barriers to such adaptation, we aim to pave the way for the development of new architectures and learning algorithms that can sustain plasticity indefinitely, leading to truly lifelong learning agents.

REFERENCES

Wickliffe C Abraham and Anthony Robins. Memory retention–the synaptic stability versus plasticity dilemma. *Trends in Neurosciences*, 28(2):73–78, 2005.

Shun-ichi Amari, Hyeyoung Park, and Tomoko Ozeki. Singularities affect dynamics of learning in neuromanifolds. *Neural computation*, 18(5):1007–1065, 2006.

Jordan Ash and Ryan P. Adams. On warm-starting neural network training. In *Advances in Neural Information Processing Systems*, volume 33, pages 3884–3894, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/288cd2567953f06e460a33951f55daaf-Abstract.html.

Tudor Berariu, Wojciech Czarnecki, Stefano De, Jörg Bornschein, Samuel Smith, Razvan Pascanu, and Claudia Clopath. A study on the plasticity of neural networks. *arXiv preprint arXiv:2106.00042*, 2021. URL https://arxiv.org/abs/2106.00042.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547. Springer, 2018. URL https://openaccess.thecvf.com/content_ECCV_2018/html/Arslan_Chaudhry__Riemannian_Walk_ECCV_2018_paper.html.

Feng Chen, Daniel Kunin, Atsushi Yamamura, and Surya Ganguli. Stochastic collapse: How gradient noise attracts sgd dynamics towards simpler subnetworks. *Advances in Neural Information Processing Systems*, 36:35027–35063, 2023.

Shibhansh Dohare, Richard S. Sutton, and A. Rupam Mahmood. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021. URL https://arxiv.org/abs/2108.06325.

Shibhansh Dohare, Juan F. Hernandez-Garcia, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Maintaining plasticity in deep continual learning. *arXiv preprint arXiv:2306.13812*, 2023. URL https://arxiv.org/abs/2306.13812.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, 2024.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4): 128–135, 1999.

Kenji Fukumizu and Shun-ichi Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural networks*, 13(3):317–327, 2000.

Caglar Gulcehre, Srivatsan Srinivasan, Jakub Sygnowski, Georg Ostrovski, Mehrdad Farajtabar, Matt Hoffman, Razvan Pascanu, and Arnaud Doucet. An empirical study of implicit regularization in deep offline rl. *arXiv preprint arXiv:2207.02099*, 2022.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. In *International Conference on Learning Representations (ICLR)*, 2023.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. Maintaining plasticity in continual learning via regenerative regularization. In *Proceedings of the 3rd Conference on Lifelong Learning Agents*. PMLR, 2024. URL https://arxiv.org/abs/2308.11958.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. *arXiv preprint arXiv:2204.09560*, 2022.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 23190–23211. PMLR, 2023. URL https://proceedings.mlr.press/v202/lyle23b.html.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16828–16847. PMLR, 2022. URL https://proceedings.mlr.press/v162/nikishin22a.html.

Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in neural information processing systems*, 29, 2016.

Roger Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285, 1990.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017. Submitted March 2, 2017, last revised April 29, 2017.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32145–32168. PMLR, 2023. URL https://proceedings.mlr.press/v202/sokar23a.html.

Zhenyi Wang, Enneng Yang, Li Shen, and Heng Huang. A comprehensive survey of forgetting in deep learning beyond continual learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

A THEORETICAL APPENDIX

This section contains detailed proofs of theorems and lemmas, further theoretical derivations, and discussions extending the concepts presented in the main paper.

A.1 FORMAL PROOF OF THE RANK–GAIN THEOREM UNDER NON-LINEAR ACTIVATIONS

This section states and proves in full detail the theorem used in Sec. 3. We work with standard Gaussian inputs and use Hermite expansions to characterize the correlation kernel of a nonlinearity.

Hermite basis and notation Let $(h_k)_{k\geq 0}$ be the orthonormal probabilists' Hermite polynomials in $L^2(\gamma)$ where $\gamma = \mathcal{N}(0,1)$, with $h_0(z) = 1$, $h_1(z) = z$, and $\mathbb{E}[h_k(Z)h_\ell(Z)] = \delta_{k\ell}$ for $Z \sim \gamma$. Any $\phi \in L^2(\gamma)$ admits a Hermite expansion

$$\phi(z) = \sum_{k=0}^{\infty} a_k h_k(z), \qquad a_k = \mathbb{E}\big[\phi(Z)h_k(Z)\big].$$

Write $\sigma_\phi^2 = \mathrm{Var}(\phi(Z)) = \sum_{k\geq 1} a_k^2$. We call ϕ nonlinear when at least one coefficient with $k \geq 2$ is nonzero.

Correlation kernel For jointly Gaussian (X, Y) with $\mathbb{E}X = \mathbb{E}Y = 0$, $\mathrm{Var}(X) = \mathrm{Var}(Y) = 1$, and $\mathrm{corr}(X, Y) = r \in [-1, 1]$, define

$$K_\phi(r) \;=\; \mathrm{Corr}\big(\phi(X), \phi(Y)\big) \;=\; \frac{\mathbb{E}[\phi(X)\phi(Y)] - \mathbb{E}[\phi(X)]\mathbb{E}[\phi(Y)]}{\sqrt{\mathrm{Var}(\phi(X))\,\mathrm{Var}(\phi(Y))}}\,.$$

Using the Hermite expansion and Mehler's identity $\mathbb{E}[h_k(X)h_\ell(Y)] = \delta_{k\ell}\, r^k$ gives

$$K_\phi(r) = \frac{\sum_{k\geq 1} a_k^2 r^k}{\sum_{k\geq 1} a_k^2} = \sum_{k\geq 1} w_k r^k, \qquad w_k := \frac{a_k^2}{\sum_{\ell\geq 1} a_\ell^2} \geq 0, \;\; \sum_{k\geq 1} w_k = 1. \tag{1}$$

Hence K_ϕ is a convex combination of the monomials r^k for $k \geq 1$.

Lemma A.1 (basic properties of K_ϕ)**.** *If ϕ is nonlinear with $\phi \in L^2(\gamma)$, then the correlation kernel K_ϕ defined in* (1) *satisfies*

$$K_\phi(0) = 0, \qquad K_\phi(1) = 1, \qquad |K_\phi(r)| < |r| \text{ for all } |r| < 1.$$

Proof. The first two identities follow by evaluating (1) at $r = 0$ and $r = 1$. For the strict inequality, write $K_\phi(r) = \sum_{k\geq 1} w_k r^k$ with weights $w_k \geq 0$, $\sum_k w_k = 1$. If $|r| < 1$ and there exists $k \geq 2$ with $w_k > 0$ (nonlinearity), then

$$|K_\phi(r)| \leq \sum_{k\geq 1} w_k |r|^k = |r|\Big(w_1 + \sum_{k\geq 2} w_k |r|^{k-1}\Big) < |r|\Big(w_1 + \sum_{k\geq 2} w_k\Big) = |r|.$$

□

Lemma A.2 (entrywise action on Gaussian correlation matrices)**.** *Let $Z = (Z_1, \ldots, Z_d)^\top \sim \mathcal{N}(0, C)$ with C a correlation matrix. For $\phi \in L^2(\gamma)$,*

$$\mathrm{Corr}\big(\phi(Z_i), \phi(Z_j)\big) \;=\; K_\phi(C_{ij}) \qquad \text{for all } i, j.$$

Equivalently, the post-activation correlation matrix equals $K_\phi(C)$ entrywise. Moreover,

$$K_\phi(C) = \sum_{k\geq 1} w_k\, C^{\odot k},$$

where $C^{\odot k}$ is the k-th Hadamard power and $(w_k)_{k\geq 1}$ are as in (1)*. Hence $K_\phi(C)$ is a correlation matrix: it is positive semidefinite and has unit diagonal.*

Proof. Fix i, j. By the same Hermite calculation used for (1), with $r = C_{ij}$ we have

$$\mathrm{Corr}\big(\phi(Z_i), \phi(Z_j)\big) = \frac{\sum_{k\geq 1} a_k^2 r^k}{\sum_{k\geq 1} a_k^2} = K_\phi(r).$$

Stacking these equalities over all pairs (i, j) yields the entrywise identity and the series $K_\phi(C) = \sum_{k\geq 1} w_k C^{\odot k}$. Each Hadamard power $C^{\odot k}$ is positive semidefinite by the Schur product theorem, and the diagonal entries are $(C_{ii})^k = 1$, so the nonnegative convex combination is positive semidefinite with unit diagonal. □

Lemma A.3 (Frobenius contraction)**.** *For any correlation matrix C and nonlinear ϕ with kernel K_ϕ as above,*

$$\sum_{i\neq j} K_\phi(C_{ij})^2 \;\leq\; \sum_{i\neq j} C_{ij}^2,$$

with strict inequality if there exists $i \neq j$ such that $|C_{ij}| < 1$.

Proof. By Lemma A.1, $|K_\phi(r)| < |r|$ for every $|r| < 1$, and trivially $|K_\phi(r)| \leq |r|$ for $|r| \leq 1$. Apply this pointwise to each off-diagonal entry C_{ij} and sum the squares. If some $|C_{ij}| < 1$, the corresponding term is strictly reduced, and no term increases, so the sum is strictly reduced. □

We can now state and prove the theorem used in the main text.

Theorem A.1 (rank gain across one linear–nonlinear step)**.** *Let C be a $d \times d$ correlation matrix and let $\phi \in L^2(\gamma)$ be nonlinear. Define K_ϕ by* (1)*. Then $K_\phi(C)$ is a correlation matrix and*

$$\frac{\mathrm{er}_2\big(K_\phi(C)\big)}{\mathrm{er}_2(C)} \;=\; \frac{d + \sum_{i\neq j} C_{ij}^2}{d + \sum_{i\neq j} K_\phi(C_{ij})^2} \;\geq\; 1.$$

Moreover, the ratio is strictly greater than 1 *whenever there exists $i \neq j$ with $|C_{ij}| < 1$. If the ratio equals* 1*, then every off-diagonal magnitude satisfies $|C_{ij}| \in \{0, 1\}$.*

Proof. By Lemma A.2, $K_\phi(C)$ is a correlation matrix and $\mathrm{tr}\, K_\phi(C) = d = \mathrm{tr}\, C$. For any correlation matrix M,

$$\|M\|_F^2 \;=\; \sum_{i,j} M_{ij}^2 \;=\; d + \sum_{i\neq j} M_{ij}^2,$$

hence

$$\mathrm{er}_2(M) = \frac{(\mathrm{tr}\, M)^2}{\|M\|_F^2} = \frac{d^2}{d + \sum_{i\neq j} M_{ij}^2}.$$

Applying this to $M = C$ and $M = K_\phi(C)$ yields the displayed ratio. Lemma A.3 gives $\sum_{i\neq j} K_\phi(C_{ij})^2 \leq \sum_{i\neq j} C_{ij}^2$, which makes the ratio at least 1, and strictly larger than 1 if some $|C_{ij}| < 1$.

