Where You Place the Norm Matters: From Prejudiced to Neutral Initializations

E. Francazi
Physics Department
EPFL
emanuele.francazi@epfl.ch

F. Pinto
Department of Statistics
Sapienza, University of Rome
pinto.1871045@studenti.uniroma1.it

A. Lucchi
Department of Mathematics and Computer Science
University of Basel
aurelien.lucchi@unibas.ch

M. Baity-Jesi
SIAM Department
Eawag (ETH)
marco.baityjesi@eawag.ch

Abstract

Normalization layers were introduced to stabilize and accelerate training, yet their influence is critical already at initialization, where they shape signal propagation and output statistics before parameters adapt to data. In practice, both *which* normalization to use and *where* to place it are often chosen heuristically, despite the fact that these decisions can qualitatively alter a model's behavior. We provide a theoretical characterization of how normalization *choice* and *placement* (Pre-Norm vs. Post-Norm) determine the distribution of class predictions at initialization, ranging from unbiased (Neutral) to highly concentrated (Prejudiced) regimes. We show that these architectural decisions induce systematic shifts in the initial prediction regime, thereby modulating subsequent learning dynamics. By linking normalization design directly to prediction statistics at initialization, our results offer principled guidance for more controlled and interpretable network design, including clarifying how widely used choices such as BatchNorm vs. LayerNorm and Pre-Norm vs. Post-Norm shape behavior from the outset of training.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

1 INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable success across domains such as vision (Zoph et al., 2018), language (Chen and Wu, 2017), and decision-making (Silver et al., 2017), driven by architectural innovations like convolutions (LeCun et al., 1989), residual connections (He et al., 2016), and attention mechanisms (Vaswani et al., 2017).

Among these, normalization layers represent a particularly impactful development. They were introduced to stabilize signal propagation and improve trainability, and this stabilizing role is especially consequential at initialization, when the network's outputs, and therefore its predictions, are entirely determined by architectural choices and random weights. Despite their widespread empirical success, however, normalization layers are still often adopted through heuristic design patterns rather than grounded in a clear theoretical understanding of their implications. Two decisions are routinely made by experimentation: *which* normalization to use (e.g., BatchNorm versus alternatives such as LayerNorm or RMSNorm) and *where* to place it within a block (before or after the activation). These choices are ubiquitous in practice. For example, original Transformer architectures commonly applied LayerNorm in a post-activation/post-sublayer fashion, whereas some newer large language model architectures (e.g., LLaMA) adopt RMSNorm in a pre-activation/pre-sublayer configuration. Such shifts have largely been driven by empirical tuning, even though they can induce qualitatively different behaviors at initialization.

Understanding initialization behavior is particularly crucial in light of recent findings showing that architectural design choices can profoundly affect the behavior

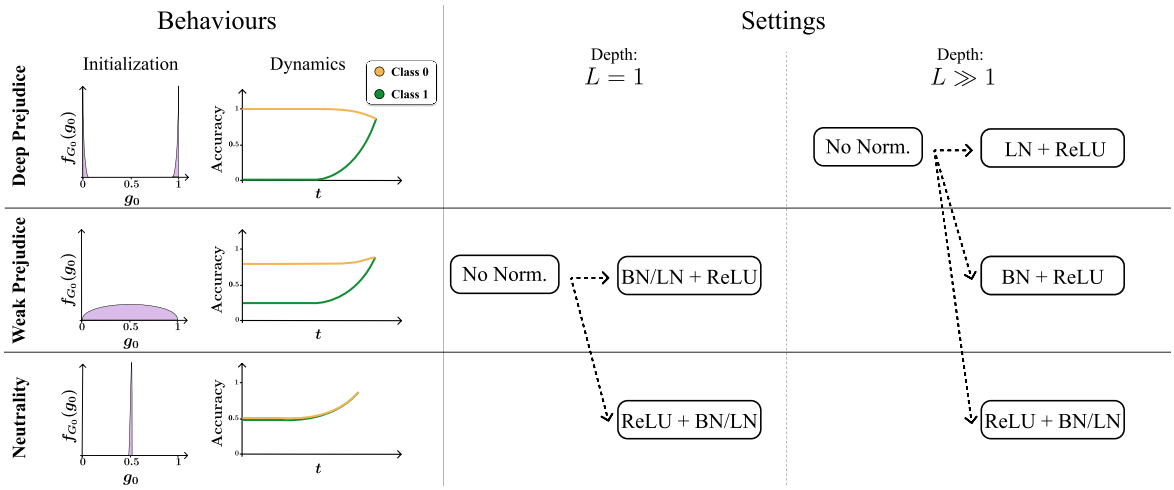

Figure 1: Diagram summarizing the effects of normalization type and placement on IGB. The left part of the diagram introduces three distinct predictive behaviors at initialization—ranging from neutral to deeply prejudiced. Each regime is illustrated through its characteristic initialization state, represented by the distribution $f_{G_0}(g_0)$, and its typical class-wise training dynamics (assuming class 0 as the initially favored class). On the right, all analyzed normalization configurations are grouped according to the behavioral regime they induce, comparing them to the baseline architecture without normalization ("No Norm.", as analyzed in Francazi et al. (2024)). Variants with LN and BN—examined in our work—are connected to the baseline by dashed arrows.

of untrained neural networks. Specifically, these choices have been shown to shape initial logit statistics, leading to qualitatively different prediction states (Francazi et al., 2024). These initial states can be intuitively categorized into two types: **neutrality**, characterized by an unbiased initial condition where predictions are roughly equally distributed among classes; and **prejudice**, where predictions initially concentrate predominantly on one or a subset of classes. These initial conditions have significant practical implications, as they strongly affect subsequent training dynamics, including convergence speed, training stability, and fairness across classes (Bassi et al., 2026). In particular, when hyperparameter tuning relies on early training trajectories, as is common under finite compute budgets or in few-shot regimes, initialization-induced biases can steer selection toward configurations that perform well early, yet predominantly on a subset of classes. A deeper understanding of these effects is therefore essential, not only for advancing theory but also for making architectural design choices more controlled and interpretable.

In this work, we tackle the broad question: **How do normalization layers influence the initial prediction bias in DNNs?** To build intuition, Fig. 1 visually summarizes the range of initialization behaviors explored in this work, and how they relate to different normalization configurations. It previews the core theoretical and empirical findings developed throughout the paper.

Our theoretical analysis characterizes how normalization layers affect the initial state by altering the distribution $f_{G_c}(g_c)$ of the fraction of data-points predicted as class c by the untrained model. [1] We then examine the role of normalization placement relative to activation functions. Our results indicate that placing normalization layers after activations consistently promotes a neutral initialization state, mitigating initial prejudices. Conversely, the widely used heuristic of placing normalization before activations frequently results in prejudiced initialization regimes.

Our empirical experiments support these theoretical predictions. Simulations clearly demonstrate the substantial influence of these initial prediction regimes on training dynamics, particularly regarding convergence speed and fairness among classes. Additionally, experiments on widely used architectures (such as MLP-Mixer, ResNet) suggest that re-evaluating conventional normalization placement in light of initialization behavior can lead to meaningful gains in realistic scenarios. By bridging architectural design with rigorous theoretical insights into initial prediction biases, our study advances theoretical understanding and provides practical design guidelines. In particular, it clarifies how everyday normalization decisions, such as those distinguishing Transformer-style and LLaMA-style designs, map to distinct and predictable initialization regimes. Because these regimes shape early learning dynamics, our framework provides a principled alternative to purely heuristic trial-and-error design.

[1] In the plots, we use $c = 0$ as the reference class.

2 BACKGROUND AND FRAMEWORK

In this section, we introduce the main notation and summarize the key concepts from prior work that are relevant to our analysis. For a more detailed discussion, we refer the reader to App. A and App. B.

3 BACKGROUND AND FRAMEWORK

In this section, we introduce the main notation and summarize the key concepts from prior work that are relevant to our analysis. For a more detailed discussion, we refer the reader to App. A and App. B.

Initial State: Trainability And Fairness. The study of the initial state of DNNs has gained increasing attention in recent years, driven by a growing recognition of its impact on the subsequent training dynamics (He et al., 2015; Mishkin and Matas, 2015; Boulila et al., 2024). A first line of research has focused on how initialization choices affect model trainability, with particular emphasis on the propagation of signals and gradients through depth. In this context, mean field theory has provided a powerful framework to characterize initialization regimes, linking the correlation behaviour of forward signal propagation to gradient stability at initialization (Schoenholz et al., 2016; Xiao et al., 2018; Noci et al., 2022). A central insight is the existence of two phases—*ordered* and *chaotic*—distinguished by the correlation between output signals for different inputs as they propagate through layers. Only at the critical boundary between these phases, known as the *edge of chaos* (EOC), gradients are stable ensuring trainability.
A complementary line of research has more recently examined how architectural design shapes the network's predictions at initialization. In particular, certain design patterns can induce a systematic predictive bias toward specific classes—a phenomenon termed *Initial Guessing Bias* (IGB) (Francazi et al., 2024). While previous mean field approaches analyze how a fixed input propagates across layers, IGB examines how a randomly initialized DNN transforms the entire dataset distribution as it propagates through the network. Bias intuitively arises because the hidden layer representations of inputs drift away from the center of the activation space, causing the decision boundary—initialized near the origin—to no longer intersect the region occupied by these representations. As a result, the decision boundary initially assigns most or all datapoints to the same side, yielding prejudiced class predictions. We provide a more detailed intuitive discussion of this phenomenon in App. C.1. Given an architecture $\mathcal{A}$ and a pre-processed dataset $\psi(\chi)$, a natural and informative measure of this effect is the *variance ratio* $\gamma(\mathcal{A}, \psi(\chi))$, introduced by Francazi et al. (2024) as a proxy for the level of IGB. It compares the typical value of centers of the node output distributions to the variability of each node across data inputs. Large values of $\gamma(\mathcal{A}, \psi(\chi))$ indicate that the output representations are consistently displaced from the origin, causing the decision boundary—initially centered—to miss the data cloud and produce prejudiced predictions. In contrast, small values of $\gamma(\mathcal{A}, \psi(\chi))$ correspond to output representations that remain near the origin, allowing the decision boundary to intersect the data cloud and yield more balanced predictions. For a formal definition of $\gamma(\mathcal{A}, \psi(\chi))$ see App. C.
Notably, Bassi et al. (2026) established a connection between IGB and the gradient stability analysis of mean field theory (Schoenholz et al., 2016), showing how prejudiced initial outputs relate to conditions for stable training. Interestingly, in certain architectures without normalization layers, IGB and gradient stability can coexist at the EOC: despite an initial prejudiced state, the prejudice is transient and gets reabsorbed during the early stages of training. However, this favorable coexistence breaks down in architectures with Batch Normalization (BN). As shown in Yang et al. (2019), in BN networks, tuning the initialization parameters alone does not suffice to reach the EOC, suggesting that alternative architectural adjustments are needed to ensure stable learning while avoiding the risk of persistent prejudice in the predictions.

Normalization. Bassi et al. (2026) connected prejudiced initial states to increased signal correlations at initialization, a quantity empirically linked to convergence time (Lubana et al., 2021). Normalization can reduce such correlations, but different methods (*e.g.*, BN vs. LN) affect deep networks differently (Lubana et al., 2021).
A related distinction appears in Transformer residual blocks: *Post-Norm* applies normalization after the sublayer and residual addition, whereas *Pre-Norm* normalizes the sublayer input. This choice has been studied mainly through trainability: Post-Norm can require careful warm-up, while Pre-Norm improves gradient flow and stabilizes deep training (Xiong et al., 2020; Nguyen and Salazar, 2019; Liu et al., 2020). Recent work refines this picture, showing that normalization placement also affects depth utilization, variance growth, and gradient allocation across layers (Li et al., 2024; Sun et al., 2025). RMSNorm variants further remove mean-centering while retaining similar practical effects in pre-normalized Transformers (Zhang and Sennrich, 2019; Jiang et al., 2023).
These works show that normalization placement shapes signal propagation and optimization, but they do not

characterize how *normalization type and placement jointly* affect initial output distributions and class predictions. The IGB framework (Francazi et al., 2024) describes the initialization-time predictive state at the *per-class* level, while remaining linked to learning stability (Bassi et al., 2026); however, existing IGB theory mainly treats architectures *without* normalization layers.

In this paper, we study how BN and LN shape predictive behavior before training.[2] We show that normalization systematically changes prediction statistics, with effects visible in class-specific performance rather than only global metrics. In particular, even in the full-batch limit BN does *not* reduce to LN: under IGB, hidden nodes within a layer are not identically distributed at fixed initialization, so averaging across samples at a node (BN) differs from averaging across nodes for a sample (LN), yielding distinct initial prediction statistics (see Fig. 3). Finally, since the layer-wise mean estimator in LN vanishes at initialization, LN is formally equivalent to RMSNorm in this regime, so our LN results extend directly to RMSNorm.

Setting And Main Notation. Our theory focuses on the most fundamental building block in deep learning, the Multi-Layer Perceptron (MLP) (Rosenblatt, 1958). MLP components are prevalent in successful modern architectures, including convolutional networks and transformers (Krizhevsky et al., 2012; Vaswani et al., 2017). We consider MLPs with Rectified Linear Unit (ReLU) activation functions processing a dataset composed of N inputs $\{\mathbf{X}_a^{(0)}\}_{a=1}^N$ through L hidden layers. When normalization is included, we analyze both pre-activation (Pre-Norm) and post-activation (Post-Norm) configurations; we denote these respectively as *Norm. + Activation* (e.g., Norm. + ReLU) and *Activation + Norm.* (e.g., ReLU + Norm.). We focus on ReLU activations (Nair and Hinton, 2010; Krizhevsky et al., 2012) as they constitute one of the most widespread choice in modern architectures and represent a prototype of activation that lead to prejudiced states (Francazi et al., 2024). For the sake of clarity, we primarily focus on binary classification (so $N_C = 2$ classes). Our findings can be straightforwardly extended to multiclass classification, through the procedure outlined in Francazi et al. (2024).