For the equality case, suppose the ratio equals 1. Then the two Frobenius norms coincide, so

$$\sum_{i \neq j} \Big(C_{ij}^2 - K_\phi(C_{ij})^2 \Big) = 0.$$

Each term in the sum is nonnegative by Lemma A.1. Therefore every term must vanish, that is, for all $i \neq j$,

$$K_\phi(C_{ij})^2 = C_{ij}^2.$$

If $|C_{ij}| < 1$ and ϕ is nonlinear, Lemma A.1 gives $|K_\phi(C_{ij})| < |C_{ij}|$, a contradiction. Hence for all $i \neq j$ we must have $|C_{ij}| \in \{0, 1\}$, which proves the final claim. □

Remark A.1 (dependence on operating regime)**.** *If the pre-activation is reparameterized as $\phi_{a,b}(z) = \phi(az + b)$, the Hermite coefficients and thus the weights (w_k) in* (1) *change. The contraction at intermediate correlations can be summarized by the decorrelation strength*

$$\kappa_\phi \;=\; \max_{r \in [0,1]} \big(r^2 - K_\phi(r)^2 \big),$$

which quantifies the maximal per-entry reduction of squared correlation. For instance, more negative effective bias for ReLU and larger effective gain for $\tanh$ *increase κ_ϕ, while also reducing typical derivatives, connecting rank gains to the emergence of frozen units.*

A.1.1 Activation modulation increases decorrelation and also induces frozen units

Training changes pre–activation statistics, which modifies the operating regime of ϕ. Parameterize $\phi_{a,b}(z) = \phi(az + b)$. Both the kernel K_ϕ and the decorrelation strength κ_ϕ vary with (a, b). Appendix A.1 shows that regimes that raise decorrelation also reduce typical derivatives, yielding frozen units:

- For $\tanh$, increasing gain a raises the local decorrelation rate $\alpha_\phi = \mathbb{E}[\phi'(Z)^2]/\mathbb{E}[\phi(Z)^2] - 1$ while $\phi'(az) \to 0$ for almost all z as $a \to \infty$.
- For ReLU, making the effective bias b negative with $|b|/a$ large increases α_ϕ while $\mathbb{P}(az + b < 0) \to 1$, so $\phi'(az + b) = 0$ on most inputs.

The empirical relation between higher decorrelation and freezing is shown in Fig. A.1 and Fig. A.2.

A.2 LoP manifolds: Formal Statement and Proof

This section provides the formal definitions, statement, and proof for the LoP manifold theorem. Since the frozen manifold argument is self explanatory, we will only prove the cloning manifold result that is more non-trivial. First, let us introduce our neural network network formalization. A feed-forward neural network is defined by a directed acyclic graph $G = (V, E, w)$, where V is the set of nodes (neurons), E is the set of directed edges (connections), E representations the structure of the computational graph of the network, and $w : \mathbb{E} \to \mathbb{R}$, is the weight parameters of the network, which will be denoted $w(u, v)$ for each edge $(u, v) \in E$. Furthermore, $V_{\text{in}} \subset V$ are the input nodes, and $V_{\text{out}} \subset V$ are the output nodes. The post-activation $h(v)$ of a node $v \in V$ is computed as:

$$h(v) = \begin{cases} x_v, & \text{if } v \in V_{\text{in}}, \\ f_v\Big(\underbrace{\Sigma_{u \in \text{in}(v)} w_{u,v}\, h(u)}_{\text{pre-activation } z(v) :=} \Big), & \text{otherwise.} \end{cases}$$

Here, x_v is the input value for input node v, f_v is the activation function associated with node v, $\text{in}(v)$ is the set of nodes with edges towards v. The network output is the vector of activations $h(V_{\text{out}})$. With the formal

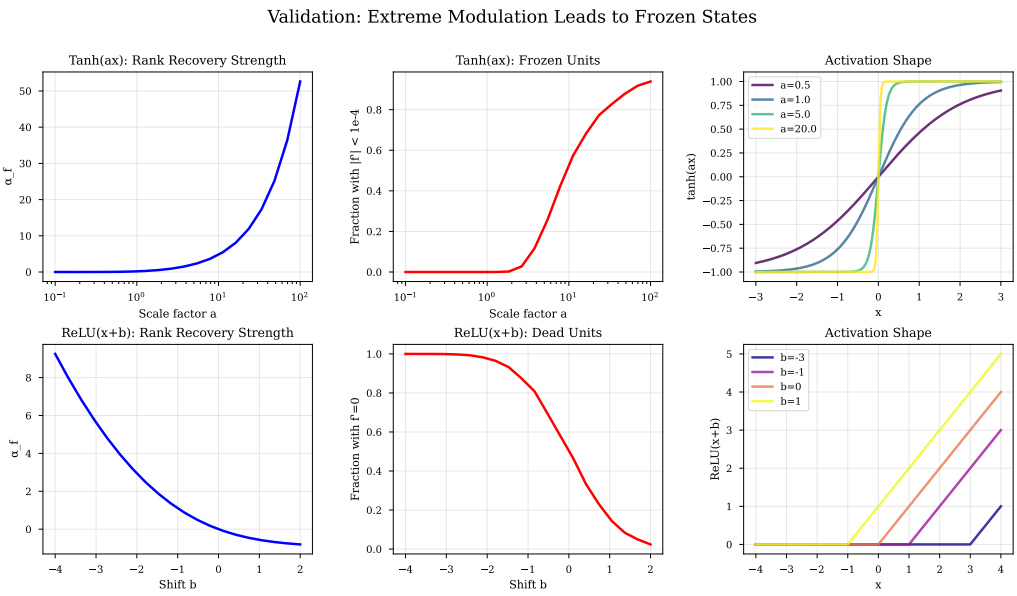

Figure A.1: Validation that extreme modulation leads to frozen states. **Top row:** Analysis of Tanh(ax) with increasing scale a. Left: Rank recovery strength α_f increases with a. Middle: Fraction of frozen units (with $|f'| < 10^{-4}$) approaches 1 as a increases. Right: Activation shapes showing saturation for large a. **Bottom row:** Analysis of ReLU(x+b) with negative shift b. Left: α_f varies with shift. Middle: Fraction of dead units increases as b becomes more negative. Right: Activation shapes showing increasing dead zones. These results substantiate that maximizing rank recovery strength drives activations into regimes with zero gradients almost everywhere.

pass formally define, we can now define our backward passes. Given a loss function $\mathcal{L}(h(V_{\text{out}}), y)$ comparing the network output $h(V_{\text{out}})$ to a target y, the back-propagation algorithm computes gradients via error signals $\delta(v)$. The error signal is defined recursively:

$$\delta(v) = \begin{cases} \partial \mathcal{L}(h(V_{\text{out}}), y)/\partial h(V_{\text{out}}), & \text{for output nodes ,} \\ \displaystyle\sum_{u \in \text{out}(v)} \delta(u)\, w(v,u)\, f'_u(z(u)), & v \notin V_{\text{out}}, \end{cases}$$

where $\text{out}(v)$ is the set of nodes receiving input from v, and f'_u is the derivative of the activation function f_u. The gradient of the loss with respect to a weight $w(u,v)$ is then given by $\partial \mathcal{L}/\partial w(u,v) = \delta(v)\, f'_v(z(v))\, h(u)$.

Network partition and base network definitions. Let $G = (V, E, w)$ be the main network. A partitioning refers to a partitioning of nodes defined as:

$$\cup_{i=1}^{k} S_i = V, \qquad S_j \cap S_i = \emptyset \text{ for all } i \neq j.$$

Given the partitioning, we define the base network $\widetilde{G} = (\widetilde{V}, \widetilde{E}, \widetilde{w})$ where each partition is a node, the edges are union of edges between two corresponding partitions, and weights are the sum total sum of edges divided by the number of rows:

$$\widetilde{V} := \{S_i : i \in [k]\} \qquad \widetilde{E} = \{(S_i, S_j) : S_i \times S_j \cap E \neq \emptyset\} \qquad \widetilde{w}_{ij} = \frac{1}{|S_i|} \sum_{u \in S, v \in S_j} w_{uv}.$$

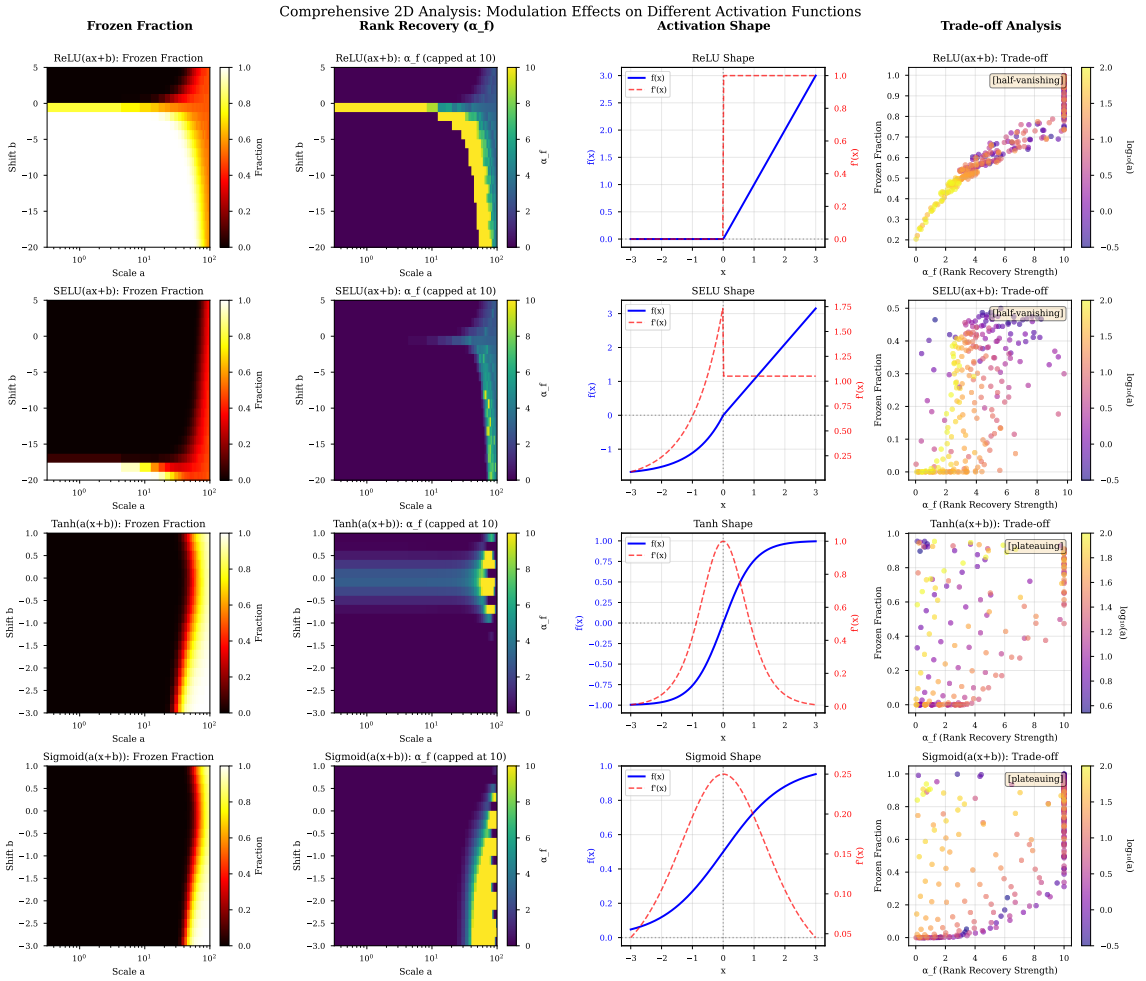

Figure A.2: Comprehensive 2D analysis of modulation effects on activation functions. Each row shows a different activation function (ReLU, SELU, Tanh, Sigmoid) with their modulation scheme. **Column 1:** Heatmaps of frozen fraction as a function of scale a and shift b. Red regions indicate parameter combinations leading to frozen/dead units. **Column 2:** Heatmaps of rank recovery strength α_f (capped at 10 for visualization). **Column 3:** Activation function shapes showing both $f(x)$ (blue) and $f'(x)$ (red dashed). **Column 4:** Trade-off analysis showing the correlation between α_f and frozen fraction, with colors indicating $\log_{10}(a)$. The analysis reveals that half-vanishing activations (ReLU, SELU) and plateauing activations (Tanh, Sigmoid) exhibit different pathways to frozen states, but all show the fundamental tension between rank recovery and maintaining gradient flow discussed in Sec. 3.

We can view the base graph as a "meta" graph, whose nodes are set of nodes, and its edges correspond to set of edges of the main graph. While the node and edge definitions are standard, the weight definition is slightly deviating from one might expect from standard quotient graph definitions, where the weights are total sum without averaging. The reason for this is more specific to our construction and is there to ensure similarity of the cloned and base networks forward and backward passes.

Definitions of Weight Manifolds Given a network partitioning $S_1, \ldots, S_k$, and the corresponding base graph $\widetilde{G} = (\widetilde{V}, \widetilde{E}, \widetilde{w})$, here are the manifold definitions:

- The Row-wise Equitable (RE) manifold consists of all cloned weight matrices w such that for every connection $(i, j) \in \widetilde{E}$ in the base network, each block $w[S_i, S_j]$ all row-sums are equal:

$$\mathcal{M}_{RE} = \left\{ w \in \mathbb{R}^{|E|} \;\middle|\; \forall (i,j) \in \widetilde{E}, \text{ and } \forall r, r' \in S_i, \text{ it holds } \sum_{u \in S_j} w_{ru} = \sum_{u \in S_j} w_{r'r} \right\}$$

  The Column-wise Equitable (CE) manifold, consists of all cloned weight matrices w such that for partitioned block $w[S_i, S_j]$, all column sums are equal:

$$\mathcal{M}_{CE} = \left\{ w \in \mathbb{R}^{|E|} \;\middle|\; \forall (i,j) \in \widetilde{E}, \text{ and } \forall c, c' \in S_j, \text{ it holds } \sum_{u \in S_i} w_{uc} = \sum_{u \in S_i} w_{uc'} \right\}$$

- The Block-wise Constant (BC) manifold consists of all cloned weight matrices w such that for every block $w[S_i, S_j]$, all its elements are equal:

$$\mathcal{M}_{BC} = \left\{ w \in \mathbb{R}^{|E|} \;\middle|\; \forall (i,j) \in \widetilde{E}, \text{ and } \forall u, u' \in S_i, \forall v, v' \in S_j, \text{ it holds } w_{uv} = w_{u'v'} \right\}$$

- Finally, we can define the family of all duplicate manifolds, that are affine sub-spaces of the parameters. For any matrix with row and column equitability, $w \in \mathcal{M}_{RE} \cap \mathcal{M}_{CE}$, they shift the block constant manifold $\mathcal{M}_D$. Formally:

$$\mathbb{M}_D = \{\mathcal{M}_D(w) \mid w \in \mathcal{M}_{RE} \cap \mathcal{M}_{CE}\}, \qquad \mathcal{M}_D(w) := \{w + T \mid T \in \mathcal{M}_{BC}\}$$

Note that all the manifolds defined above are linear or affine sub-spaces, as their constraints are all linear. There are two important facts worth mentioning that will shed more light on the upcoming theorem.

Remark A.2. *Note that the dimensionality of manifolds in the family $\mathbb{M}_D$ are given by the number of blocks in W, as opposed to number of its elements. Thus, for example if the partitioning of units forms blocks of size n, we would roughly expect $1/n^2$ fewer dimensions in $\mathbb{M}_D$ than in the original full parameter space.*

Furthermore, the following remark clarifies why we define these networks as cloned networks. Because when we are on these manifolds, the clone network units form perfect copies of the base network units.

Remark A.3. *If $W \in \mathcal{M}_{RE}$, any unit in a block $v \in S_k$, the forward activations will be identical to the corresponding base unit $h(v) = h(\tilde{v})$, where $\tilde{v}$ is the corresponding unit in the base network to block S_k. If we further assume $W \in \mathcal{M}_{RE} \cap \mathcal{M}_{CE}$, we will have a similar property for the backwards $\delta(v) = \delta(\tilde{v})$.*

Let us re-state the theorem on cloning to make this section more self-contained.

Theorem A.2 (Cloned-Unit Manifold (Re-stated))**.** *Let $G = (V, E, W)$, denote a network that is partitioned with $S_1, \ldots, S_k$. For any input and label (x, y):*

1. *If $W \in \mathcal{M}_{RE}$, then all units in the same cluster $u, v \in S_k$ have identical forward activations $h(u) = h(v)$.*
2. *If $W \in \mathcal{M}_{RE} \cap \mathcal{M}_{CE}$, then all units in the same cluster $u, v \in S_k$ have identical backward activations $\delta(u) = \delta(v)$. Furthermore, the gradients $\partial\mathcal{L}/\partial W$ will have a block-wise constant structure, such that gradients between any two units in two blocks will be equal, i.e., for any $u, u' \in S_i$ and $v, v' \in S_j$, we have $\partial\mathcal{L}/\partial W_{uv} = \partial\mathcal{L}/\partial W_{u'v'}$.*

3. *If the model weights at initialization or any point in training touch, if they lie on a manifold from the family $W \in \mathcal{M}_D$ where $\mathcal{M}_D \in \mathbb{M}_D$, given any arbitrary batches of input label pairs used to obtain subsequent model parameters $W(t)$,, any subsequent training parameter trajectory constrained to the same manifold:*

$$W(0) \in \mathcal{M}_D \implies W(t) \in \mathcal{M}_D \qquad \mathcal{M}_D \in \mathbb{M}_D, \text{ t gradient steps}$$

Proof of Theorem A.2 (Cloned-Unit Manifold). The proof will be done as a series of inductions. First, let us assume that we have sorted the units in a topological order $v_1, \ldots, v_n$ which exists because the network is a directed acyclic graph. Let us further assume that input nodes appear first in this list, and that outputs as the last edges in the list. Finally, because we assume no edges inside each block between the units, let us assume that the units in the same block are adjacent in our topological sort. Thus, for any two distinct blocks $S_i \neq S_j$, we either have all nodes in S_i before S_j or vice versa, but cannot have a mix.

Forward cloning. Row-equitability assumption implies identical forward for units in the same block. The induction hypothesis is that for all k, any preceding unit $p \leq k$, that belongs same partition $u_p, u_k \in S_i$, will have identical forward $h(u_p) = h(u_k)$. Because cloning does not apply to input units, meaning that every unit is a separate block, the hypothesis trivially holds for all input units $k = 1, \ldots, d$ where d is input dimension. Now, let us prove the induction step, assuming step k. Let $p \leq k$ correspond to a unit in the same block $u_p, u_k \in S_i$. Now, consider all the units that have incoming edges to these two units, which necessarily must appear before p. Let's consider all such units within the same block S_j. Because these units appear before k, the induction hypothesis tells us that they have identical forward. Thus, the total contribution from these units to pre-activations $z(u_p)$ and $z(u_k)$ will be proportional to sum of edge weights from units in S_j. Because of our construction of the ordering, all the units in S_j that feed into S_i must occur before them. Now, the row-equal assumption implies that the sum of weights from all these units to u_p and u_k must be equal weight sum. Thus, we have proven that pre-activation contribution from units in S_j will be identical for u_p and u_k. Because we chose S_j arbitrarily and it could have been any block, we have proven that pre-activation these units must be identical $z(u_p) = z(u_k)$. Since they also have identical activation function, they will have identical outputs $h(u_p) = h(u_k)$. This completes the induction hypothesis for forward pass cloning.