For our theory, we adopt the following notation and statistical assumptions:

- Datapoints are sampled as independent Gaussian random variables: $\mathrm{X}_{a;i}^{(0)} \sim \mathcal{N}(0,1)$. Here a indexes the sample and i the input coordinate (e.g., a pixel).
- Weights are independently initialized as Gaussian random variables: $W_{ij}^{(l)} \sim \mathcal{N}\left(0, \frac{\sigma_w^2}{n^{(l-1)}}\right)$, with biases set to zero. Here, $n^{(l)}$ denotes the number of nodes in the l-th hidden layer, and σ_w^2 is a layer-independent constant, consistent with standard initialization practices.

The choice of random, unstructured Gaussian inputs aligns with common practice in theoretical studies (Pennington and Worah, 2018; Koehler et al., 2021; Loureiro et al., 2021; Mignacco et al., 2020), simplifying analyses and removing confounding effects introduced by structured or correlated data. Importantly, this approach allows us to isolate and rigorously quantify class prejudice inherent to network architecture and initialization choices, free from data-induced effects.

Similarly, initializing bias parameters to zero is standard in both theoretical and practical settings (Glorot and Bengio, 2010; He et al., 2015), fostering symmetry and simplifying interpretation. Extending our analysis to nonzero biases is straightforward, as shown in Bassi et al. (2026).

Finally, although our main analysis adopts the popular Gaussian (Kaiming) initialization (He et al., 2015), our theoretical insights naturally extend to other initialization schemes, provided they preserve independence and scale appropriately with layer size (Francazi et al., 2024).

Given a DNN with a set of weights $\mathcal{W}$, a key quantity in our analysis is the fraction of datapoints guessed as class c, denoted by $G_c(\mathcal{W})$. Because the weights are randomly initialized, $G_c(\mathcal{W})$ is itself a random variable across the ensemble of initializations. We denote by $f_{G_0}(g_0)$ its probability density function (p.d.f.) over this ensemble.[3]

While we often focus on class 0 for clarity, the analysis is symmetric under label permutation. Accordingly, the distribution $f_{G_0}(g_0)$ is symmetric across random initializations: in a prejudiced state, the model strongly favors one class, but which class is favored varies randomly from one initialization to another.

To avoid averaging out important class-wise effects in our empirical analysis, especially when reporting per-class performance across multiple runs, we adopt a consistent labeling convention: in MLP experiments, we relabel class 0 to denote the most-predicted class at initialization. This ensures clarity and comparability of per-class statistics, particularly in regimes of deep prejudice.

[2] In our framework, other normalizations such as Instance Norm or Group Norm are trivial extensions of the discussion on LN.

[3] We denote by $f_X(x)$ the probability density function (p.d.f.) of a random variable X.

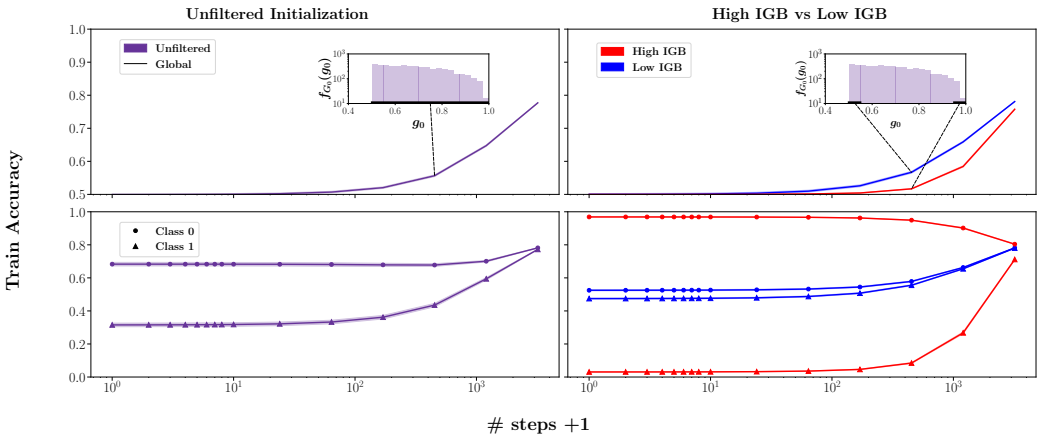

Figure 2: Training accuracy dynamics measured for the whole dataset (global, top) and for single-class performance (bottom). Curves are averaged over multiple simulations. **Left**: unfiltered initializations, randomly sampled. **Right**: simulations filtered based on G_0 to isolate neutral and prejudiced initializations. Curves are averaged over multiple runs; in each run, class 0 corresponds to the initially dominant class to avoid averaging out class-specific effects. Filtering enables comparison of the distinct convergence behaviors associated with each initialization regime. Model: `SHLP`, Data: `GB` (see App. G for more details).

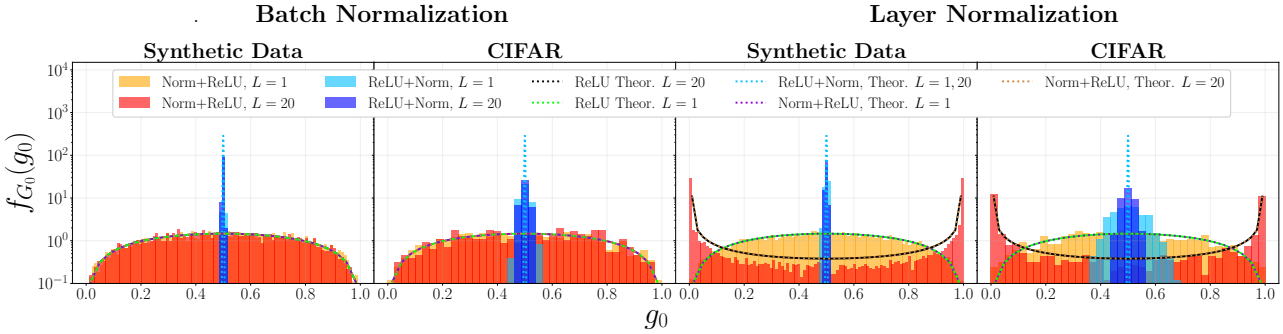

Figure 3: Distribution of the initial guessing bias $f_{G_0}(g_0)$ under different normalization settings and depths. **Left subplot:** BN; **Right subplot:** LN. For each normalization scheme, we report the distribution of G_0 when normalization is applied before or after the activation function. Within each subplot, we compare results from single hidden layer and deep MLP architectures. Each plot displays the empirical distribution for networks initialized on random unstructured inputs (left), the corresponding theoretical prediction, and the empirical distribution on structured real data (CIFAR10) (right). Post-activation normalization (ReLU + Norm.) consistently concentrates the distribution around 0.5, whereas pre-activation normalization (Norm. + ReLU) preserves or amplifies the prejudice depending on architecture depth and normalization type.

4 NOT ALL INITIALIZATIONS ARE EQUAL: IMPACT OF NEUTRALITY VS. PREJUDICE

4.1 Comparative Dynamics Of Neutral *vs.* Prejudiced Initializations

Recent studies (Francazi et al., 2024) have demonstrated that architectural choices significantly shape the initial predictions of untrained DNNs. In particular, MLPs with ReLU exhibit a pronounced *prejudice* in their initial predictions, a bias that amplifies with increasing depth. Specifically, deeper MLP architectures tend to assign predictions predominantly to a single class at initialization, resulting in highly prejudiced initial states. This phenomenon is quantitatively captured by the probability density function $f_{G_0}(g_0)$, which broadens and shifts toward the extremes as depth increases.

To explore the practical implications of these differences, we compare training dynamics initiated from distinct regions of the G_0 distribution. We select randomly initialized networks, dividing them into two groups:

- **Neutral Initializations**, *i.e.*, **Low IGB** ($G_0 \approx$

0.5): networks start with predictions uniformly distributed across classes; this corresponds to a sharply peaked distribution $f_{G_0}(g_0)$ centered at 0.5 (Fig. 1–bottom-left), indicating minimal prejudice in predictions.

- **Prejudiced Initializations** ($G_0 > 0.5$): networks exhibit biased predictions at initialization. [4] We distinguish between two subtypes: *weak prejudice*, where $f_{G_0}(g_0)$ is broadly spread over $[0.5, 1]$ (Fig. 1–center-left); and *deep prejudice*, *i.e.*, **High IGB** ($G_0 \approx 1$), where it is sharply peaked near 1, indicating extreme initial favoring of one class (Fig. 1–top-left).

The experiments in Fig. 2, conducted on synthetic data (`GB`) using a ReLU MLP without normalization (`SHLP`) (setting details in App. G.1), showcase notable differences in training dynamics between these prejudiced and neutral initial states. Our experiments reveal that these two filtered groups of initializations display nontrivial differences in convergence time (Fig. 2, upper right). In addition to this, we observe significant differences in the behavior of per-class performance metrics (Fig. 2, bottom right). Networks initialized in a neutral state exhibit a more symmetric evolution in class-wise performance, allowing global metrics to more faithfully represent the learning behavior across all classes. By contrast, prejudiced initializations lead to a marked divergence in per-class accuracy that persists throughout the onset of training. As a result, the overall performance metric becomes dominated by the initially favored class and no longer serves as a reliable indicator of model behavior on the full dataset. Interestingly, this persistent performance gap resembles the behavior typically observed in imbalanced classification problems (Francazi et al., 2023)—yet here, the dataset is balanced, and the asymmetry originates purely from initialization picking.

These differences hold important practical implications, particularly for hyperparameter optimization procedures, which typically rely on initial performance trends. Our findings demonstrate that prejudiced initializations can result in hyperparameter choices that inadvertently optimize performance exclusively for the dominant class at initialization, neglecting balanced class performance and thereby impairing fairness and overall training efficiency. App. C.1 provides an illustration of how the decision boundary evolves under neutral versus prejudiced initializations, together with an intuitive argument of how these differences shape the ensuing training dynamics.

[4] In MLP dynamics we relabel each initialization so that class 0 is the majority prediction at $t = 0$ (see end of Sec. 3); thus $G_0 = \max_c G_c$ and, for $N_C = 2$, $G_0 \in [0.5, 1]$. Static experiments keep fixed labels.

5 HOW NORMALIZATION LAYERS INFLUENCE THE INITIAL GUESSING BIAS

Sec. 4 discussed how the initial prediction bias—quantified by the statistic G_0—plays a key role in shaping the training dynamics of neural networks. Different values of G_0, sampled from the ensemble of random initializations, induce qualitatively different learning processes. Thus, the distribution of G_0 across initializations is a critical factor that determines the expected learning trajectory of a model. Francazi et al. (2024) have shown that activation functions and network depth significantly influence the distribution of G_0. In this section, we investigate how normalization layers affect $f_{G_0}(g_0)$, highlighting the significant impact of their position relative to activation functions.

Fig. 3 visually illustrates these outcomes, highlighting that post-activation normalization consistently yields initializations that fall within a neutral state, while pre-activation normalization leads to a distribution spread over the whole support, including both neutral and prejudiced initializations.

These differences translate into the dynamics. Building on the setup from Fig. 2, Fig. 4 incorporates BN or LN either before or after ReLU in the `SHLP`. The resulting dynamics, along with the distributions $f_{G_0}(g_0)$ shown in the inset plots, resemble the trends observed in Fig. 2 when normalization is applied **before** activation, for both BN and LN. However, moving the normalization **after** the activation dramatically alters the distribution: only the neutral region remains populated, effectively eliminating access to prejudiced states through random sampling. As a consequence, unfiltered (random) initializations exhibit the behavior characteristic of neutral regimes and depart significantly from the dynamics shown in Fig. 2.

5.1 Proving The Impact Of Normalizations On Networks Of Arbitrary Depth

Now, we provide theorems showing the impact of BN and LN on the distribution of G_0 at initialization. We will see that, BN and LN are equivalent for single-hidden layer models, but their effect becomes different as depth increases. We also prove that placing the norms before or after the activation has qualitatively different effects in terms of IGB. The four theorems are empirically corroborated in Fig. 3 through the theoretical curves corresponding to each case.

Our proofs are for ReLU networks because the effect of normalizations is generally a reduction of IGB, so it does not make sense to analyse activations such as tanh, which do not produce IGB (Francazi et al., 2024). Extension to other activations causing IGB

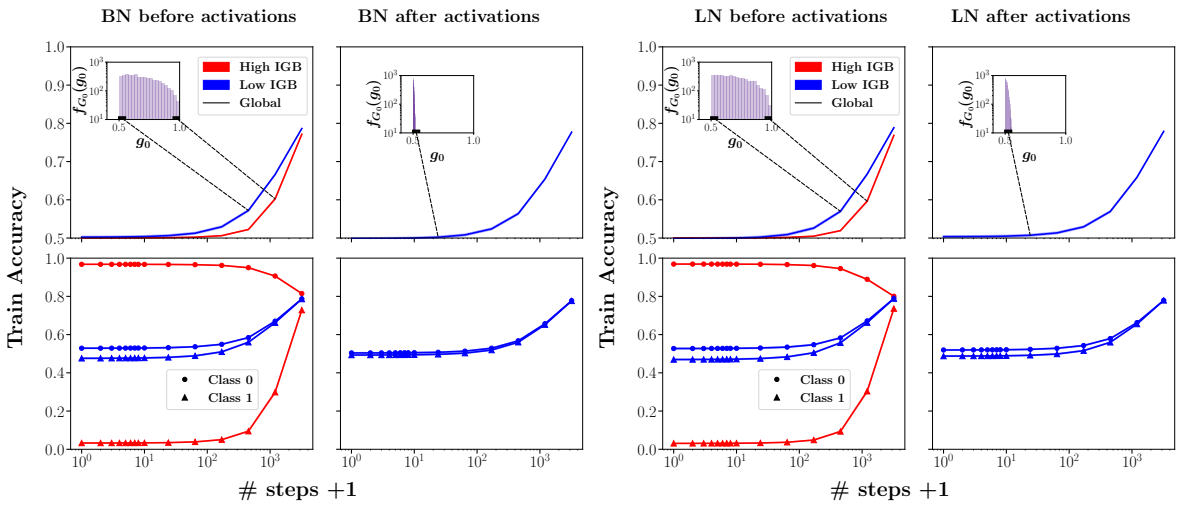

Figure 4: Training accuracy dynamics measured for the whole dataset (global, top) and for single-class performance (bottom). From the left to the right: BN before ReLU, BN after ReLU, LN before ReLU, LN after ReLU. Curves are averaged over multiple simulations. Simulations are filtered based on G_0 to isolate neutral and prejudiced initializations. In each run, class 0 corresponds to the initially dominant class to avoid averaging out class-specific effects. Filtering enables comparison of the distinct convergence behaviors associated with each initialization regime. Model: `SHLP`, Data: `GB` (see App. G for more details).

is straightforward and we do not expect qualitative differences.

5.1.1 Batch Normalization

We provide two theorems for BN, depending on its placement with respect to the ReLU activation.

Theorem 5.1 (Informal: BatchNorm + ReLU)**.** *Consider a deep MLP with ReLU activation, within the general setting described in Sec. 3. Placing BN* ***before*** *the activation leads to a weakly prejudiced initialization, that remains stable with increasing depth.*

The formal formulation of Th. 5.1 is provided in Th. E.1, alongside its proof.

Theorem 5.2 (Informal: ReLU + BatchNorm)**.** *Consider a deep MLP with ReLU activation, within the general setting described in Sec. 3. Placing BN* ***after*** *the activation promotes a neutral initialization state.*

The formal formulation of Th. 5.2 is provided in Th. E.2, alongside its proof.
These results show that Batch Normalization induces two qualitatively different behaviors depending on placement. When applied *after* the activation, BN effectively suppresses the initial prejudice and drives the model toward neutrality, reducing the spread of the guessing distribution $f_{G_0}(g_0)$ around 0.5.
In contrast, placing BN *before* the activation retains the initial prejudice seen in shallow unnormalized networks. However, crucially, this prejudice does not grow with depth: the distribution $f_{G_0}(g_0)$ remains broad but does not become increasingly concentrated at the extremes. Thus, BN-before acts as a stabilizing mechanism, preventing the depth-induced amplification of IGB that would otherwise occur in the absence of normalization (Francazi et al., 2024), though without actively eliminating the prejudice.

5.1.2 Layer Normalization

Also for LN, we provide two theorems, depending on its placement with respect to the ReLU activation.

Theorem 5.3 (Informal: LayerNorm + ReLU)**.** *Consider a deep MLP with ReLU activation, within the general setting described in Sec. 3. Placing LN* ***before*** *the activation leads to an increasingly prejudiced initialization as depth increases.*

The formal formulation of Th. 5.3 is provided in Th. F.2, alongside its proof.

Theorem 5.4 (Informal: ReLU + LayerNorm)**.** *Consider a deep MLP with ReLU activation, within the general setting described in Sec. 3. Placing LN* ***after*** *the activation promotes a neutral*

initialization state.

The formal formulation of Th. 5.4 is provided in Th. F.3, alongside its proof.

The behavior of Layer Normalization exhibits a more polarized dependence on placement. When LN is placed *after* the activation, it mirrors the prejudice-reducing effect of ReLU + BN: the initialization becomes neutral, with the distribution $f_{G_0}(g_0)$ sharply centered around 0.5, indicating balanced class predictions. However, placing LN *before* the activation leads to a significantly different behavior. In this case, the network retains the depth-amplified prejudice observed in architectures without normalization: as depth increases, the distribution $f_{G_0}(g_0)$ becomes increasingly concentrated near the extremes (0 and 1), reflecting highly prejudiced predictions. Unlike BN-before, which dampens this growth, LN-before leaves the depth-induced amplification of prejudice unchanged—mirroring the behavior observed in networks without normalization (Francazi et al., 2024).

5.1.3 LayerNorm vs RMSNorm.

Under LN + ReLU, the distribution of an individual hidden layer node is no longer centered at 0: each node becomes centered around a random shift. These shifts vary across nodes and are themselves drawn from a distribution that is symmetric and centered at 0 across the layer. Consequently, although single-node distributions are not symmetric around 0, the *layer-level* distribution obtained by aggregating over all nodes remains symmetric.

Crucially, in LN the mean is estimated over the entire layer (i.e., across nodes for a given input), rather than across samples at a fixed node (as in BN). At initialization, this layer-wise mean estimator is null. Therefore, even in the LayerNorm + ReLU configuration—where individual nodes exhibit non-zero means—LayerNorm is *formally equivalent to RMSNorm at initialization.* The practical implication is that RMSNorm avoids explicit mean estimation while preserving the same initialization-time stabilization effect, providing a principled explanation for why RMSNorm-style choices are natural, even in Pre-Norm architectures.

5.1.4 Norm Methods: Overview Of Effects

Together, these results reveal critical differences in how normalization layers interact with architectural depth and activation placement. Both BN and LN placed *after* activations consistently suppress prejudice and maintain a balanced initialization across depths. However, when placed *before* activations, their behaviors diverge sharply: BN + ReLU stabilizes the level of prejudice, whereas LN + ReLU allows it to keep growing with depth.

This divergence is not merely a quantitative effect but reflects a qualitative difference in the mechanisms at play. Under IGB, neurons within the same layer are not identically distributed: averaging across samples (as in BN) and averaging across neurons (as in LN) lead to fundamentally different normalizing factors. As a result, BN + ReLU and LN + ReLU cannot be regarded as interchangeable design choices, despite both belonging to the same family of normalization methods.

These theoretical results are confirmed by the simulations in Fig. 3: the theoretical curves closely match the empirical distributions across synthetic inputs and real data. The figure illustrates how placement and normalization type shape $f_{G_0}(g_0)$ across depths: BN-before flattens out the depth-induced amplification, LN-before maintains the extremal, prejudiced states—similarly to the unnormalized case—and both BN-after and LN-after maintain narrow, centered distributions indicative of neutrality.

6 COMPLEMENTARY EXPERIMENTS

6.1 Observations On Real Data.

Our empirical observations extend beyond MLPs and synthetic data. In Fig. 3, we report the distribution $f_{G_0}(g_0)$ computed on CIFAR-10 in lieu of random Gaussian inputs. The resulting profiles closely track those obtained with random data, indicating a regime in which architectural factors chiefly govern the distribution $f_{G_0}(g_0)$ while dataset structure plays a secondary role. *In this regime, the theoretical prediction from random-input analysis is not only qualitatively accurate but also descriptively faithful for real data.* These similarities, visible in the static distributions at initialization, are corroborated by further experiments on the dynamics shown in App. G.2, which mirror the random-data case.

Note however that our predictions are not expected to quantitatively describe any real dataset. As argued in (Francazi et al., 2024), strong correlations in the data are expected to amplify IGB. To this end, we examine MNIST, where pixels hold similar values through the whole dataset. As seen in Figs. G.5 and G.6, MNIST departs more from the random-input predictions than CIFAR-10 (the qualitative trends remain, but there are quantitative deviations), confirming the argument of (Francazi et al., 2024). Crucially, the *relative* impact of normalization placement remains consistent with our analysis: placing normalization *before* (rather than *after*) the activation increases the level of prejudice. The same qualitative pattern holds across different archi-

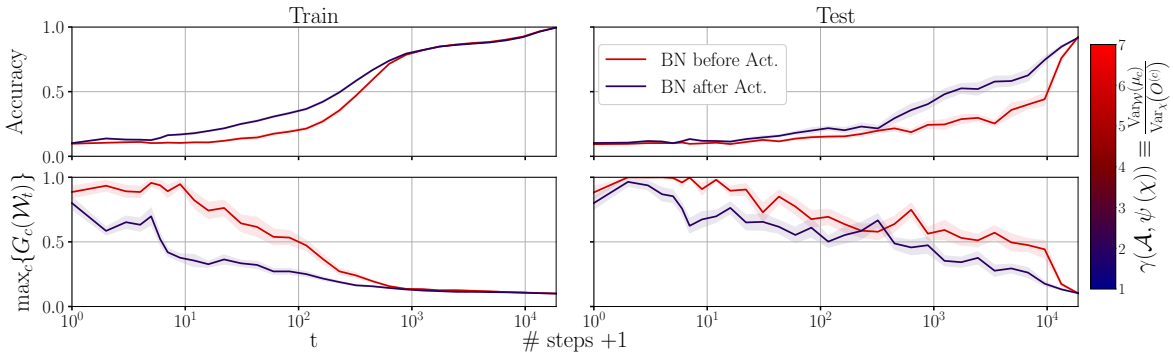

Figure 5: Comparison of ResNet-9 training on CIFAR-10 with the standard configuration (BN before activation) and an alternative design (BN after activation). The level of IGB in each setting is quantified using the variance ratio $\gamma(\mathcal{A}, \psi(\chi))$. **Top panels:** accuracy evaluated over the entire dataset. **Bottom panels:** evolution of the initial guessing bias, quantified at each training step by $\max_c G_c(\mathcal{W}_t)$, the maximum fraction of datapoints assigned to a single class. As predicted by our analysis, the post-activation configuration yields a lower $\gamma(\mathcal{A}, \psi(\chi))$, indicating a more balanced initial state. See App. G.1 for further experimental details.

tectures and datasets (see App. G.3). Taken together, our results show regimes in which architecture-induced effects alone yield descriptive predictions for structured data, and others where data structure perturbs $f_{G_0}(g_0)$, leaving the architecture-only theory at a qualitative level. Delineating these regimes and extending the theory to explicitly incorporate data-induced factors remains a rich direction for future work.

6.2 Exploring Impact Of Normalization Placement.

Given the difference between *Norm.+ReLU* and *ReLU+Norm.*, we examine widely adopted architectures (ResNets with BN and MLP-Mixers with LN) comparing the standard placement (*before* the activation) with the alternative (*after* the activation) suggested by our analysis (see App. G.3). Without filtering initializations, we directly evaluate the training dynamics to assess the practical implications of these choices. As shown in Fig. 5, and consistent with our theoretical and empirical insights, we observe clear differences in convergence and class-wise behaviour between the two placements. Placing normalization *after* the activation yields faster convergence and more balanced dynamics, especially in the early phase. This is evident not only in performance (top panels) but also in the bias trajectory, $\max_c G_c(\mathcal{W}_t)$ (bottom), which decays earlier under post-activation normalization. As model selection and hyperparameter tuning rely on early validation windows, persistent onset bias can skew the chosen configuration toward the initially over-predicted classes rather than the dataset as a whole. Notably, even when training curves later overlap, test performance continues to reflect a separation consistent with the earlier convergence-speed gap, suggesting that initialization prejudice can persist and influence generalization in a non-trivial way. These findings indicate that standard normalization placement heuristics may deserve closer scrutiny, and that revisiting them in light of initialization behavior can yield tangible improvements.

7 CONCLUSION

This work provides a theoretical characterization of how normalization design shapes the predictive behavior of DNNs at initialization, yielding distinct regimes of initial guessing bias (IGB) ranging from neutral to prejudiced. A key practical implication is that *placement* (Pre-Norm vs. Post-Norm) is not an implementation detail: Post-Norm shifts DNNs toward neutral regimes, while Pre-Norm preserves prejudice. This initialization gap in turn impacts convergence and class-wise dynamics during the onset of training. Importantly, because this bias can persist during the onset stage, global accuracy may be misleading for hyperparameter selection, as it can reflect the behavior of the initially dominant class more than that of the dataset as a whole.

Our results also clarify why *normalization type* can induce different regimes in deep settings. Under IGB, neurons within a layer are not identically distributed, so estimators computed at a single node (BN: averaging across samples) are not equivalent to estimators computed at the layer level (LN: averaging across nodes), especially in Pre-Norm configurations. Finally, while the single-node mean estimator can be non-zero in Pre-Norm, the layer-wise mean estimator is null at initialization; consequently, LayerNorm is formally equivalent to RMSNorm at initialization, explaining why RMSNorm can avoid mean estimation without changing the initialization-time stabilization effect in that regime.

Acknowledgements

This work was supported by the Swiss National Foundation, SNF grant # 196902.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Balestriero, R. and Baraniuk, R. G. (2022). Batch normalization explained. *arXiv preprint arXiv:2209.14778*.

Bassi, A., Baity-Jesi, M., Lucchi, A., Albert, C., and Francazi, E. (2026). When bias meets trainability: Connecting theories of initialization. In *International Conference on Learning Representations (ICLR)*. `https://openreview.net/forum?id=75ce9hnKne`, arXiv:2505.12096.

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. *Advances in neural information processing systems*, 31.

Boulila, W., Alshanqiti, E., Alzahem, A., Koubaa, A., and Mlaiki, N. (2024). An effective weight initialization method for deep learning: Application to satellite image classification. *Expert Systems with Applications*, 254:124344.

Casella, G. and Berger, R. (2002). *Statistical inference*. Duxbury.

Chen, Q. and Wu, R. (2017). Cnn is all you need. *arXiv preprint arXiv:1712.09662*.

Daneshmand, H., Kohler, J., Bach, F., Hofmann, T., and Lucchi, A. (2020). Batch normalization provably avoids ranks collapse for randomly initialised deep networks. *Advances in Neural Information Processing Systems*, 33:18387–18398.

Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Francazi, E., Baity-Jesi, M., and Lucchi, A. (2023). A theoretical analysis of the learning dynamics under class imbalance. In *International Conference on Machine Learning*, pages 10285–10322. PMLR.

Francazi, E., Lucchi, A., and Baity-Jesi, M. (2024). Initial guessing bias: How untrained networks favor some classes. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F., editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 13783–13839. PMLR.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

Jiang, Z., Gu, J., Zhu, H., and Pan, D. (2023). Prermsnorm and pre-crmsnorm transformers: equivalent and efficient pre-ln transformers. *Advances in Neural Information Processing Systems*, 36:45777–45793.

Joudaki, A., Daneshmand, H., and Bach, F. (2023). On bridging the gap between mean field and finite width deep random multilayer perceptron with batch normalization. In *International Conference on Machine Learning*, pages 15388–15400. PMLR.

Joudaki, A. and Hofmann, T. (2024). Emergence of globally attracting fixed points in deep neural networks with nonlinear activations. *arXiv preprint arXiv:2410.20107*.

Keller, J. B. and Kuske, R. (2001). Rate of convergence to a stable law. *SIAM Journal on Applied Mathematics*, 61(4):1308–1323.