Backward cloning. We want to prove that column and row-equitability assumption implies identical backward for units in the same block. The proof strategy will be highly similar to the forward cloning case, with the key difference that our induction will be backward in our ordering, starting from latest output units and then moving in backward in the list. The induction hypothesis for step k is that, for al $q > k$, if they are in the same block $u_k, u_q \in S_i$, they will have identical backwards $\delta(u_k) = \delta(u_q)$. Because output units are not themselves cloned, the induction step holds trivially for the last output nodes. Now let us prove the induction hypothesis for k assuming that it holds for all higher steps. Now, for some arbitrary block S_j that units in S_i feed into, consider all outgoing connections from u_k, u_q to the units in this block. Because of our construction of the ordering, all the units in S_j that S_i feeds into must occur after S_j. Thus, by induction hypothesis, all these units must have identical backwards. Furthermore, from our column-equitability assumption we know that total edge weights from u_k, u_q to these units must be identical. Thus, the summation formulas in the backward of u_k and u_q are similar. Finally, since S_j was chosen arbitrarily, this summation is identical for all subsequent blocks, which implies the overal sum is also identical. To conclude the proof, note that because of row-equitability condition we already inherit the proof from the forward case, implying $f'(z(u_k)) = f'(z(u_q))$. Thus, both parts to the backward formula for u_k, u_q will be identical, which proves they have identical backwards. This finishes the induction step.

Gradient cloning. This step is a straightforward consequence of the forward and backward cloning steps, and the formula that gradient of an edge from u to v is simply $h(u)\delta(v)$. Thus, the cloning structures in forward and backward, manifest themselves as a block structure in the gradients.

Constrained training trajectory. Here, the key induction step is over the gradient steps. For step t, the induction hypothesis is that $W(t) \in \mathcal{M}_{CE} \cap \mathcal{M}_{RE}$, and that $W(t) - W(0) \in \mathcal{M}_{BC}$. This trivially holds for initial step $t = 0$. Let us prove the induction step $t + 1$ assuming that it holds for t. Suppose gradient at this step $\Delta W(t)$ is defined over the loss arbitrary number of samples $\{(x_i, y_i)\}$. Because of the induction hypothesis $W(t) \in \mathcal{M}_{CE} \cap \mathcal{M}_{RE}$, our earlier results imply that the gradients for each sample $\partial \mathcal{L}_i / \partial W(t)$, will have a block-wise constant structure $\partial \mathcal{L}_i / \partial W(t) \in \mathcal{M}_{BC}$. Thus, the sum of these gradients will also have a block-wise constant structure $\Delta W(t) := \partial \mathcal{L}_/ \partial W(t) \in \mathcal{M}_{BC}$. Because block-wise matrices are also row- and column-equitable, this implies that the new weights will inherit those $W(t + 1) = W(t) + \Delta W(t) \in \mathcal{M}_{CE} \cap \mathcal{M}_{RE}$. Finally, our parameter shift can be written as $W(t + 1) - W(0) = \Delta W(t) + W(t) - W(0)$, where $W(t) - W(0)$ is a block-wise constant matrix and thus $W(t + 1) - W(0)$ becomes sum two block-wise constant matrices, which is itself block-wise constant $W(t + 1) - W(0) \in \mathcal{M}_{BC}$. This finishes our induction step. □

A.3 Modular cloning profiles and a composition theorem

This subsection formalizes a modular extension of the cloning theorem in Theorem 2.1 (see also Appx. A.2) and proves that local, module-level cloning certificates glue to yield cloning for the entire composed architecture.

Modules, interfaces, and profiles. A *module* is a feed-forward sub-DAG $G_M = (V_M, E_M, W_M)$ together with disjoint sets of *interface nodes* I_M (inputs) and O_M (outputs). We allow internal nodes $V_M^\circ := V_M \setminus (I_M \cup O_M)$ and edges that connect interface nodes to internal nodes or to other interface nodes as permitted by the DAG. Let $\widetilde{G}_M = (\widetilde{V}_M, \widetilde{E}_M, \widetilde{W}_M)$ denote a smaller *base* module.

A *cloning profile* for M relative to $\widetilde{M}$ consists of surjections

$$\pi_M^{\text{in}} : I_M \twoheadrightarrow \widetilde{I}_M, \qquad \pi_M^{\text{out}} : O_M \twoheadrightarrow \widetilde{O}_M,$$

inducing partitions $\mathcal{P}_M^{\text{in}} = \{\, (\pi_M^{\text{in}})^{-1}(i) \;:\; i \in \widetilde{I}_M \,\}$ and $\mathcal{P}_M^{\text{out}} = \{\, (\pi_M^{\text{out}})^{-1}(o) \;:\; o \in \widetilde{O}_M \,\}$. Intuitively, all interface units in the same block are *clones* of the corresponding base port.

We say two wired modules $A \to B$ have *matching profiles* if their shared interface partitions coincide after applying the wiring map $\omega_{A\to B} : O_A \to I_B$, i.e.

$$\omega_{A\to B}\big(\mathcal{P}_A^{\text{out}}\big) = \mathcal{P}_B^{\text{in}} \quad \text{as partitions of } I_B,$$

and dually for the reversed wiring used by backpropagation. More generally, a whole network has matching profiles if this holds on every inter-module edge set.

Module-level cloning manifold. Fix a module M with profile $(\mathcal{P}_M^{\text{in}}, \mathcal{P}_M^{\text{out}})$. Extend these interface partitions to a partition of *all* nodes V_M by assigning each internal node of M to the block of its corresponding base-node in the collapsed base graph $\widetilde{G}$ defined in Appx. A.2. On M, define the (affine) *module cloning manifold*

$$\mathcal{M}_D(M) \;=\; \{\, W_M :\; W_M \in \mathcal{M}_{RE} \cap \mathcal{M}_{CE} \text{ with respect to the induced partition of } V_M \,\},$$

i.e., each inter-block weight submatrix is row- and column-equitable (block-wise constant up to redistribution), reusing the notation of Appx. A.2. This generalizes the block-constant manifold $\mathcal{M}_{BC}$ by allowing intra-block redistribution while preserving equal in/out block-sums.

Definition A.1 (Module-level cloning certificate)**.** *A module M endowed with profile $(\mathcal{P}_M^{\text{in}}, \mathcal{P}_M^{\text{out}})$ admits a* cloning certificate *if the following hold for every batch:*

(MC1) ***Forward interface preservation.*** *If inputs in the same block of $\mathcal{P}_M^{\text{in}}$ carry identical values, then for any $W_M \in \mathcal{M}_D(M)$ all outputs in the same block of $\mathcal{P}_M^{\text{out}}$ are identical (forward cloning).*

(MC2) ***Backward interface preservation.*** *If the output adjoints (backprop signals) are blockwise identical on $\mathcal{P}_M^{\text{out}}$, then for any $W_M \in \mathcal{M}_D(M)$ the input adjoints are blockwise identical on $\mathcal{P}_M^{\text{in}}$ (backward cloning).*

(MC3) ***Gradient closedness.*** *Under (MC1)–(MC2), the per-edge gradient $\partial\mathcal{L}/\partial W_M$ is block-wise constant on each inter-block submatrix, hence $\nabla\mathcal{L}(W_M)$ is tangent to $\mathcal{M}_D(M)$ and first-order parameter updates initialized on $\mathcal{M}_D(M)$ remain on $\mathcal{M}_D(M)$.*

Remark A.4 (Optimizers covered)**.** *(MC3) implies closure under any first-order optimizer whose update is a (possibly stateful) scalar multiple of the local gradient on each parameter and whose internal state is identical across clones at initialization (e.g., SGD, momentum, RMSProp, Adam with tied clone states). Weight decay that acts per-parameter independently may break exact symmetry; see Appx. A.2. Dropout violates (MC1)–(MC2) because independent masks destroy blockwise equality in the forward/backward signals.*

Lemma A.4 (Module certificate from $\mathcal{M}_{RE} \cap \mathcal{M}_{CE}$)**.** *If $W_M \in \mathcal{M}_{RE} \cap \mathcal{M}_{CE}$ for the induced partition of V_M, then M satisfies (MC1)–(MC3).*

Proof. This is the restriction of Theorem 2.1 to the subgraph G_M with its node partition: row-equitability yields identical forward values within blocks, column-equitability yields identical backward adjoints within blocks, and $d\mathcal{L}/dW_M = h\,\delta^\top$ is block-wise constant across inter-block submatrices. Tangency of the gradient to $\mathcal{M}_{RE} \cap \mathcal{M}_{CE}$ follows exactly as in Appx. A.2. □

Theorem A.3 (Composition theorem for modular cloning)**.** *Let a feed-forward network be formed by wiring modules $\{M_\ell\}_{\ell=1}^L$ with matching profiles at every interface. Suppose each M_ℓ admits a cloning certificate (Def. A.1) and that parameters are initialized on the product manifold $\prod_\ell \mathcal{M}_D(M_\ell)$. Then:*

1. ***Global forward cloning.*** *If the external inputs respect the input profile of the first modules, then all internal interfaces and the final outputs are blockwise identical according to the propagated profiles. Equivalently, the composed network is a cloned enlargement of the composed base network.*
2. ***Global backward cloning.*** *For any loss, if the final output adjoints are blockwise identical, then all internal interface adjoints and the external input adjoints are blockwise identical according to the propagated profiles.*
3. ***Persistence under training.*** *The network gradient is tangent to $\prod_\ell \mathcal{M}_D(M_\ell)$, hence any first-order parameter update that preserves (MC3) at the module level preserves the global cloning manifold and items 1–2 continue to hold at all subsequent steps.*

Proof. Forward. Order modules topologically. Assume the external inputs are blockwise identical on the first-layer profiles. Applying (MC1) to the first module yields blockwise-identical outputs on its output profile. By profile matching, these outputs equal the input profile of the next module, so (MC1) applies again. Induction over modules yields blockwise equality at every interface and at the final outputs.