Koehler, F., Zhou, L., Sutherland, D. J., and Srebro, N. (2021). Uniform convergence of interpolators: Gaussian width, norm bounds and benign overfitting. *Advances in Neural Information Processing Systems*, 34:20657–20668.

Kohler, J., Daneshmand, H., Lucchi, A., Hofmann, T., Zhou, M., and Neymeyr, K. (2019). Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 806–815. PMLR.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2.

Li, P., Yin, L., and Liu, S. (2024). Mix-ln: Unleashing the power of deeper layers by combining pre-ln and post-ln. *arXiv preprint arXiv:2412.13795*.

Liu, L., Liu, X., Gao, J., Chen, W., and Han, J. (2020). Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5747–5763.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

Loureiro, B., Sicuro, G., Gerbelot, C., Pacco, A., Krzakala, F., and Zdeborová, L. (2021). Learning gaussian mixtures with generalized linear models: Precise asymptotics in high-dimensions. *Advances in Neural Information Processing Systems*, 34:10144–10157.

Lubana, E. S., Dick, R., and Tanaka, H. (2021). Beyond batchnorm: Towards a unified understanding of normalization in deep learning. *Advances in Neural Information Processing Systems*, 34:4778–4791.

Lyu, K., Li, Z., and Arora, S. (2022). Understanding the generalization benefit of normalization layers: Sharpness reduction. *Advances in Neural Information Processing Systems*, 35:34689–34708.

Mele, M., Menichetti, R., Ingrosso, A., and Potestio, R. (2024). Density of states in neural networks: an in-depth exploration of learning in parameter space. *arXiv preprint arXiv:2409.18683*.

Mignacco, F., Krzakala, F., Urbani, P., and Zdeborová, L. (2020). Dynamical mean-field theory for stochastic gradient descent in gaussian mixture classification. *Advances in Neural Information Processing Systems*, 33:9540–9550.

Mishkin, D. and Matas, J. (2015). All you need is a good init. *arXiv preprint arXiv:1511.06422*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Nguyen, T. Q. and Salazar, J. (2019). Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th international conference on spoken language translation*.

Ni, Y., Guo, Y., Jia, J., and Huang, L. (2024). On the nonlinearity of layer normalization. *arXiv preprint arXiv:2406.01255*.

Noci, L., Anagnostidis, S., Biggio, L., Orvieto, A., Singh, S. P., and Lucchi, A. (2022). Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 35:27198–27211.

Pennington, J. and Worah, P. (2018). The spectrum of the fisher information matrix of a single-hidden-layer neural network. *Advances in neural information processing systems*, 31.

Petrov, V. V. (2012). *Sums of independent random variables*, volume 82. Springer Science & Business Media.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? *Advances in neural information processing systems*, 31.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. (2016). Deep information propagation. *arXiv preprint arXiv:1611.01232*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Sun, W., Song, X., Li, P., Yin, L., Zheng, Y., and Liu, S. (2025). The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. (2021). Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S., and Pennington, J. (2018). Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pages 5393–5402. PMLR.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. (2020). On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR.

Xu, J., Sun, X., Zhang, Z., Zhao, G., and Lin, J. (2019). Understanding and improving layer normalization. *Advances in neural information processing systems*, 32.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. (2019). A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*.

Zhang, B. and Sennrich, R. (2019). Root mean square layer normalization. *Advances in neural information processing systems*, 32.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   Justification: The paper presents the mathematical setting and intuition in the main text and enumerates assumptions alongside the formal statements in the supplemental material.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   Justification: The work provides a theoretical analysis of initialization bias rather than proposing a new algorithm that would necessitate a runtime or memory complexity study.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
   Justification: An anonymized repository with all code and instructions to reproduce figures and results is provided; environment and dependency details are documented in App. G.1.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   Justification: Each theoretical result is accompanied by a complete list of assumptions in the appendix/supplement.
   (b) Complete proofs of all theoretical results. [Yes]
   Justification: Full, formal proofs for all stated results are included in the supplemental material, with intuition provided in the main text.
   (c) Clear explanations of any assumptions. [Yes]
   Justification: Assumptions (e.g., unstructured/random data) are motivated in the main paper, and their scope and limitations are discussed, including robustness considerations in Section 6.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   Justification: Reproduction materials (code, scripts, and step-by-step instructions) are supplied via an anonymized repository; details are summarized in App. G.1.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   Justification: Training/test splits, hyperparameters, optimizer settings, and selection criteria are comprehensively documented in App. G.1.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   Justification: Results are averaged over multiple random seeds; error bars report the standard error of the mean as stated in App. G.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No]
   Justification: Detailed compute specifications (hardware type, execution time) are not reported; experiments were designed to be lightweight and feasible without reliance on large-scale GPU infrastructure, ensuring reproducibility.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   Justification: Public datasets used (e.g., CIFAR-10, MNIST) are properly cited in the manuscript.
   (b) The license information of the assets, if applicable. [Yes]
   Justification: Licenses and terms of use for all datasets are respected and explicitly mentioned as appropriate.
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   Justification: No new datasets or models are introduced beyond the reproducibility code package.
   (d) Information about consent from data providers/curators. [Not Applicable]
   Justification: No new data were collected; only standard publicly available datasets are used.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
   Justification: The work relies on common benchmarks and synthetic data and does not involve PII or sensitive content.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   Justification: The study does not involve crowdsourcing or human-subject experiments.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   Justification: No human participants were involved, so IRB approval is not required.
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]
   Justification: No participants were recruited and no compensation was provided.

Appendix

Contents

A Notation **16**

B Theoretical Setting and additional Related Work **19**

C Overview of Initial Guessing Bias (IGB) **20**

C.1 Geometric Interpretation of IGB Effects: Intuition Behind Differing Dynamics 21

D Conditions for Distribution convergence **23**

D.1 Convergence in Distribution 23

D.2 Sufficient Conditions for Convergence 23

D.3 Sufficient Conditions for Theorem D.2 24

E Batch Normalization **25**

E.1 BN + ReLU 25

E.2 ReLU + BN 37

F Layer Normalization **41**

F.1 LN + ReLU 42

F.2 ReLU + LN 48

G Experiments **51**

G.1 Reproducibility 51

G.2 Experiments on MLPs 52

G.3 Experiments on Other Architectures 53

A Notation

In this section, we thoroughly present the setting and notation used in our analysis. Our theory examines Multi-Layer Perceptrons (MLPs), where the propagation of an input signal, the datapoint "a", denoted as $\mathbf{X}_{;a}^{(0)}$, through the network is governed by the following set of equations:

$$\mathrm{Z}_{i;a}^{(l+1)} = \sum_j w_{ij}^{(l+1)} \mathrm{X}_{j;a}^{(l)} \,, \tag{1}$$

$$\mathbf{X}_{;a}^{(l)} = X\left(\left\{\mathbf{Z}_{;a}^{(l)}\right\}_{a=1}^{D}\right) \tag{2}$$

$$\mathrm{Z}_{c;a}^{(L+1)} \equiv \mathrm{O}_{c;a} \,, \tag{3}$$

where $l \in \{0, \ldots, L\}$ indicate the layer, $a \in \{1, \ldots, D\}$ the sample index, $i, j \in \{1, \ldots, N_l\}$ the node index. The operator $X(\cdot)$ is given by the composition of two operations, the activation function and the normalization. In our theoretical analysis, we will consider two different normalization schemes (BN and LN) and, for each of them, two different settings defined by the placement of the normalization with respect to the activations. We will therefore consider in total 4 different settings:

- activation + BN : $X(\cdot) = \mathrm{B} \circ \phi(\cdot)$
- BN + activation: $X(\cdot) = \phi \circ \mathrm{B}(\cdot)$
- activation + LN: $X(\cdot) = \mathrm{L} \circ \phi(\cdot)$
- LN + activation: $X(\cdot) = \phi \circ \mathrm{L}(\cdot)$

$\phi(\cdot) : \mathbb{R} \to \mathbb{R}$ represents the activation function, applied elementwise across the layer.

We now define the normalization operators used throughout our analysis. For a fixed layer l, node index i, and sample index a, the Batch Normalization (BN) and Layer Normalization (LN) operators are defined as follows:

Batch Normalization (BN). The BN operator normalizes across the minibatch:

$$\mathrm{B}\left(\mathrm{Z}_{i;a}^{(l)}\right) = \alpha_i^l \cdot \frac{\mathrm{Z}_{i;a}^{(l)} - \hat{\mu}_i^{(l)}}{\sqrt{\left(\hat{\sigma}_i^{(l)}\right)^2 + \epsilon}} + \beta_i^l \,, \tag{4}$$

where the statistics are computed as:

$$\hat{\mu}_i^{(l)} = \frac{1}{|B|} \sum_{b \in B(a)} \mathrm{Z}_{i;b}^{(l)},$$

$$\left(\hat{\sigma}_i^{(l)}\right)^2 = \frac{1}{|B|} \sum_{b \in B(a)} \left(\mathrm{Z}_{i;b}^{(l)} - \hat{\mu}_i^{(l)}\right)^2 .$$

where $B(a)$ indicates the batch of where the datapoint a belongs. The parameters α_i^l and β_i^l are trainable scaling and shifting factors, and ϵ is a small constant added for numerical stability.

Layer Normalization (LN). The LN operator normalizes across the layer dimensions for each sample independently:

$$\mathrm{L}\left(\mathrm{Z}_{i;a}^{(l)}\right) = \alpha_i^l \cdot \frac{\mathrm{Z}_{i;a}^{(l)} - \hat{\mu}_a^{(l)}}{\sqrt{\left(\hat{\sigma}_a^{(l)}\right)^2 + \epsilon}} + \beta_i^l \,, \tag{5}$$

where the mean and variance are computed as:

$$\hat{\mu}_a^{(l)} = \frac{1}{N_l}\sum_{i=1}^{N_l} \mathrm{Z}_{i;a}^{(l)},$$
$$\left(\hat{\sigma}_a^{(l)}\right)^2 = \frac{1}{N_l}\sum_{i=1}^{N_l}\left(\mathrm{Z}_{i;a}^{(l)} - \hat{\mu}_a^{(l)}\right)^2.$$

The principal notation is summarized in Fig. A.1.

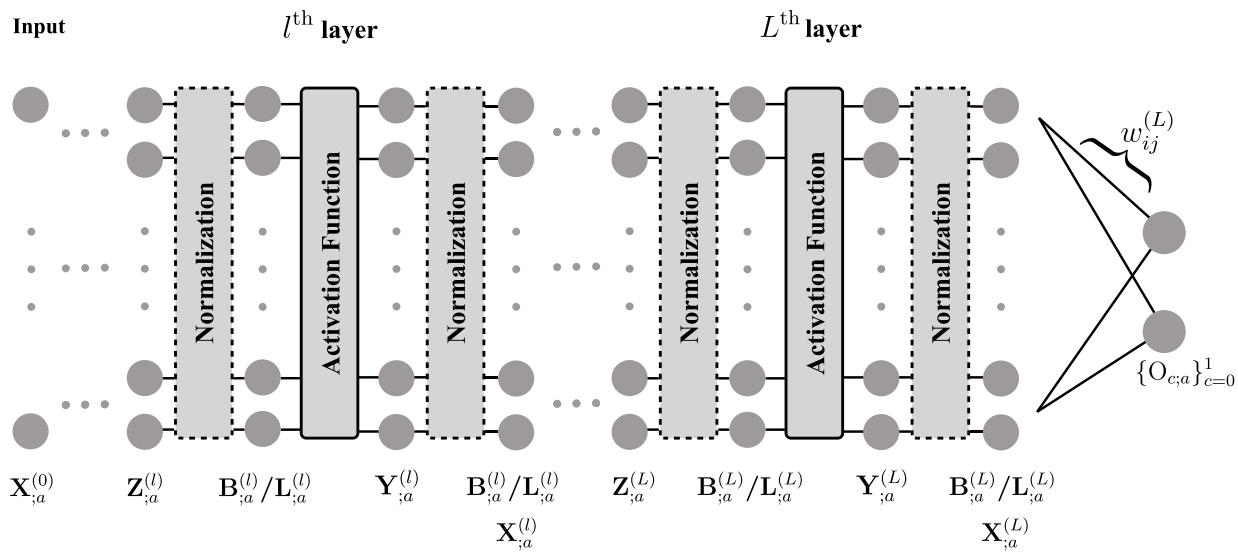

Figure A.1: Illustration of a generic feedforward neural network used for binary classification, along with the main notation for node-level variables. Normalization layers (representing either Batch Normalization or Layer Normalization) are shown with dashed contour to indicate that they may be positioned either before or after the activation function, depending on the configuration.

We also use the following notation:

- $\{\cdot\}_{i=0}^{M-1}$: set of M elements. If some of the indices of the variables are fixed (*i.e.* equal for every element of the set), the set indices (indices that vary across different elements of the set) are reported explicitly on the right. If the index of the set elements is not explicitly reported, it means the absence of fixed indices for the set variables (*i.e.* all indices are set indices).
- $\mathbf{B}_{;a}^{(l)}$: activation vector at layer l for input sample a, given by $\left(\mathrm{B}_{1;a}^{(l)}, \ldots, \mathrm{B}_{N_l;a}^{(l)}\right)$, after passing through the BN; $\mathrm{B}_{i;a}^{(l)}$ indicate the component corresponding to node i.
- $\mathbb{E}\left(x\right)$: Indicate the expectation value of the argument, x. If the average involves only one source of randomness this is explicitly indicated, *e.g.* $\mathbb{E}_{\chi}\left(x\right)$ indicates an average over the dataset distribution, while $\mathbb{E}_{\mathcal{W}}\left(x\right)$ an average over the distribution of network weights.
- $\mathrm{erf}\left(\cdot\right)$: Error function. $\mathrm{erf}\left(\mathrm{y}\right) \equiv \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$
- $F_X\left(x\right)$:Given a *r.v.* X, we denote its cumulative distribution function (*c.d.f.*) as $F_X\left(x\right)$, i.e., $F_X\left(x\right) = \mathbb{P}\left(X < x\right)$. Considering the *r.v.* $X(\chi, \mathcal{W})$, which is a function of two independent sets of random variables χ and $\mathcal{W}$, when one of these sources of randomness is fixed, we explicitly indicate the active source in the notation. For example, $F_X^{(\chi)}\left(x\right) = \mathbb{P}\left(X < x \mid \mathcal{W}\right)$.

- G_c: fraction of dataset elements classified as belonging to class c. The argument M indicates the total number of output nodes for the variable definition, *i.e.* the number of classes considered. For binary problems we omit this argument (G_0) as there is only one non-trivial possibility, *i.e.* $M = 2$.
- $G_{\tilde{c}}$: the set $\{G_c\}_{c=0}^{M-1}$ contain the same elements of $\{G_{\tilde{c}}\}_{c=0}^{M-1}$, but these are ranked by magnitude, such that $G_{\tilde{0}}$ is the greatest output value between the M elements of the set, $G_{\tilde{1}}$ the second one and so on.
- $\mathbf{L}_{;a}^{(l)}$: activation vector at layer l for input sample a, given by $\left(\mathrm{L}_{1;a}^{(l)}, \ldots, \mathrm{L}_{N_l;a}^{(l)}\right)$, after passing through LN; $\mathrm{L}_i^{(l)}$ indicate the component corresponding to node i.
- $N_C \equiv N_{L+1}$: number of output nodes, *i.e.* the number of classes.
- N_l: number of nodes in the lth layer; $N_0 \equiv d$ indicates the dimension of the input data (number of input layer nodes) while N_{L+1} the number of classes (number of output layer nodes).
- $\mathcal{N}\left(x; \mu, \sigma^2\right)$: Given a Gaussian *r.v.* X, we indicate with $\mathcal{N}\left(x; \mu, \sigma^2\right)$ the *p.d.f.* computed at $X = x$, *i.e.* $\mathcal{N}\left(x; \mu, \sigma^2\right) \equiv f_X\left(x\right) = \frac{e^{-\frac{1}{2\sigma^2}(x-\mu)^2}}{\sqrt{2\pi\sigma^2}}$
- $\mathrm{O}_{c,M}$: output layer node; c is the node index; the index M, instead, indicate the total number of nodes considered. For binary problems we omit the subscript index to keep the notation lighter, *i.e.* O_c .
- $\mathrm{O}_{\tilde{0},M}$: the set $\{\mathrm{O}_{\tilde{c},M}\}_{c=0}^{M-1}$ contain the same elements of $\{\mathrm{O}_{\tilde{c},M}\}_{\tilde{c}=0}^{M-1}$, but these are ranked by magnitude, such that $\mathrm{O}_{\tilde{0},M}$ is the greatest output value between the M, $\mathrm{O}_{\tilde{1}M}$ the second one, and so on.
- $\mathbb{P}\left(A\right)$: Denotes the probability associated with event A. $\mathbb{P}\left(A \mid B\right)$ indicates the probability of event A given event B.
- $f_X\left(x'\right)$: Given a random variable X, $f_X\left(x'\right)$ denotes the probability density function (*p.d.f.*) evaluated at $X = x'$. Formally, $f_X\left(x'\right) = \frac{d}{dx} F_X\left(x\right)\Big|_{x=x'}$. For variables with multiple sources of randomness, if some of these sources are either fixed or marginalized, we will specify the active sources of randomness as subscripts. For example, given $X(\chi, \mathcal{W})$, a function of two independent sets of random variables χ and $\mathcal{W}$, $f_X^{(\chi)}\left(x'\right)$ equals $F_X^{(\chi)}\left(x\right)\Big|_{x=x'}$.
- $\mathrm{Var}\left(\cdot\right)$: Indicate the variance of the argument. Since we have *r.v.*s with multiple sources of randomness where necessary we will specify in the subscript the source of randomness used to compute the expectation. For example $\mathrm{Var}_\chi\left(\cdot\right) \equiv \mathbb{E}_\chi\left(\cdot - \mathbb{E}_\chi\left(\cdot\right)\right)^2$. For the sake of compactness, we will employ sometimes the shorthand notation $\mathrm{Var}_\chi\left(\cdot\right) = \sigma_\cdot^2$.
- $\mathcal{W}_t$: shorthand notation for the set of network weights, $\{w_{ij}^{(l)}\}$ at time t; $\mathcal{W} \equiv \mathcal{W}_0$. We use, instead the notation $\mathcal{W}^l \subseteq \mathcal{W}$ to indicate the subset of weights relative to a specific layer, *i.e.* $\mathcal{W}^l \equiv \{w_{ij}^{(l)}\}_{\substack{j\in[0,\ldots,N_l] \\ i\in[0,\ldots,N_{l+1}]}}$. $\mathcal{W}^{<l}$, $\mathcal{W}^{>l}, \ldots$ are defined analogously.
- $w_{ij}^{(l)}$: element ij of the matrix $\boldsymbol{W}^{(l)}$, connecting two consecutive layers ($l \in [0, \ldots, L]$). Given the matrix $\boldsymbol{W}^{(l)}$ we use a 'placeholder' index, $\cdot$, to return column and row vectors from the weight matrices. In particular $w_{j\cdot}^{(l)}$ denotes row j of the weight matrix $\boldsymbol{W}^{(l)}$; similarly, $w_{\cdot j}^{(l)}$ denotes column j.
- $\mathbf{X}_{;a}^{(0)} \in \mathbb{R}^d$: a^{th} input vector; when the index a is omitted we mean a generic vector, $\mathbf{X}^{(0)}$, drawn from the population distribution.
- $\mathbf{Y}_a^{(l)}$: activation vector at layer l for input sample a, given by $\left(\mathrm{Y}_{1;a}^{(l)}, \ldots, \mathrm{Y}_{N_l;a}^{(l)}\right)$. Each component $\mathrm{Y}_{i;a}^{(l)}$ corresponds to the post-activation value of node i.
- $\mathbf{Z}_{;a}^{(l)}$: pre-activation vector at layer l for input sample a, obtained after the linear transformation of the activations from layer $l-1$ and before applying any activation or normalization. Explicitly, $\mathbf{Z}_{;a}^{(l)} = \left(\mathrm{Z}_{1;a}^{(l)}, \ldots, \mathrm{Z}_{N_l;a}^{(l)}\right)$.
- $\delta\left(x\right)$: Dirac delta function.

- $\Theta(x)$: Heaviside step function.
- $\mu_X \equiv \mathbb{E}_\chi(X)$: expected value (mean) of a generic random variable X over the dataset distribution, conditioned on a fixed realization of the weights $\mathcal{W}$. For example, μ_{O_c} denotes the average value of the output node c over the dataset. As shorthand, we write $\mu_c \equiv \mathbb{E}_\chi(\mathrm{O}_c)$.
- $\sigma_X \equiv \sqrt{\mathbb{E}_\chi((X-\mu_X)^2)}$: standard deviation of a generic random variable X over dataset randomness, given a fixed realization of the weights $\mathcal{W}$. This quantifies variability in X across different input samples from the dataset.
- $\chi = (\mathbf{X}_{;a}^{(0)}, y_a)_{a=1}^D$: dataset composed by D pairs of input vectors-labels.

Abbreviations

- BN: Batch Normalization
- *c.d.f.*: cumulative distribution function
- CNN: Convolutional neural network
- DNN: Deep neural network
- EOC: edge of chaos
- IGB: Initial guessing bias
- LN: Layer Normalization
- MLP: Multi-layer perceptron
- *p.d.f.*: probability density function
- *r.v.*: random variable
- *w.h.p.*: with high probability

B Theoretical Setting and additional Related Work

Multi-Layer Perceptrons (MLPs) are fundamental building blocks for various contemporary architectures He et al. (2016); Vaswani et al. (2017); Dosovitskiy et al. (2020), including recent models such as MLP-mixer Tolstikhin et al. (2021), achieving notable results on complex tasks. Understanding the initialization behavior of MLPs is essential due to their broad applicability and evolving importance. We specifically examine MLPs integrated with normalization techniques—Batch Normalization (BN) and Layer Normalization (LN)—and their effects on output distributions and Initial Guessing Bias (IGB).

Normalization methods, particularly BN and LN, have become essential components in modern neural network architectures. Originally introduced to address internal covariate shifts and stabilize training processes (Ioffe and Szegedy, 2015; Ba et al., 2016), these techniques normalize activations by adjusting their means and variances, significantly impacting prediction statistics. Their empirical success has spurred widespread adoption across various domains, such as computer vision and natural language processing (Bjorck et al., 2018; Xu et al., 2019). While normalization's influence on training dynamics, optimization, and function representation has been extensively explored (Santurkar et al., 2018; Yang et al., 2019; Xu et al., 2019), comparatively fewer studies have focused on its effects on prediction behavior at initialization (Daneshmand et al., 2020; Kohler et al., 2019; Ni et al., 2024; Lyu et al., 2022; Balestriero and Baraniuk, 2022).

We consider an MLP with L hidden layers, applied to a dataset of input vectors $\{\mathbf{X}_{;a}^{(0)}\}_{a=1}^D$. Each input is processed through the network to produce two output values, $\{\mathrm{O}_{c;a}\}_{c=0,1}$, with the predicted class determined by the larger of the two. We focus on unstructured Gaussian inputs, where each component is independently sampled as $\mathrm{X}_i^{(0)} \sim \mathcal{N}(0,1)$.

Weights are initialized using **Kaiming Initialization**, with $w_{ij}^{(l)} \sim \mathcal{N}\left(0, \frac{\sigma_w^2}{N_l}\right)$, where N_l is the number of neurons

in the lth hidden layer. In our experiments we adopt the standard setting $\sigma_w^2 = 2$. While our analysis focuses on this concrete case for clarity, the framework extends to more general classes of random input distributions and initialization schemes, as discussed in (Francazi et al., 2024). For complete notation, see App. A.

Our analysis focuses on the infinite-width limit, employing tools such as the Central Limit Theorem (CLT) and its generalizations (Francazi et al., 2024), assuming Gaussian-distributed inputs and using Kaiming Normal initialization He et al. (2015). These assumptions provide a symmetric and unbiased environment, isolating network-intrinsic behaviors and facilitating the study of IGB and normalization effects. While Gaussian inputs simplify theoretical analysis, our findings generalize to other input distributions satisfying CLT conditions. Similarly, although we primarily utilize ReLU activation functions for explicit calculations, our theoretical results hold broadly across generic activation functions.

The normalization layers' positions are flexible in our study, considering BN and LN applied either before or after the activation functions. BN normalizes across batches, while LN normalizes within individual layers. This variability helps us systematically explore their influence on network initialization and the resulting biases. Furthermore, while biases are set to zero to maintain symmetry, extensions to non-zero biases and structured input data scenarios remain feasible, following methodologies similar to those in (Francazi et al., 2024, 2023).

Beyond these directions, other works have examined predictive behavior at initialization and normalization placement from complementary perspectives. Joudaki et al. (2023) introduced an entropy-based characterization of predictive imbalance. While this approach highlights empirical, indirect signatures of class bias, the IGB framework provides a direct theoretical formalization of the per-class prediction distribution. These views are complementary: entropy-based insights can be embedded within the IGB formalism to yield explicit theoretical predictions. Joudaki and Hofmann (2024) analyzed how changes in LayerNorm placement affect signal correlations across weight configurations. This classical mean-field measure can be directly connected to IGB bias measures, as shown in Bassi et al. (2026). Recent Transformer studies further reinforce the importance of normalization placement for depth-wise trainability: **?** show that Pre-LN and Post-LN allocate gradient strength differently across layers and propose a hybrid placement to improve deep-layer effectiveness, while Sun et al. (2025) argue that Pre-LN induces depth-wise variance growth that can make deeper blocks increasingly identity-like, motivating a depth-dependent LayerNorm scaling. These works are closely related in showing that normalization placement shapes signal propagation, gradients, and representation diversity; our focus is different and complementary, namely to characterize how BN and LN modify initialization-time output distributions and class-assignment regimes under the IGB framework. Finally, density-of-states methods have been proposed at finite width, where the energy is defined via misclassified examples, thereby granting access to predictive behaviour across possible configurations (Mele et al., 2024).

C Overview of Initial Guessing Bias (IGB)

Initial Guessing Bias (IGB) characterizes the phenomenon where untrained neural networks exhibit an inherent bias in their initial class predictions, even in symmetric settings with identically distributed classes and the absence of non-null bias terms. Specifically, given an architecture parameterized by weights $\mathcal{W}$, the fraction of points initially guessed as class c by the network is denoted by $G_c(\mathcal{W})$. IGB manifests as an imbalance in this fraction across classes.

For clarity, we focus on binary classification problems, noting that the analysis naturally extends to multi-class scenarios. Without loss of generality, class 0 is used as representative, leveraging the statistical symmetry between classes.