Backward. Reverse the topological order. Start from blockwise-identical adjoints at the final outputs. By (MC2) for the last module, the incoming adjoints to its inputs are blockwise identical. Profile matching identifies these with the previous module's output profile, so (MC2) applies again. The inductive step propagates back to the external inputs.

Persistence. By (MC3), in each module the gradient is block-wise constant on inter-block submatrices, i.e., tangent to $\mathcal{M}_D(M_\ell)$. The product of affine manifolds is an affine manifold with tangent equal to the product of tangents, so the global gradient is tangent to $\prod_\ell \mathcal{M}_D(M_\ell)$. Thus first-order updates initialized on this product manifold remain on it, and the previous two items re-apply at every step. □

Remark A.5 (Coverage: modern architectures)**.** *The certificate (Lemma A.4) is satisfied by the standard width/channel/heads expansions used in practice:*

- ***MLPs / Linear layers:*** *Duplicate hidden units; enforce RE/CE by tiling weights with appropriate $1/$(input expansion) scaling; duplicate biases. Matches the implementation in* `clone_linear`*.*
- ***CNNs / Conv layers:*** *Duplicate channels (in/out); tile kernels with $1/$(input expansion) scaling; duplicate biases (*`clone_conv1d`*,* `clone_conv2d`*). Spatial pooling is per-channel and thus profile-preserving.*
- ***Normalization:*** *BN/LN/GN with duplicated (γ, β) and running stats per clone are profile-preserving (*`clone_normalization`*).*
- ***Activations and elementwise ops:*** *Elementwise maps are profile-preserving (*`clone_activation`*); parameter-free ops are trivially preserved (*`clone_parameter_free`*).*
- ***ResNets:*** *Residual addition preserves cloning provided both branches use the same profile; block-level expansions meet RE/CE at each addition.*
- ***Transformers/ViTs:*** *(i) Embedding/patch-projection expansions via tiling (*`clone_embedding`*); (ii) Multi-head attention via head duplication; per-head linear maps satisfy RE/CE; concatenation is a profile-preserving reshape; (iii) MLP sub-blocks as in MLPs; (iv) LayerNorm is profile-preserving. The* `CloneAwareFlatten` *operator ensures profile-preserving reshapes between conv/linear stages.*

By contrast, ***Dropout*** *with independent masks across clones breaks (MC1)–(MC2) and thus is excluded from this corollary (see also discussion in the main text).*

Remark A.6 (Minimal check-list for a new module)**.** *To certify a new module M:*

1. *Choose interface partitions $(\mathcal{P}_M^{\text{in}}, \mathcal{P}_M^{\text{out}})$ and extend them to V_M.*
2. *Verify $W_M \in \mathcal{M}_{RE} \cap \mathcal{M}_{CE}$ for the induced partition (row/column equitability per inter-block submatrix).*
3. *Conclude (MC1)–(MC3) by Lemma A.4.*
4. *Ensure adjacent modules use matching profiles at shared interfaces.*

Under these conditions, Theorem A.3 guarantees network-level cloning and its persistence under training.

Observation A.1 (Connection to the implementation)**.** *The functions* `clone_{linear,conv1d,conv2d,normalization,embedding,activation}` *and* `model_clone` *implement the RE/CE tiling and profile-preserving reshapes described above, while* `test_activation_cloning` *empirically verifies (MC1)–(MC2) layer-wise via forward/backward R^2. The* `CloneAwareFlatten` *operator is a profile-preserving connector that keeps duplicated channels adjacent, ensuring that profiles match across CNN→FC boundaries.*

A.4 Stability of LoP manifolds.

While Theorem 2.1 establishes the *existence* of LoP manifolds under exact conditions (perfect saturation, perfect cloning), in practice, these conditions might only be approximately reached during training. This leads to the question of whether near-LoP states will move back closer to the LoP manifold under gradient descent dynamics, or will they move away from it. To address this, we introduce the notion of the stability of an LoP manifold.

Definition A.2 (Stability of LoP Manifold)**.** *Let $\mathcal{M}$ be an LoP manifold and $N_\theta\mathcal{M}$ be the normal space to $\mathcal{M}$ at $\theta \in \mathcal{M}$. The stability of $\mathcal{M}$ is characterized by the Hessian $\nabla^2_\theta \mathcal{L}(\theta)$ in directions normal to $\mathcal{M}$:*

- ***Stable LoP:*** $\forall v \in N_\theta\mathcal{M} \setminus \{0\} : v^\top \nabla^2_\theta \mathcal{L}(\theta) v > 0$. *(Perturbations revert to LoP)*
- ***Unstable LoP:*** $\forall v \in N_\theta\mathcal{M} \setminus \{0\} : v^\top \nabla^2_\theta \mathcal{L}(\theta) v < 0$. *(Perturbations escape LoP)*
- ***Saddle LoP:*** $\exists v_1, v_2 \in N_\theta\mathcal{M}$ *s.t.* $v_1^\top \nabla^2_\theta \mathcal{L} v_1 > 0$ *and* $v_2^\top \nabla^2_\theta \mathcal{L} v_2 < 0$. *(Escape is direction-dependent)*

Remark A.7. *Stability in the normal space to the manifold (convexity of the loss in these directions) does not imply that the loss is convex in general (i.e., also within the manifold or in other directions). These conditions are local characterizations of the loss landscape geometry around the manifold.*

To understand the practical implications of these stability types, consider injecting a small perturbation $\Delta\theta$ that pushes the parameters θ slightly off the manifold $\mathcal{M}$. If $\mathcal{M}$ is stable, the subsequent gradient steps $-\nabla\mathcal{L}(\theta + \Delta\theta)$ will tend to project back towards $\mathcal{M}$. If $\mathcal{M}$ is unstable, these steps will tend to move further away. For a saddle LoP manifold, escape depends on the direction of the initial perturbation relative to the eigenvectors of the Hessian in the normal space. Therefore, the strongest form of LoP corresponds to a *stable* LoP manifold, as it actively resists escape. An unstable manifold is the easiest to escape. A saddle manifold presents a mixed scenario, where random perturbations may or may not escape depending on the perturbation vector being in a positively or negative space orientation.

B Empirical Appendix

This section provides comprehensive details of the experimental setups, additional empirical results, figures supporting claims made in the main text, and visualizations.

B.1 Experimental Details

This section outlines the experimental setup, methodologies, and general procedures employed for the empirical analysis of Loss of Plasticity (LoP) in neural networks.

B.1.1 Overview of Experimental Paradigms

Our investigation into LoP encompasses three primary experimental paradigms.

Continual Learning Experiments These experiments involve training models on a sequence of temporally independent tasks where data from previously learned tasks is unavailable. Tasks are typically formulated by partitioning the output classes of standard datasets, and for any given task t, the model is trained exclusively on its assigned class subset $\mathcal{C}_t$. We trained our models on Tiny ImageNet, which consists of 200 classes, by creating a sequence of 40 tasks, each containing a disjoint subset of 5 classes. Each task is trained for 500 steps, and validation is performed periodically, resulting in 20,000 total training steps. The training protocol included optional reinitialization of model output layer weights and biases are reset to zero before starting each new task to mitigate interference.

Neural Network Cloning Experiments These experiments study the effects of neuron duplication using a two-stage training protocol. Initially, a base model is trained on a target task to establish baseline performance. Subsequently, this base model is expanded by a specified factor (always fixed to two), using the cloning procedures detailed later. The expanded (cloned) model is then trained. To compare the base and cloned model, we also keep training the base model at the same time during this second phase. The results presented here are all on the CIFAR-10 dataset, and we used 20 epochs to train the base model and 500 epochs to train the cloned model. Functional equivalence post-cloning is verified by ensuring the cloned model produces

activations identical to its base, assessed via R^2 scores between corresponding layer activations. R^2 scores, computed for each layer, measure if the mean of cloned units can explain the variance of all units in that block.

Bit Flipping Experiments These experiments simulate a slowly-changing regression problem to evaluate network adaptability to gradually drifting input distributions. An illustrative benchmark for studying adaptability is the 'bit-flipping' experiment, an online regression task where the model receives an m-bit input vector x and must predict an output y. The environment is non-stationary: a subset of f input bits are designated 'flipping bits,' and at regular T-step intervals, one of these f bits is randomly inverted. The remaining $m - f$ input bits are randomly sampled at each step. The target output y is generated by a fixed (but unknown to the learning model) two-layer network, and a two-layer MLP is trained to learn this continuously drifting target function. The complexity of the learning model is typically designed to be less than that of the data-generating process, thereby creating a challenging scenario for maintaining plasticity. A target network with Linear Threshold Units (LTUs) implements $h_i = \text{LTU}(w_i^T x - \theta_i)$ and $y = w_{\text{out}}^T h + b_{\text{out}}$. A Linear Threshold Unit operation is defined as $\text{LTU}(z) = 1$ if $z \geq 0$, and 0 otherwise (a Heaviside step function). For the target network, the specific thresholds are $\theta_i = (m \cdot \beta) - S_i$, and $S_i = \sum_{j:w_{ij}<0} 1 - 0.5 \cdot w_{i,m+1}$. Input consists of m bits plus a bias bit; f of these bits are "flipping bits" changing every T time steps (one randomly selected flipping bit is inverted), while the remaining $m - f$ bits are randomly sampled each step. A two-layer MLP with a configurable activation function is trained online to learn this target.

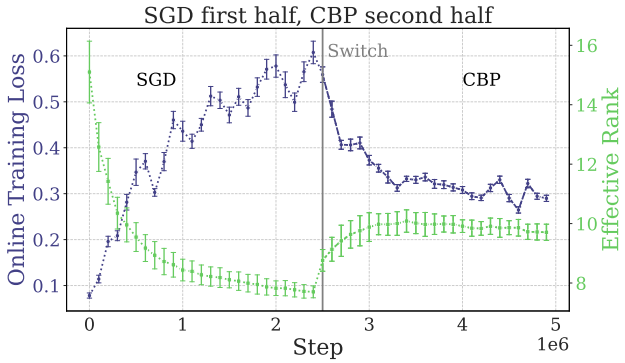

Figure B.1: Bit Flipping experiment on 5M samples, switching from SGD to CBP at 2.5M samples. Low rank structures emerge during training with standard Backpropagation (SGD), but after the switch Continual Backpropagation (CBP) is able to recover representational diversity, suggesting that CBP-like training could be effective for cloning too. (Experimental details in Appx. B).