In the limit of infinitely many datapoints ($D \to \infty$), by the Law of Large Numbers, we have:

$$\begin{aligned} \lim_{D\to\infty} G_0\left(\mathcal{W}\right) &= \mathbb{P}\left(\mathrm{O}_0 > \mathrm{O}_1 \mid \mathcal{W}\right) \\ &= \int_0^\infty f^{(\chi)}_{\Delta_{\mathrm{O}_0}}\left(\delta_{\mathrm{o}}\right)\, d\delta_{\mathrm{o}}\,, \quad \text{with} \quad \Delta_{\mathrm{O}_0} \equiv \mathrm{O}_0 - \mathrm{O}_1. \end{aligned} \tag{6}$$

The IGB framework allows the derivation of the distribution $f^{(\chi)}_{\mathrm{O}_c}(\mathrm{o})$. Specifically, we recall the following result from Francazi et al. (2024):

Theorem C.1 (Informal)**.** *Consider a Gaussian distributed dataset processed through an MLP with L hidden layers and weights initialized according to Kaiming normal initialization (or schemes with similar scaling properties). In the limit of infinite width, the distribution of an output node O_c, at initialization, converges to:*

$$f^{(\chi)}_{\mathrm{O}_c}(\mathrm{o}) \xrightarrow{|\mathcal{W}|\to\infty} \mathcal{N}\left(\mathrm{o}; \mu_c, Var_\chi\left(\mathrm{O}_c\right)\right), \tag{7}$$

where $\mu_c = \mathbb{E}_\chi\left(\mathrm{O}_c\right) \equiv \mathbb{E}_{\mathbf{X}^{(0)}}[\mathrm{O}_c]$ and $Var_\chi\left(\mathrm{O}_c\right) \equiv Var_{\mathbf{X}^{(0)}}(\mathrm{O}_c)$. Moreover, the center μ_c is itself a random variable varying across initializations, converging to:

$$f_{\mu_c}(m) \xrightarrow{|\mathcal{W}|\to\infty} \mathcal{N}\left(m; 0, Var_{\mathcal{W}}\left(\mu_c\right)\right). \tag{8}$$

In other words, the IGB framework reveals that initial guessing biases can be characterized comparing two types of fluctuations: those in $f_{\mu_c}(m)$, representing variability across different output nodes, and those in $f^{(\chi)}_{\mathrm{O}_c}(\mathrm{o})$, representing variability across datapoints for a fixed initialization. Intuitively:

- **Absence of IGB:** If initialization fluctuations ($f_{\mu_c}(m)$) are significantly smaller than data-driven fluctuations ($f^{(\chi)}_{\mathrm{O}_c}(\mathrm{o})$), outputs for both classes largely overlap, resulting in nearly equal fractions of assigned classes ($G_0 \approx 1/2$).
- **Strong IGB:** Conversely, if initialization fluctuations dominate, output distributions for each class become well-separated, leading to an overwhelming fraction assigned to a single class ($G_0 \approx 1$).

A natural translation of this intuition into a quantifiable metric is provided by the following definition:

Definition C.2 (Variance Ratio)**.** Given an architecture $\mathcal{A}$ and a dataset $\psi(\chi)$, we define the Variance Ratio $\gamma(\mathcal{A}, \psi(\chi))$ as:

$$\gamma(\mathcal{A}, \psi(\chi)) \equiv \frac{\mathrm{Var}_{\mathcal{W}}\left(\mu_c\right)}{\mathrm{Var}_\chi\left(\mathrm{O}_c\right)}, \tag{9}$$

where:

- $\mathrm{Var}_{\mathcal{W}}\left(\mu_c\right)$ is the variance (over random initializations $\mathcal{W}$) of the output node mean over the dataset randomness $\mu_c = \mathbb{E}_\chi\left(\mathrm{O}_c\right)$,
- $\mathrm{Var}_\chi\left(\mathrm{O}_c\right)$ is the variance (over data inputs $\mathbf{X}^{(0)}$) of the output O_c, for a fixed initialization.

We further generalize this definition by introducing $\gamma^{(l)}$, computed analogously at any layer l of the network.

C.1 Geometric Interpretation of IGB Effects: Intuition Behind Differing Dynamics

We provide an intuitive argument for the interpretation of the distinct training dynamics observed under neutral and prejudiced initializations, as shown in Fig. 2. These differences stem from IGB, which reflects the spatial relationship between the decision boundary and the distribution of input representations (Francazi et al., 2024). In the prejudiced state, IGB is caused by a drift in the hidden representation of the last layer. As the data representations move away from the center of the activation space—*i.e.*, the vector space formed by the layer's output activations—the decision boundary—typically initialized near the origin—fails to bisect the signal. Consequently, all data points fall predominantly on one side of the hyperplane, causing the network to assign the vast majority of inputs to a single class (Fig. C.1 (c)). Conversely, in the neutral state, the distribution of the hidden representation of the last layer remains approximately centered. The decision boundary intersects this central region, partitioning the datapoints more evenly between classes (Fig. C.1 (a)). This leads to a more balanced class-wise prediction behavior at initialization.

These initial conditions can naturally give rise to structurally different learning dynamics, which appear to operate over distinct regimes and characteristic timescales. In prejudiced initializations (Fig. C.1 (c,d,e)), training unfolds in two steps. During the initial regime ($0 < t < \tau_{P'}$), the decision boundary approaches the region where data

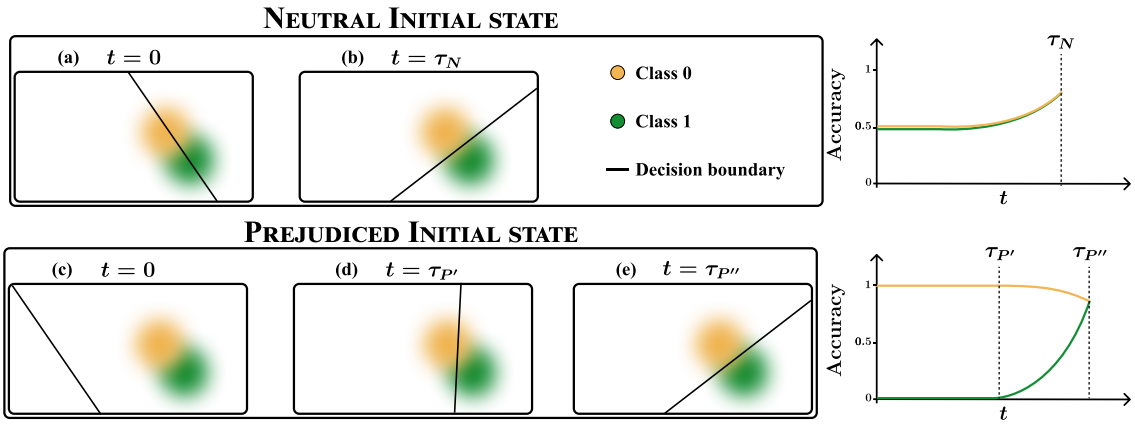

Figure C.1: Low-dimensional representation of decision boundary dynamics for prejudiced vs. neutral initializations. In the neutral case, the boundary already intersects the relevant region (a), and training involves only directional adjustment (b). In the prejudiced case, instead, the distribution of the hidden representation occupy a region far from the decision boundary (c). In this case one can argue that the training proceeds in two steps: translating the boundary to the high-density region of datapoint representations (d), and adjusting the orientation to optimize class separation (e). These regimes reflect fundamentally different underlying dynamics.

representations are concentrated. This is followed by a second phase ($\tau_{P'} < t < \tau_{P''}$), where the boundary is gradually rotated and refined to achieve effective class separation. In contrast, neutral initializations involve only a single training phase (Fig. C.1 (a,b)). Since the decision boundary already intersects the central region of the data representations, training proceeds via local directional adjustments to align the boundary with the true class separation. This single-phase regime is governed by a unified timescale τ_N.

To illustrate these differences visually, Fig. C.1 presents a low-dimensional schematic of the decision boundary evolution. In the neutral case, the boundary initially intersects the region of interest (a) and requires only directional tuning (b). In the prejudiced case, the boundary must first be translated toward the signal region (d) before undergoing orientation adjustments to optimize separation (e). The right side of Fig. C.1 translates the geometric interpretation discussed on the left into the temporal structure of training dynamics, summarizing how the different phases and timescales associated with each initialization regime can give rise to the distinct convergence behaviors observed in Fig. 2.

D Conditions for Distribution convergence

In this section, we briefly review the key results on the convergence in distribution of random variables, which form the foundation of our theoretical analysis. These results have been extensively discussed in earlier works; for more detailed explanations and proofs, we refer the reader to Petrov (2012); Francazi et al. (2024). Below, we outline the necessary conditions for convergence and characterize the resulting asymptotic distributions.

D.1 Convergence in Distribution

Our analysis focuses on the convergence of random variable combinations to their asymptotic distributions. The cornerstone of this discussion is the *Central Limit Theorem (CLT)*, which describes the conditions under which sums of random variables converge to a Gaussian distribution.

D.1.1 Central Limit Theorem

The *Central Limit Theorem (CLT)* applies to sequences of independent and identically distributed (*i.i.d.*) random variables. Given a sequence of *i.i.d.* random variables $x_1, x_2, \dots$ with finite mean μ and variance σ^2, the sample mean $\bar{x}_n$ of the first n variables converges to a normal distribution:

$$Z = \lim_{n\to\infty} \frac{\bar{x}_n - \mu}{\sigma/\sqrt{n}} \sim \mathcal{N}(0,1).$$

Corrections to this asymptotic form for finite n are discussed in Keller and Kuske (2001).

D.1.2 Extensions of the CLT

The CLT can be extended to sequences of independent but non-identically distributed random variables. Under conditions like the Lyapunov or Lindeberg conditions, it remains possible to demonstrate convergence to a Gaussian distribution. One such condition, the *Lindeberg condition*, for a sequence of independent random variables $X_1, X_2, \dots$, ensures that:

$$\lim_{n\to\infty} \sum_{i=1}^{n} \frac{1}{s_n^2} \int_{|x|\geq \epsilon s_n} (x-\mu_i)^2 dF_{X_i}(x) = 0,$$

where $s_n^2 = \sum_{i=1}^{n} \mathbb{E}[(X_i - \mu_i)^2]$ and $F_{X_i}(x)$ is the cumulative distribution function of X_i.

Theorem D.1 (Lindeberg's Theorem)**.** *For a sequence of independent random variables $\{X_i\}_{i=1}^n$, if the Lindeberg condition is satisfied for all $\epsilon > 0$, then the normalized sum $Z_n = \frac{S_n - \sum_i \mu_i}{s_n}$ converges to a standard normal distribution:*

$$f_{Z_n}(z) \xrightarrow{n\to\infty} \mathcal{N}(z;0,1).$$

For additional details, see Ref. Petrov (2012).

D.2 Sufficient Conditions for Convergence

We will also rely on an alternative set of necessary and sufficient conditions that guarantee convergence to a normal distribution. These conditions are outlined below, and the full proof can be found in Refs. Petrov (2012); Francazi et al. (2024).

Theorem D.2 (Distribution Convergence)**.** *Consider a set of independent zero-mean random variables*

$\{X_i\}_{i=1}^n$. *If, for every* $\epsilon > 0$, *the following conditions are satisfied:*

$$\textit{Concentration:} \quad \sum_{i=1}^{n} \mathbb{P}\left(|X_i| \geq \epsilon\right) \xrightarrow{n\to\infty} 0 \quad \forall \epsilon \in \mathbb{R}^+, \tag{10}$$

$$\textit{Mean Normalization:} \quad \sum_i \left(\int_{|x|<\epsilon} x f_{X_i}(x)\, dx \right) \xrightarrow{n\to\infty} 0, \tag{11}$$

$$\textit{Variance Normalization:} \tag{12}$$

$$s_n^2 = \sum_i \left(\int_{|x|<\epsilon} x^2 f_{X_i}(x)\, dx - \left(\int_{|x|<\epsilon} x f_{X_i}(x)\, dx \right)^2 \right) \xrightarrow{n\to\infty} \sigma^2, \tag{13}$$

then the distribution of the sum $\sum_i X_i$ *converges to* $\mathcal{N}(0, \sigma^2)$.

D.3 Sufficient Conditions for Theorem D.2

In many cases, it is sufficient to check simpler conditions for convergence. Specifically, the following conditions are sufficient for convergence to a Gaussian distribution (the full proof can be found in Ref. Francazi et al. (2024).):

Theorem D.3 (Sufficient Conditions for Distribution Convergence)**.** *Let* $\{X_i\}_{i=1}^n$ *be a sequence of independent, zero-mean random variables. If they satisfy the following conditions:*

$$\textit{Variance Convergence:} \quad \sum_{i=1}^{n} \mathbb{E}[X_i^2] \xrightarrow{n\to\infty} n\tilde{\sigma}^2, \tag{14}$$

$$\textit{Fast Decreasing Tails:} \quad \lim_{x\to\pm\infty} f_{X_i}(x) = \mathcal{O}\left(\frac{1}{x^4}\right) \quad \forall i, \tag{15}$$

then the normalized sum of the random variables converges in distribution to $\mathcal{N}(0, \sigma^2)$.

E Batch Normalization

E.1 BN + ReLU

In this section, we show that, when BN is applied before the activation function, the output distribution of each layer remains consistent throughout the network, regardless of depth. Specifically, after applying BN, the output distribution of a deep network with many hidden layers is equivalent to that of a network with just a single hidden layer without any normalization. We establish this result using an induction argument, demonstrating that the distribution of nodes at each hidden layer, at initialization, follows the same distribution, independent of the network depth. For clarity, we focus on neural networks with weights initialized using the Kaiming Normal scheme, but the analysis extends to other initialization methods (*e.g.*, Xavier Uniform) as long as the variance scaling is preserved. Furthermore, these results extend to more complex activation architectures, including when incorporating pooling layers Francazi et al. (2024).

The following is a formal formulation of Th. 5.1.

Theorem E.1 (BN + ReLU)**.** *Consider the setting described in Sec. B with an MLP where BN is applied before the activation function. Assuming that the activations* $\left\{\mathrm{Y}_i^{(l)}\right\}$ *satisfy the conditions of Thm. D.3, in the limit of infinite input size and infinite width, the distribution of the output node* O_c *at initialization converges to:*

$$f_{\mathrm{O}_c}^{(\chi)}(\mathrm{o}) \xrightarrow{|\mathcal{W}|\to\infty} \mathcal{N}\left(\mathrm{o}; \mu_c, \sigma_w^2 \mathit{Var}_\chi\left(\mathrm{Y}_0^{(1)}\right)\right), \tag{16}$$

where $\mathit{Var}_\chi\left(\mathrm{Y}_0^{(1)}\right)$ *is the variance of a generic node in the first layer activations, over the dataset randomness. Moreover, while the variance of the distribution,* $\mathit{Var}_\chi(\mathrm{O}_c)$*, converges with high probability (w.h.p.) to a deterministic value, the mean* μ_c *of the distribution converges in distribution to:*

$$f_{\mu_c}(m) \xrightarrow{|\mathcal{W}|\to\infty} \mathcal{N}\left(m; 0, \sigma_w^2 \mathbb{E}_\chi\left(\mathrm{Y}_0^{(1)}\right)^2\right), \tag{17}$$

where $\mathbb{E}_\chi\left(\mathrm{Y}_0^{(1)}\right)$ *is the mean over the dataset randomness of the activated nodes.*
Here, $|\mathcal{W}| \to \infty$ *refers to the limit where the number of neurons* N_l *in each hidden layer* $l \in \{0, \ldots, L\}$ *tends to infinity.*

Presence and Absence of IGB: Thm. E.1 gives rise to two distinct scenarios:

- **Absence of IGB**: If the mean $\mathbb{E}_\chi\left(\mathrm{Y}_1^{(0)}\right)$ converges to zero as the dataset size increases, the distribution of μ_c converges to a Dirac delta at zero:

$$\lim_{D\to\infty} f_{\mu_c}(m) = \delta(m), \tag{18}$$

  indicating the absence of Initial Guessing Bias (IGB).

- **Presence of IGB**: If $\mathbb{E}_\chi\left(\mathrm{Y}_0^{(1)}\right)$ converges to a finite value, the network exhibits IGB, and:

$$\lim_{D\to\infty} f_{\mu_c}(m) = \mathcal{N}\left(m; 0, \sigma_w^2 \mathbb{E}_\chi\left(\mathrm{Y}_0^{(1)}\right)^2\right). \tag{19}$$

  In this case, both the output node distribution (16) and the center distribution (17) depend only on the first hidden layer statistics, meaning the distributional properties remain unchanged as the network depth increases; therefore, the bias does not amplify with network depth.

In this section, we focus on the effect of activation functions, but we highlight that the analysis can be extended to deal with more complex architectures (*e.g.* when including pooling layers).

E.1.1 Full Batch Case

We begin by proving Theorem E.1 in the full-batch setting. In this case, all datapoints are used to compute the shift and scale estimators for the BN, as defined in Eq. (4). Consequently, the random variables $\mathbf{B}^{(1)}_{;a}$ no longer depend on the sample index a, since all datapoints are part of the same batch. For clarity, we omit the sample index from the notation.
We prove the result by induction, showing that the distribution $f^{(\chi)}_{\mathrm{B}^{(l)}_i}(\mathrm{b})$ remains invariant across layers l. First, we show that:

$$f^{(\chi)}_{\mathrm{B}^{(1)}_i}(\mathrm{b}) = \mathcal{N}(\mathrm{b}; 0, 1). \tag{20}$$

Starting with multivariate Gaussian-distributed inputs, *i.e.*, $\mathrm{X}^{(0)}_i \sim \mathcal{N}(\mu, \sigma^2), \forall i \in [1, \ldots, N_0]$, the nodes will remain normally distributed after passing through the dense layer, since the input undergoes a linear transformation:

$$\mathrm{Z}^{(1)}_i = \sum_{j=1}^{N_0} w^{(1)}_{ij} \mathrm{X}^{(0)}_j, \tag{21}$$

where $\mathrm{Z}^{(1)}_i$ represents the i-th node after the Dense Layer, before passing through the activation function in the first hidden layer. After the linear transformation the input is processed by BN (see recurrent Eq. (3) for the configuration BN **before** activation). The output of Batch Normalizationm for each node i, $\mathrm{B}^{(1)}_i$, follows a standard normal distribution by construction. Specifically,

$$\mathrm{B}^{(1)}_i = \frac{\mathrm{Z}^{(1)}_i - \mathbb{E}_\chi\left(\mathrm{Z}^{(1)}_i\right)}{\sigma_{\mathrm{Z}^{(1)}_i}} \sim \mathcal{N}(0, 1). \tag{22}$$

Next, we establish the inductive step: assuming $f^{(\chi)}_{\mathrm{B}^{(l)}_i}(\mathrm{b}) = \mathcal{N}(\mathrm{b}; 0, 1)$ for all i, we aim to show that $f^{(\chi)}_{\mathrm{B}^{(l+1)}_i}(\mathrm{b}) = \mathcal{N}(\mathrm{b}; 0, 1)$. This will be done via the following sequence of transformations (see Fig. A.1):

$$f^{(\chi)}_{\mathrm{B}^{(l)}_i}(\mathrm{b}) \to f^{(\chi)}_{\mathrm{Y}^{(l)}_i}(\mathrm{y}) \to f^{(\chi)}_{\mathrm{Z}^{(l+1)}_i}(\mathrm{z}) \to f^{(\chi)}_{\mathrm{B}^{(l+1)}_i}(\mathrm{b}).$$

The nodes after BN are identically distributed by hypothesis. Since the activation function applies element-wise to each node, the activated nodes $\{\mathrm{Y}^{(l)}_i\}_{i=1}^{N_l}$ are also identically distributed. Passing through the dense layer combines the activated nodes linearly:

$$\mathrm{Z}^{(l+1)}_i = \sum_{j=1}^{N_l} w^{(l)}_{ij} \mathrm{Y}^{(l)}_j. \tag{23}$$

The expectation over the dataset randomness is:

$$\mathbb{E}_\chi\left(\mathrm{Z}^{(l+1)}_i\right) = \sum_{j=1}^{N_l} w^{(l)}_{ij} \mathbb{E}_\chi\left(\mathrm{Y}^{(l)}_j\right) \overset{\text{IID}}{=} \mathbb{E}_\chi\left(\mathrm{Y}^{(1)}_0\right) \sum_{j=1}^{N_l} w^{(l)}_{ij} \equiv \mathbb{E}_\chi\left(\mathrm{Y}^{(1)}_0\right) S^{(l)}_w, \tag{24}$$

where the independence of $w^{(l)}_{ij}$ and $\mathrm{Y}^{(l)}_j$ is used to extract the mean value $\mathbb{E}_\chi\left(\mathrm{Y}^{(l)}_0\right)$, which is identical for all j. Here, $S^{(l)}_w = \sum_{j=1}^{N_l} w^{(l)}_{ij}$ represents the sum over the weights.

In the last passage, we introduced $S^{(l)}_w$ and used the fact that $f^{(\chi)}_{\mathrm{Y}^{(l)}_i}(\mathrm{y}) = f^{(\chi)}_{\mathrm{Y}^{(1)}_i}(\mathrm{y})$, meaning the distributions of the activated nodes do not change with depth.

The variance, conditional on the weights, is given by:

$$\begin{aligned}\mathrm{Var}_\chi\left(\mathrm{Z}^{(l+1)}_i\right) &= \mathrm{Var}_\chi\left(\sum_{j=1}^{N_l} w^{(l)}_{ij} \mathrm{Y}^{(l)}_j\right) = \sum_{j=1}^{N_l} \left(w^{(l)}_{ij}\right)^2 \mathrm{Var}_\chi\left(\mathrm{Y}^{(l)}_j\right) \\ &\overset{\text{ID}}{=} \mathrm{Var}_\chi\left(\mathrm{Y}^{(l)}_0\right) S^{(l)}_{w^2} \overset{N_l \to \infty}{=} \mathrm{Var}_\chi\left(\mathrm{Y}^{(l)}_0\right) \sigma^2_w,\end{aligned} \tag{25}$$

where $S_{w^2}^{(l)} = \sum_{j=1}^{N_l} \left(w_{ij}^{(l)}\right)^2$. As $N_l \to \infty$, the distribution of $S_{w^2}^{(l)}$ concentrates around a deterministic value Francazi et al. (2024).

Finally, by the Central Limit Theorem (CLT), the distribution of $\mathrm{Z}_i^{(l+1)}$ is:

$$f_{\mathrm{Z}_i^{(l+1)}}^{(\chi)}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z}; \mathbb{E}_\chi\left(\mathrm{Y}_0^{(1)}\right) S_w^{(l)}, \mathrm{Var}_\chi\left(\mathrm{Y}_0^{(1)}\right)\sigma_w^2\right). \tag{26}$$

Remark: The distribution in Eq. 26 is identical to the output distribution of a network with just one ReLU and Dense layer, i.e. a ReLU single hidden layer perceptron without normalization (Francazi et al., 2024).

E.1.2 Mini Batch case

In practice, we often do not have direct access to the underlying dataset statistics; instead, these are estimated over a random subset B of $|B|$ dataset samples (batch) . In particular, we define

- $\hat{\mu}_i^{(l)} = \frac{1}{|B|}\sum_{b\in B(a)} \mathrm{Z}_{i;b}^{(l)}$ is the estimator of $\mathbb{E}_\chi\left[\mathrm{Z}_i^{(l)}\right] = \mu_{\mathrm{Z}_i^{(l)}}$,
- $\hat{\sigma}_i^{(l)} = \sqrt{\frac{1}{|B|}\sum_{b\in B(a)}\left(\mathrm{Z}_{i;b}^{(l)} - \hat{\mu}_i^{(l)}\right)^2}$ is the estimator of the standard deviation $\sigma_{\mathrm{Z}_i^{(l)}}$,
- $\mathrm{B}_{i;a}^{(l)} = \frac{\mathrm{Z}_{i;a}^{(l)} - \hat{\mu}_i^{(l)}}{\hat{\sigma}_i^{(l)}}$ is the BN transformation.

When these estimators are computed over a finite number of samples, B, we can identify the following differences compared to the full batch case (*i.e.* the case when $|B| \to \infty$):

- In the full batch case, the estimators converge to the mean and standard deviation of the distribution ($\mu_{\mathrm{Z}_i^{(l)}}$ and $\sigma_{\mathrm{Z}_i^{(l)}}$), which are constants, while in the mini-batch case, the estimators $\hat{\mu}_i^{(l)}$ and $\hat{\sigma}_i^{(l)}$ are functions of the set $\{\mathrm{Z}_{i;b}^{(l)}\}_{b\in B(a)}$-where $B(a)$ indicates the batch whose a is element of-making the estimators themselves random variables.
- It is also noted that $\hat{\mu}_i^{(l)}$ and $\hat{\sigma}_{\mathrm{Z}_{i;a}^{(1)}}$ are therefore functions of $\mathrm{Z}_{i;a}^{(l)}$ and are thus correlated with it.

However, $\hat{\mu}_i^{(l)}$ and $\hat{\sigma}_i^{(l)}$ are not the only possible choices of estimators capable of solving the problem of internal covariate shift.

We make a small change to the estimators to facilitate the analysis, where we simply discard the datapoint a from the average and standard deviation estimator computed for each $\tilde{\mathrm{B}}_{i;a}^{(l)}$. We will see that the resulting estimator has the same expectation and only the variance is slightly increased. These modified estimators are used to center the variables $\tilde{\mathrm{B}}_{i;a}^{(l)}$ at 0 and scale them so that they have a variance of $\mathcal{O}(1)$, similar to what is done with the classic batch norm. We will examine the differences from the canonical estimators, demonstrating that as the batch size increases, the disparity between the canonical and modified versions becomes negligible.

Specifically, we define the new transformation

$$\tilde{\mathrm{B}}_{i;a}^{(l)} = \frac{\mathrm{Z}_{i;a}^{(l)} - \tilde{\mu}_i^{(l)}}{\tilde{\sigma}_i^{(l)}}, \tag{27}$$

$$\tilde{\mu}_i^{(l)} = \frac{1}{|B|-1}\sum_{\substack{b\in B(a)\\ b\neq a}} \mathrm{Z}_{i;b}^{(l)}, \tag{28}$$

$$\tilde{\sigma}_i^{(l)} = \sqrt{\frac{1}{|B|-1}\sum_{\substack{b\in B(a)\\ b\neq a}}\left(\mathrm{Z}_{i;b}^{(l)} - \tilde{\mu}_i^{(l)}\right)^2}. \tag{29}$$

$\tilde{\mu}_i^{(l)}$ and $\tilde{\sigma}_i^{(l)}$ are alternative estimators. It is straightforward to verify that:

$$\mathbb{E}_\chi\left(\tilde{\mu}_i^{(l)}\right) = \mathbb{E}_\chi\left(\hat{\mu}_i^{(l)}\right) = \mu_{\mathrm{Z}_i^{(l)}}\,, \tag{30}$$

$$\mathbb{E}_\chi\left(\left(\tilde{\sigma}_i^{(l)}\right)^2\right) = \sigma^2_{\mathrm{Z}_i^{(1)}}\left(\frac{|B|-2}{|B|-1}\right)\,. \tag{31}$$

Derivation of Eq. (31)

$$\mathbb{E}_\chi\left(\left(\tilde{\sigma}_i^{(l)}\right)^2\right) = \mathbb{E}_\chi\left(\frac{1}{|B|-1}\sum_{\substack{b\in B(a)\\ b\neq a}}\left(\mathrm{Z}_{i;b}^{(l)} - \tilde{\mu}_i^{(l)}\right)^2\right) \tag{32}$$

$$= \frac{1}{|B|-1}\sum_{\substack{b\in B(a)\\ b\neq a}}\left(\underbrace{\mathbb{E}_\chi\left(\left(\mathrm{Z}_{i;b}^{(l)}\right)^2\right)}_{\mathbf{A}} + \underbrace{\mathbb{E}_\chi\left(\left(\tilde{\mu}_i^{(l)}\right)^2\right)}_{\mathbf{B}} - 2\underbrace{\mathbb{E}_\chi\left(\mathrm{Z}_{i;b}^{(l)}\,\tilde{\mu}_i^{(l)}\right)}_{\mathbf{C}}\right) \tag{33}$$

$$\mathbf{A} = \mathbb{E}_\chi\left(\left(\mathrm{Z}_{i;b}^{(l)}\right)^2\right) = \sigma^2_{\mathrm{Z}_i^{(l)}} + \mu^2_{\mathrm{Z}_i^{(l)}}\,; \tag{34}$$

$$\mathbf{B} = \mathbb{E}_\chi\left(\left(\tilde{\mu}_i^{(l)}\right)^2\right) = \frac{1}{(|B|-1)^2}\sum_{\substack{m\in B(a)\\ m\neq a}}\sum_{\substack{n\in B(a)\\ n\neq a}}\mathbb{E}_\chi\left(\mathrm{Z}_{i;m}^{(l)}\mathrm{Z}_{i;n}^{(l)}\right)$$

$$= \frac{1}{(|B|-1)^2}\left(\sum_{\substack{m,n\in B(a)\\ m\neq a\\ m=n}}\mathbb{E}_\chi\left(\left(\mathrm{Z}_{i;m}^{(l)}\right)^2\right) + \sum_{\substack{m,n\in B(a)\\ m\neq a\\ m\neq n}}\mathbb{E}_\chi\left(\mathrm{Z}_{i;m}^{(l)}\right)\mathbb{E}_\chi\left(\mathrm{Z}_{i;n}^{(l)}\right)\right)$$

$$= \frac{1}{(|B|-1)^2}\left((|B|-1)\left(\sigma^2_{\mathrm{Z}_j^{(l)}} + \mu^2_{\mathrm{Z}_j^{(l)}}\right) + (|B|-1)(|B|-2)\mu^2_{\mathrm{Z}_i^{(l)}}\right) = \frac{\sigma^2_{\mathrm{Z}_j^{(l)}}}{(|B|-1)} + \mu^2_{\mathrm{Z}_j^{(l)}}\,; \tag{35}$$

$$\mathbf{C} = \frac{1}{(|B|-1)}\mathbb{E}_\chi\left(\mathrm{Z}_{i;b}^{(l)}\sum_{\substack{m\in B(a)\\ m\neq a}}\mathrm{Z}_{i;m}^{(l)}\right) = \frac{1}{(|B|-1)}\left(\mathbb{E}_\chi\left(\left(\mathrm{Z}_{i;b}^{(l)}\right)^2\right) + \sum_{\substack{m\in B(a)\\ m\neq a,b}}\mathbb{E}_\chi\left(\mathrm{Z}_{i;b}^{(l)}\right)\mathbb{E}_\chi\left(\mathrm{Z}_{i;m}^{(l)}\right)\right)$$

$$= \frac{1}{(|B|-1)}\left(\left(\sigma^2_{\mathrm{Z}_j^{(l)}} + \mu^2_{\mathrm{Z}_j^{(l)}}\right) + \left((|B|-2)\mu^2_{\mathrm{Z}_j^{(l)}}\right)\right) = \frac{\sigma^2}{(|B|-1)} + \mu^2_{\mathrm{Z}_j^{(l)}}\,. \tag{36}$$