B.1.2 Core Methodologies and Implementations

Several core methodologies underpin our experiments.

Cloning Implementation. Our cloning implementation is modular. For each architecture, we first need to decide the "free" parameter to expand. This is the feature dimension for MLP and ViT, and channels for CNN and ResNet. After creating a base and expanded model, our cloning implementation proceeds in a modular fashion. The key implementation idea that allowed this modular design is the principle that the cloning profile of inputs and outputs of different modules must be consistent. For example if inputs A and B to a module are assumed to be cloned, and if these are output by a different modules, that module must ensure this cloning. We can think of this as a matching cloning profile between connected modules. With this design in mind, for linear layers, weights and biases are replicated according to input/output expansion

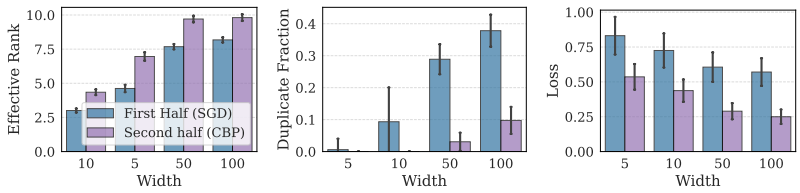

Figure B.2: Bit Flipping experiment on 5M samples, switching from SGD to CBP at 2.5M samples. For each of the two phases, we show the average over the last 100K steps. Duplicated structures (indicated by fraction of duplicate features at different layers/scales) emerge during training with standard Backpropagation (BP) but Continual Backpropagation (CBP) is able to decouple the cloned units. As the model width is increased more duplicate features emerge. The size of the data generating function is 100. (Other experimental details in Appx. B).

factors; weights connected to cloned input neurons are scaled (e.g., by $1/\alpha_{\text{in}}$ for an input duplication factor of α_{in}) to maintain activation magnitudes. Convolutional layers see similar expansion of input/output channels, with kernels tiled and appropriately scaled while preserving spatial dimensions. For normalization layer, if affine features are learned, their cloning will be a simple duplication for different cloned units. The same applies to modules such as patch embeddings, which require a simple duplication. Parameterized activations (e.g., PReLU) have their parameters correspondingly duplicated or broadcast. Any other units that does not have parameters, such as softmax layer or activations without parameter, will not require any particular treatment, because it has the potential to create cloning profiles that do not match. To fix this, we implemented a clone-aware flattening operation in CNNs ensures duplicated channels remain adjacent after flattening to preserve structure for subsequent fully-connected layers.

Noisy SGD optimizer introduces Gaussian noise $\epsilon_t \sim \mathcal{N}(0, \sigma_t^2 \|g_t\|^2 I)$ to gradients g_t, where the noise scale $\sigma_t = \sigma_0 \cdot \lambda^t$ decays over time t from an initial value σ_0. The values of σ_0 and λ are hyperparameters of the optimizer. Later, we show the effect of varying them on the cloned model dynamics.

Continual Backpropagation (CBP) , implemented in `src/utils/cbp_optimizer.py` following the Generate-and-Test framework, aims to maintain plasticity by selectively replacing low-utility neurons. Utility tracking involves measures like *Contribution Utility* ($(u^{(t)}_{\text{contrib}}))_i = |h^{(t)}_i| \cdot |\bar{w}_{\text{out},i}|$) and *Adaptable Contribution* ($(u^{(t)}_{\text{adapt}})_i = \frac{|h^{(t)}_i - \bar{h}^{(t)}_i| \cdot |\bar{w}_{\text{out},i}|}{|\bar{w}_{\text{in},i}|}$), where $h^{(t)}_i$ is activation, $\bar{h}^{(t)}_i$ is its running average, and $\bar{w}$ terms are mean weight magnitudes. Instantaneous utilities are smoothed using an exponential moving average (ρ is decay rate, $a^{(t)}_i$ is neuron age): $u^{(t)}_i = \rho u^{(t-1)}_i + (1-\rho)\tilde{u}^{(t)}_i$, with a bias-corrected version $\hat{u}^{(t)}_i = u^{(t)}_i / (1 - \rho^{a^{(t)}_i})$. Neuron replacement occurs for eligible mature neurons ($a_i > \tau_{\text{maturity}}$) with the lowest utility (a fraction r_{replace} of layer neurons N_L). Selected neurons are reinitialized (incoming weights via Kaiming, outgoing to zero, utility/age reset). A bias correction ($b_{\text{next}} \leftarrow b_{\text{next}} + W_{\text{out}}[:, i] \cdot \bar{h}_i$) is applied to the subsequent layer.

Metrics for Analysis Our comprehensive metric suite quantifies various aspects of network behavior and plasticity loss. *Single and pair feature metrics* include the fraction of "dead" neurons, identified when $\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[|H_{ij}| < 10^{-7}] > \tau_{\text{dead}}$ for neuron j across N samples, with $\tau_{\text{dead}} = 0.95$. "Duplicate" neurons are detected through cosine similarity patterns, with neurons j, k are considered duplicates if $\tilde{H}_j^T \tilde{H}_k > \tau_{\text{corr}} = 0.95$, where activations are normalized by feature $\tilde{H}_j = H_{\cdot,j} / \|H_{\cdot,j}\|_2$. "Saturated" neurons are identified when the ratio of gradient magnitude to mean activation magnitude $|G_{ij}| / \max(\mu_j, \epsilon)$ falls below $\tau_{\text{sat}} = 10^{-4}$ for more than $p_{\text{sat}} = 99\%$ of samples in a batch. *Representation diversity metrics*

include effective rank, computed as $\exp(-\sum_i p_i \log p_i)$ where $p_i = \sigma_i / \sum_j \sigma_j$ are normalized singular values from the activation matrix SVD; stable rank, calculated as $\|\tilde{H}\|_F^4 / \text{tr}((\tilde{H}^T \tilde{H})^2)$ for mean-centered activations $\tilde{H}$; *Cloning quality* is assessed by R^2 scores between base and cloned model activations, computed as $R^2 = 1 - \text{Var(residuals)}/\text{Var(total)}$ where the predictor is the mean of N cloned units and we measure explained variance across individual units relative to the total variance in that layer. This is done for both forward and backward activations across all layers, and numbers presented here are averages across all layers and both forward and backwards for the fixed batch that we are measuring the metrics. We also keep tracking all metrics for both base and cloned model after training to provide a comparison between the two.

B.1.3 General Setup and Procedures

Model Architectures include Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), ResNets, and Vision Transformers (ViTs), with configurations (depth, width, activations, normalization layer, dropout). The default configurations are as follows: Our Multi-Layer Perceptron (MLP) consists of 5 hidden layers with 128 units each, employing ReLU activations, batch normalization applied before activation, and 20% dropout. The Convolutional Neural Network (CNN) architecture comprises 3 convolutional layers with $[64, 128, 256]$ channels respectively, using 3×3 kernels with stride 1 and padding 1, followed by 2×2 max pooling operations. The convolutional features are processed by a single fully connected layer with 512 units, with ReLU activations, batch normalization, and 10% dropout throughout. For ResNet, we implement a ResNet-18 variant with $[2, 2, 2, 2]$ residual blocks per stage, starting with 64 base channels that double at each stage, using ReLU activations, batch normalization, and 10% dropout. The Vision Transformer (ViT) architecture divides input images into 8×8 pixel patches, which are projected to 384-dimensional embeddings and processed through 6 transformer layers with 6 attention heads each. The ViT employs an MLP ratio of 4.0 (yielding hidden dimensions of 1536), GELU activations, layer normalization, and 10% dropout for both general operations and attention mechanisms. All normalization layers include learnable affine parameters (γ, β), unless stated otherwise, and bias terms are enabled where applicable. Default hyperparameter configurations for each architecture can be adjusted per experiment as described in the experimental setup.

Datasets and Preprocessing involve standard image classification benchmarks: MNIST (28×28 grayscale), CIFAR-10 and CIFAR-100 (32×32 RGB with standard augmentations like random crops and flips), and Tiny ImageNet (64×64 RGB). Standard train/test splits are used. For all the figures and results reported here, we used tiny ImageNet dataset for continual learning experiments, while for cloning, CIFAR-10 was used.

Training Configuration involves optimizers like Adam or SGD without momentum and no weight decay with otherwise parameters in torch. The learning rates for the continual experiments where set to 0.001 using Adam for all architectures except for Vision Transformer, which was set to 0.0001. For cloning experiments with dropout, we varied the learning rate on a grid $0.01, 0.001, 0.0001$.

Experimental Control is maintained through comprehensive random seeding, which controls the randomness across all relevant libraries (Python, NumPy, PyTorch) and CuDNN deterministic mode. We used 5 seeds for all experiments to calculate confidence intervals. Experiments utilize GPUs when available, falling back to CPUs otherwise. Metrics are typically computed at fixed epoch intervals (e.g., every 5 epochs), often on consistent fixed data batches for reproducibility. Computationally intensive metrics like SVD may use subsampling of the features or samples to make them less expensive.

Computational Resources. For the continual learning and cloning experiments, our experimental grid consisted of approximately 2,000 individual runs (counting each random seed separately). These experiments were executed on a cluster of NVIDIA A100 GPUs, utilizing a heterogeneous mix of 40GB and 80GB

memory variants. The total computational cost for these experiments was approximately 10,000 GPU-hours. The bit flipping experiments and additional theory validation experiments were conducted on a more diverse set of hardware, utilizing lower-end computational nodes equipped with NVIDIA RTX 3090, V100, and RTX 2080 GPUs. This heterogeneous setup was sufficient for these less computationally intensive experiments, and the overall compute amounted to under 100 GPU-hours on these nodes. The theoretical validation figures and numerical simulations presented in the theory appendix (Appendix A.1.1) were generated on a MacBook using CPU computation only.

Figures details. Unless stated otherwise, all our figures report standard deviations over 5 experiment randomization, by the use of a different seed. Additionally, to reduce the number of points in the plot, in Figs. 3.1, 3.2, 4.1, B.1 and B.2 we plot the average over time windows of 1000 steps.

B.2 Additional Figures and Empirical Substantiation

This subsection includes placeholder figures for concepts discussed in the main text, for which specific existing figures were not available or suitable for direct inclusion in the main body.

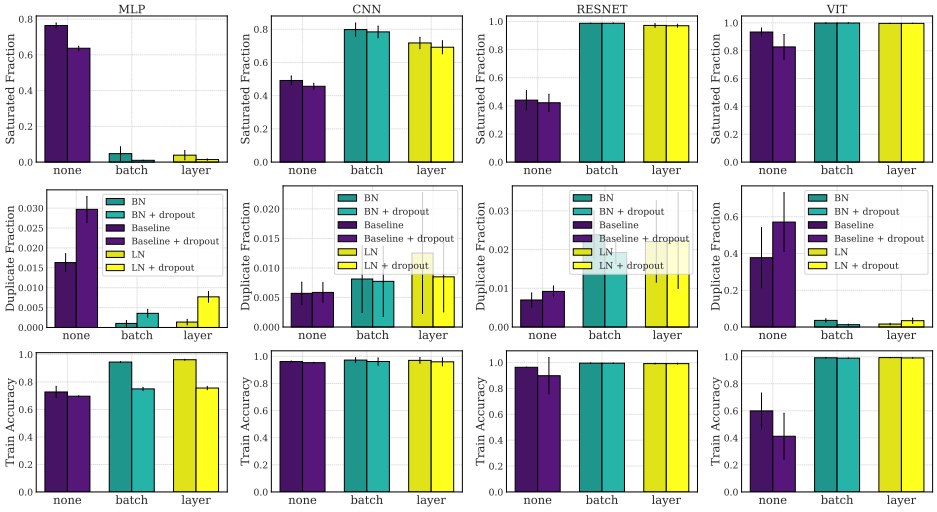

Figure B.3: Normalization reduces the number of dead/saturated units (top row) and duplicated units (middle row), and its impact on training accuracy (bottom row) across different architectures. The training accuracy displayed is calculated as the average online accuracy over the entire training length. These results highlight the role of normalization in mitigating LoP symptoms.

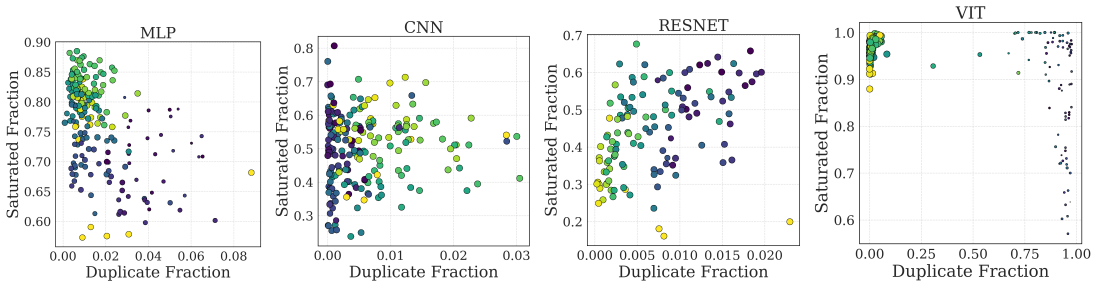

Figure B.4: Evolution of duplicate/dead unit fractions and training accuracy. The colors correspond to training steps (lighter is earlier) and the points size to the Training Accuracy (bigger is higher). This figure illustrates the correlation between the increase in LoP symptoms (duplicate/dead units) and training dynamics.

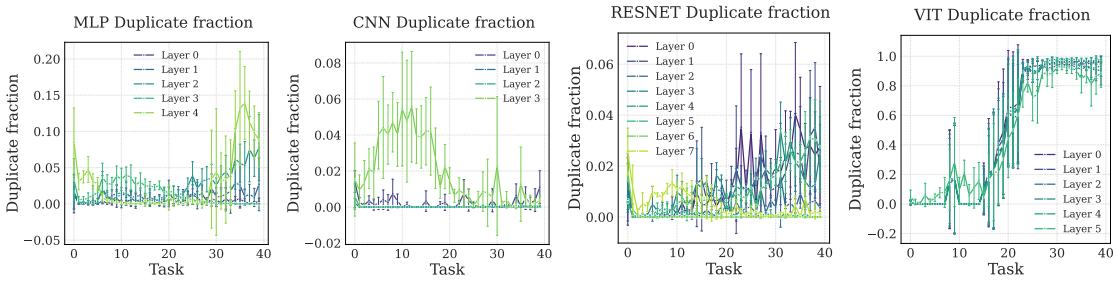

Figure B.5: Emergence of duplicate units layer-wise during training without normalization and no dropout. This figure shows the increasing fraction of duplicate units as training progresses, a symptom of LoP.

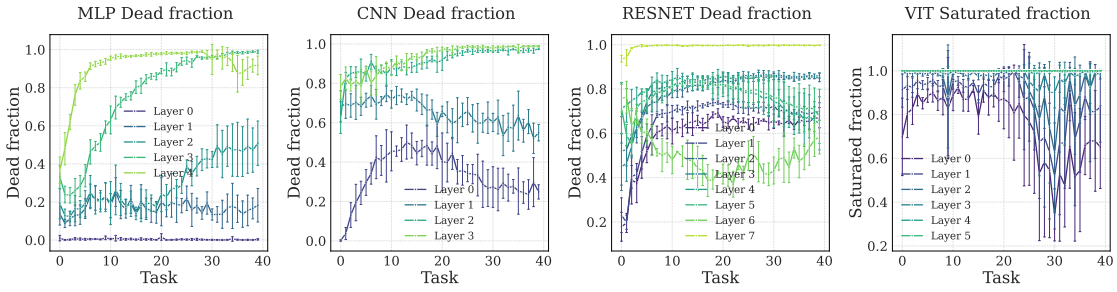

Figure B.6: Emergence of dead or saturated units layer-wise during training without normalization and no dropout. This figure shows the increasing fraction of dead units as training progresses, a symptom of LoP.

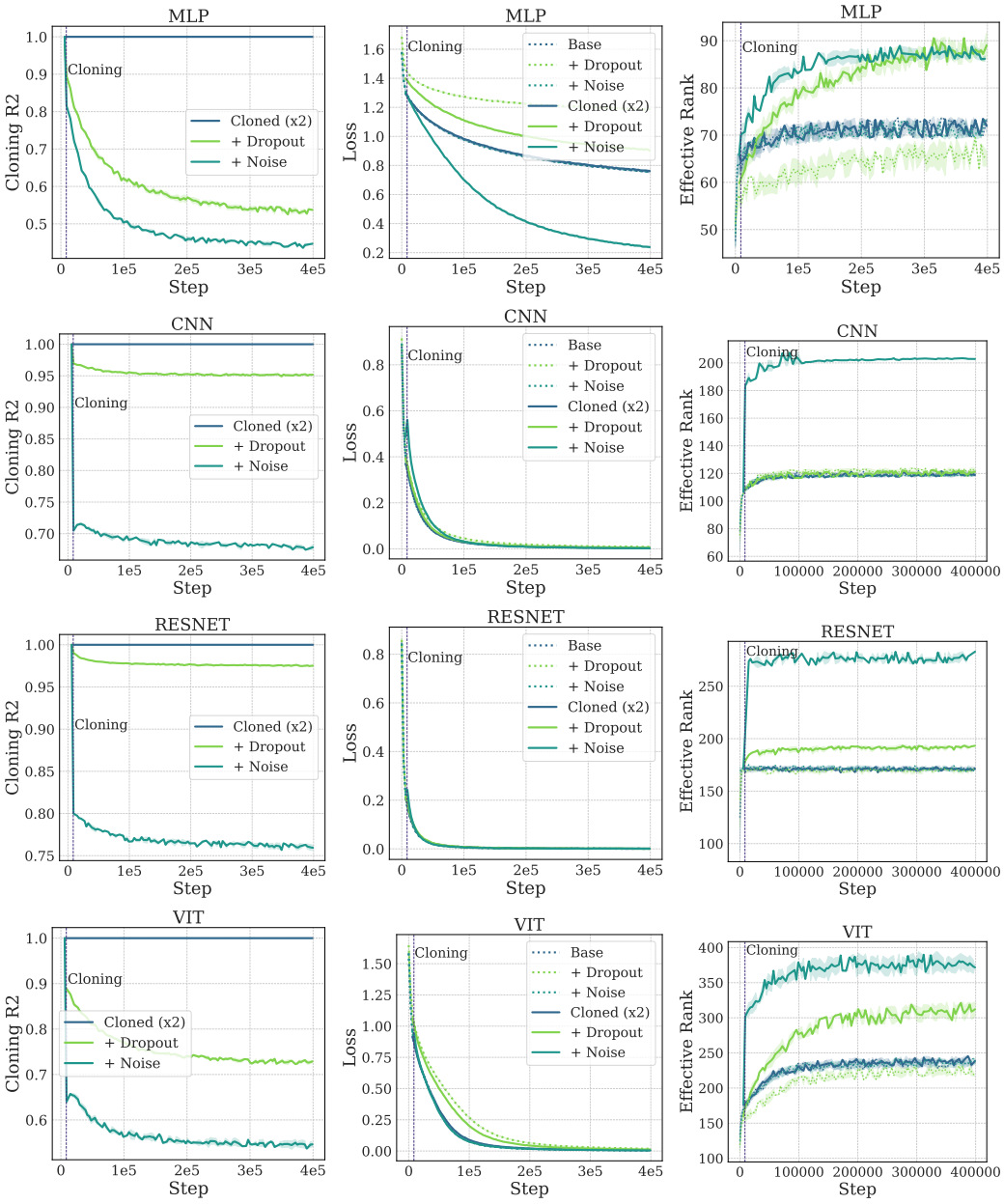

Figure B.7: Cloning experiments across architectures. Configurations details: SGD with LR=0.01, Noisy SGD with $\sigma = 0.01$ and $\lambda = 0.999$, and Dropout with probability 0.1. Normalization used: Batch Norm for all architectures, except ViTs, where we use Layer Norm.

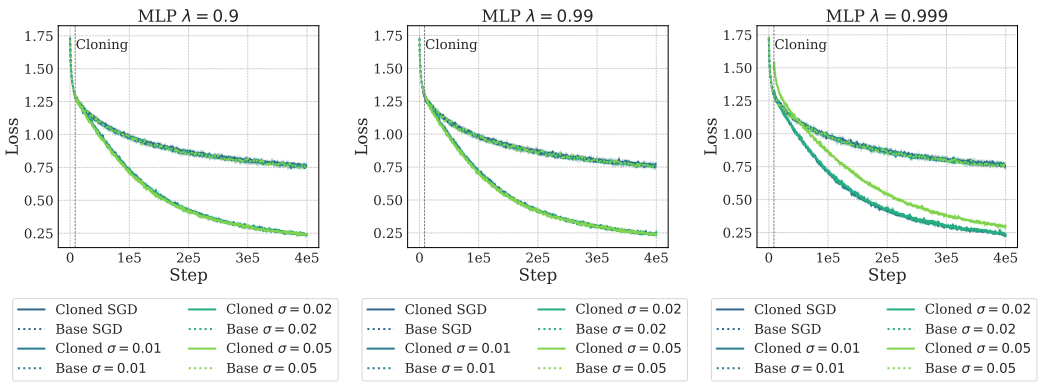

Figure B.8: Effect of noise scale parameter σ in Noisy SGD for the Cloning MLP Experiments.

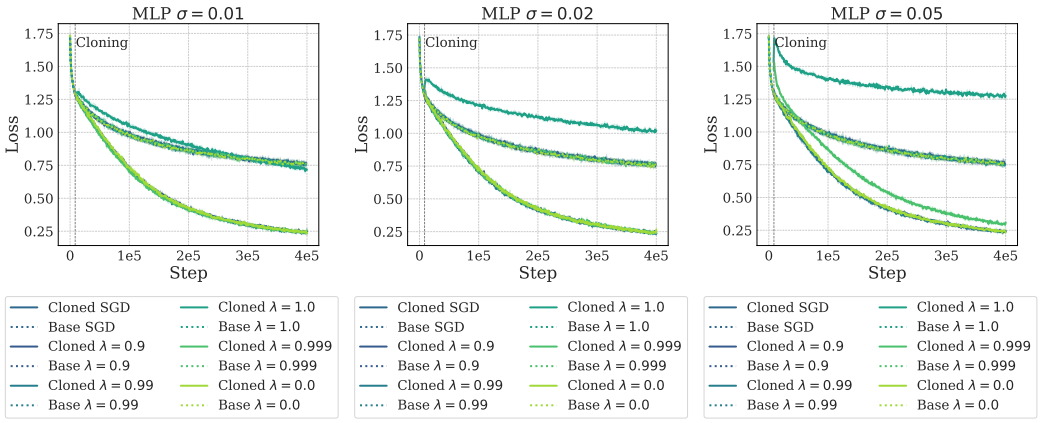

Figure B.9: Effect of noise decay parameter λ in Noisy SGD for the Cloning MLP Experiments.

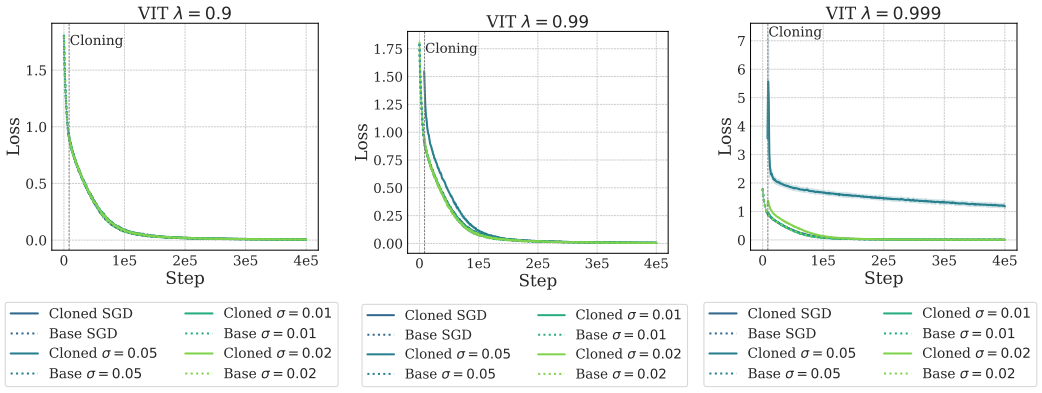

Figure B.10: Effect of noise scale parameter σ in Noisy SGD for the Cloning ViT Experiments.

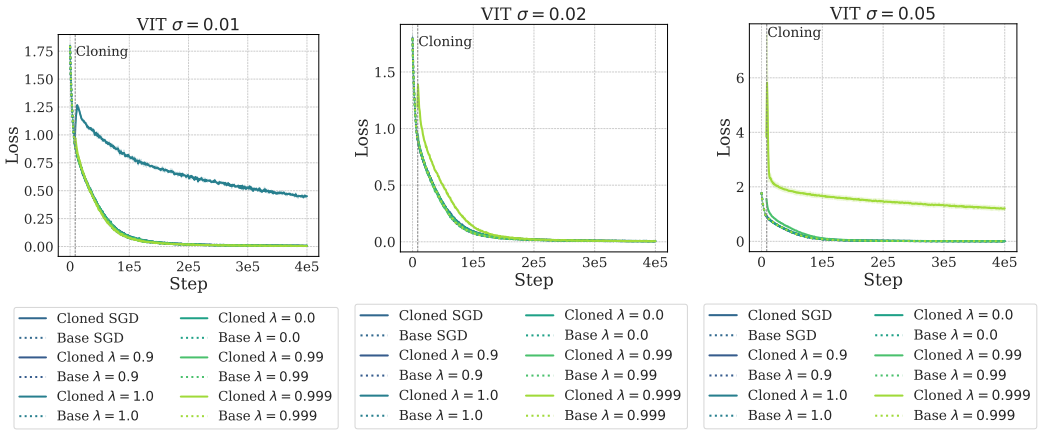

Figure B.11: Effect of noise decay parameter λ in Noisy SGD for the Cloning ViT Experiments.

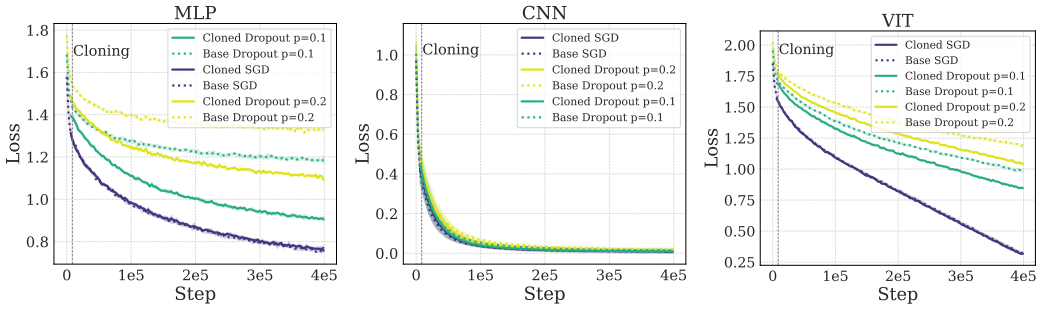

Figure B.12: Effect of dropout probability parameter for the Cloning MLP, CNN and ViT Experiments. Batch norm is used for the MLP and CNN models, and Layer norm for the ViT model.

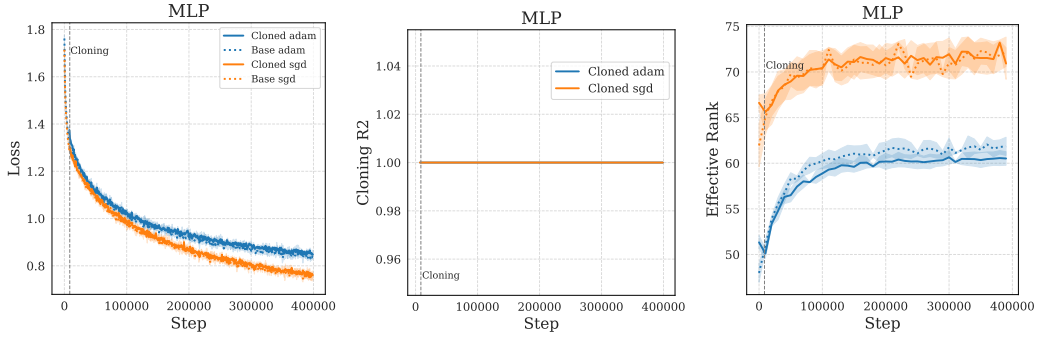

Figure B.13: Differences between SGD and Adam optimizers in the MLP Experiments. Like SGD, Adam cannot escape the base sub-manifold, although the dynamics are different.

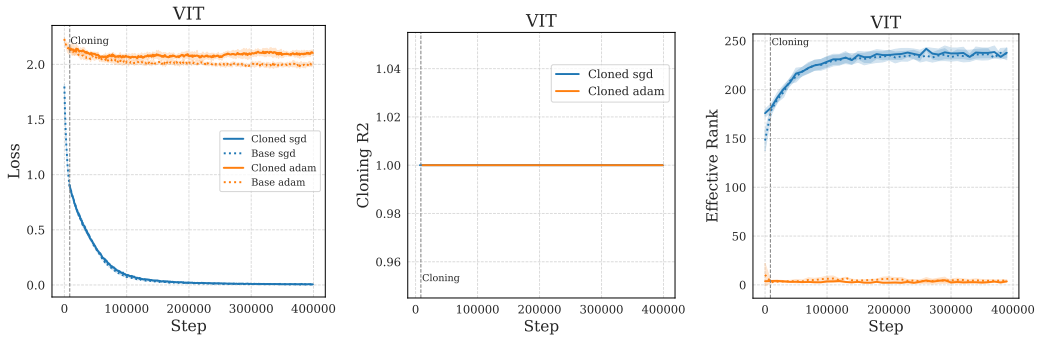

Figure B.14: Differences between SGD and Adam optimizers in the ViT Experiments. Like SGD, Adam cannot escape the base sub-manifold, although the dynamics are different.