Substituting Eqs. (34), (35) and (36) into Eq. (32) we get:

$$\mathbb{E}_\chi\left(\left(\tilde{\sigma}_i^{(l)}\right)^2\right) = \left(\sigma^2_{\mathrm{Z}_j^{(l)}} + \mu^2_{\mathrm{Z}_j^{(l)}}\right) + \left(\frac{\sigma^2_{\mathrm{Z}_j^{(l)}}}{(|B|-1)} + \mu^2_{\mathrm{Z}_j^{(l)}}\right) - 2\left(\frac{\sigma^2_{\mathrm{Z}_j^{(l)}}}{(|B|-1)} + \mu^2_{\mathrm{Z}_j^{(l)}}\right) = \sigma^2_{\mathrm{Z}_j^{(l)}}\frac{(|B|-2)}{(|B|-1)}\,. \tag{37}$$

Furthermore, it is possible to express the introduced estimators as functions of the random variables $\hat{\mu}_i^{(l)}$ and $\hat{\sigma}_i^{(l)}$ showing that the difference between the two choices tends to 0 (in probability) as the batch size $|B|$ increases. For example:

$$\hat{\mu}_i^{(l)} = \tilde{\mu}_i^{(l)}\left(\frac{|B|-1}{|B|}\right) + \frac{\mathrm{Z}_{i;a}^{(l)}}{|B|} \Rightarrow \Delta_\mu \equiv \hat{\mu}_i^{(l)} - \tilde{\mu}_i^{(l)} = \frac{1}{|B|}\left(\mathrm{Z}_{i;a}^{(l)} - \tilde{\mu}_i^{(l)}\right) \equiv \epsilon_{|B|}\,. \tag{38}$$

Analyzing the random variable $\epsilon_{|B|}$, we can show that the difference between the two estimators converges to 0 in

probability, in particular:

$$\mathbb{E}_\chi\left(\epsilon_{|B|}\right) = \frac{1}{|B|}\left(\mathbb{E}_\chi\left(\mathrm{Z}_{i;a}^{(l)}\right) - \mathbb{E}_\chi\left(\tilde{\mu}_i^{(l)}\right)\right) = \frac{1}{|B|}(\mu_{\mathrm{Z}_i^{(l)}} - \mu_{\mathrm{Z}_i^{(l)}}) = 0\,, \tag{39}$$

$$\mathrm{Var}\left(\epsilon_{|B|}\right) = \frac{1}{|B|^2}\left(\sigma^2_{\mathrm{Z}_i^{(l)}} + \frac{|B|-1}{|B|^2}\sigma^2_{\mathrm{Z}_i^{(l)}}\right) = \mathcal{O}\left(\frac{1}{|B|^2}\right)\,. \tag{40}$$

Similarly, the same can be shown for $\Delta_{\sigma^2} \equiv \hat{\sigma}^2_{\mathrm{Z}_i^{(l)}} - \tilde{\sigma}^2_{\mathrm{Z}_i^{(l)}}$.
We can also derive an expression for the distributions $f_{\tilde{\sigma}_i^{(l)}}(y)$, $f_{\tilde{\mu}_i^{(l)}}(h)$, and $f_{\mathrm{Z}_i^{(l)}}(w)$.

$\boldsymbol{f_{\mathrm{Z}_i^{(l)}}(w)}$**:** Let us assume that the variables we are transforming with the normalization are normally distributed, *i.e.*:

$$f_{\mathrm{Z}_i^{(l)}}(w) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\left(w-\mu_{\mathrm{Z}_i^{(l)}}\right)^2}{2\sigma^2_{\mathrm{Z}^{(l)}}}\right), \tag{41}$$

with $\sigma^2_{\mathrm{Z}_i^{(l)}} = \sigma^2_{\mathrm{Z}^{(l)}}\ \ \forall i$.

We will derive the other relevant distributions, included the p.d.f. of $\tilde{\mathrm{B}}_i^{(l)}$ starting from this assumption. We will then use this result to complete the proof in the IGB setting, *i.e.* assuming random data as input.

Derivation of $\boldsymbol{f_{\tilde{\mu}_i^{(l)}}(h)}$ **:** $\tilde{\mu}$ is a sum of $(|B|-1)$ i.i.d. Gaussian variables [Eq. (28)]; thus, it is also distributed normally. Specifically:

$$f_{\tilde{\mu}_i^{(l)}}(h) = \sqrt{\frac{(|B|-1)}{2\pi\sigma^2_{\mathrm{Z}^{(l)}}}}\exp\left(-\frac{-y^2}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}}}\right)\,. \tag{42}$$

Derivation of $\boldsymbol{f_{\tilde{\sigma}_i^{(l)}}(y)}$ **:** To derive the p.d.f., we start from the sample variance calculated on $(|B|-1)$ elements, which is distributed according to a chi-squared distribution with $(|B|-2)$ degrees of freedom. Specifically, defining

$$S = \left(\tilde{\sigma}_i^{(l)}\right)^2\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}}} = \frac{1}{\sigma^2_{\mathrm{Z}^{(l)}}}\sum_{\substack{j\in B\\ j\neq i}}^{|B|-1}(\mathrm{Z}_j^{(l)} - \tilde{\mu}_i^{(l)})^2\,, \tag{43}$$

we have $S \sim \chi(|B|-2)$. Since S is proportional to $\tilde{\sigma}^2_{\mathrm{Z}_i^{(l)}}$, with a simple change of variables, we obtain

$$f_{\left(\tilde{\sigma}_i^{(l)}\right)^2}(y') = \frac{1}{2^{\frac{(|B|-2)}{2}}\Gamma\left(\frac{|B|-2}{2}\right)}\left(\frac{|B|-1}{\sigma^2_{\mathrm{Z}^{(l)}}}\right)^{\frac{|B|}{2}-1}(y')^{\frac{|B|}{2}-2}\ \exp\left(-\frac{y'}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}}}\right)\,. \tag{44}$$

Finally, with a further change of variables, we obtain

$$f_{\tilde{\sigma}_i^{(l)}}(y) = \frac{1}{2^{\frac{|B|}{2}-2}\Gamma\left(\frac{|B|-2}{2}\right)}\left(\frac{|B|-1}{\sigma^2_{\mathrm{Z}^{(l)}}}\right)^{\frac{|B|}{2}-1}y^{|B|-3}\ \exp\left(-\frac{y^2}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}}}\right)\,. \tag{45}$$

It is noteworthy that the three random variables $\mathrm{Z}_{i;a}^{(l)}$, $\tilde{\mu}_i^{(l)}$, and $\tilde{\sigma}_i^{(l)}$ are independent due to the independence between the sample mean and sample variance of Gaussian variables (See, *e.g.*, Theorem 5.3.1 in Casella and Berger (2002)), and because $\tilde{\mu}_i^{(l)}$ and $\tilde{\sigma}_i^{(l)}$ are functions of a set of variables independent from $\mathrm{Z}_{i;a}^{(l)}$. This independence among the three random variables significantly simplifies the analysis. For example, we can simply calculate

$\mathbb{E}_\chi\left(\tilde{\mathrm{B}}_i^{(l)}\right)$ and $\mathrm{Var}_\chi\left(\tilde{\mathrm{B}}_i^{(l)}\right)$ as follows:

$$\begin{aligned}
\mathbb{E}_\chi\left(\tilde{\mathrm{B}}_i^{(l)}\right) &= \int_0^\infty dy \int_{-\infty}^\infty dx \int_{-\infty}^\infty dw\ \frac{w-x}{y} f_{\tilde{\sigma}_i^{(l)}}(y)\, f_{\tilde{\mu}_i^{(l)}}(x)\, f_{\mathrm{Z}_i^{(l)}}(w) \\
&= \int_0^\infty dy\ \frac{f_{\tilde{\sigma}_i^{(l)}}(y)}{y} \int_{-\infty}^\infty dx\ f_{\tilde{\mu}_i^{(l)}}(x) \int_{-\infty}^\infty dw\ (w-x) f_{\mathrm{Z}_i^{(l)}}(w) \\
&= \int_0^\infty dy\ \frac{f_{\tilde{\sigma}_i^{(l)}}(y)}{y} \int_{-\infty}^\infty dx\ f_{\tilde{\mu}_i^{(l)}}(x)\,(w-\mu_{\mathrm{Z}_i^{(l)}}) \\
&= \int_0^\infty dy\ \frac{f_{\tilde{\sigma}_i^{(l)}}(y)}{y} (\mu_{\mathrm{Z}_i^{(l)}} - \mu_{\mathrm{Z}_i^{(l)}}) = 0\,, \qquad (46)\\
\mathrm{Var}_\chi\left(\tilde{\mathrm{B}}_i^{(l)}\right) &= \mathbb{E}_\chi\left(\left(\tilde{\mathrm{B}}_i^{(l)}\right)^2\right) = \int_0^\infty dy \int_{-\infty}^\infty dx \int_{-\infty}^\infty dw\ \frac{(w-x)^2}{y^2} f_{\tilde{\sigma}_i^{(l)}}(y)\, f_{\tilde{\mu}_i^{(l)}}(x)\, f_{\mathrm{Z}_i^{(l)}}(w) \\
&= \int_0^\infty dy\ \frac{f_{\tilde{\sigma}_i^{(l)}}(y)}{y^2} \int_{-\infty}^\infty dx\ f_{\tilde{\mu}_i^{(l)}}(x) \int_{-\infty}^\infty dw\ f_{\mathrm{Z}_i^{(l)}}(w)\,(w^2+x^2-2xw) \\
&= \int_0^\infty dy\ \frac{1}{2^{\frac{|B|}{2}-2}\Gamma\left(\frac{|B|-2}{2}\right)} \left(\frac{|B|-1}{\sigma^2_{\mathrm{Z}^{(l)}}}\right)^{\frac{|B|}{2}-1} y^{|B|-5} \exp\left(-\frac{y^2}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}}}\right) \cdot \\
&\cdot \left(\sigma^2_{\mathrm{Z}^{(l)}} + \mu^2_{\mathrm{Z}_i^{(l)}} + \frac{\sigma^2_{\mathrm{Z}^{(l)}}}{|B|-1} + \mu^2_{\mathrm{Z}_i^{(l)}} - 2\mu^2_{\mathrm{Z}_i^{(l)}}\right) \\
&\overset{\boldsymbol{a}}{=} \sigma^2_{\mathrm{Z}^{(l)}} \frac{|B|}{|B|-1} \frac{1}{2^{\frac{|B|}{2}-2}\Gamma\left(\frac{|B|-2}{2}\right)} \left(\frac{|B|-1}{\sigma^2_{\mathrm{Z}^{(l)}}}\right)^{\frac{|B|}{2}-1} \frac{\Gamma\left(\frac{|B|-4}{2}\right)}{2} \left(\frac{2\sigma^2_{\mathrm{Z}^{(l)}}}{(|B|-1)}\right)^{\frac{|B|-4}{2}} \\
&= \frac{|B|}{|B|-4}\,. \qquad (47)
\end{aligned}$$

where in step $\boldsymbol{a}$ we used the identity

$$\int_0^\infty x^n e^{-ax^b} dx = \frac{\Gamma\left(\frac{n+1}{b}\right)}{b a^{\frac{(n+1)}{b}}}\,. \qquad (48)$$

The use of the estimators $\tilde{\mu}_i^{(l)}$ and $\tilde{\sigma}_i^{(l)}$ greatly simplifies the analysis and allows us, for example, to derive the p.d.f. of the random variable $\tilde{\mathrm{B}}_i^{(l)}$, distribution $f^{(\chi)}_{\tilde{\mathrm{B}}_i^{(l)}}(z)$, defined by the transformation in Eq. (27). Specifically:

$$
\begin{aligned}
f^{(\chi)}_{\tilde{\mathrm{B}}^{(l)}_i}\left(\tilde{\mathrm{b}}\right) &= \int_0^\infty dy \int_{-\infty}^\infty dx \int_{-\infty}^\infty dw \delta\left(\tilde{\mathrm{b}} - \frac{(w-x)}{y}\right) f_{\tilde{\sigma}^{(l)}_i}(y)\, f_{\tilde{\mu}^{(l)}_i}(x)\, f_{\mathrm{Z}^{(l)}_i}(w) \\
&= C \int_0^\infty dy \int_{-\infty}^\infty dx \int_{-\infty}^\infty dw\ \delta\left(\tilde{\mathrm{b}} - \frac{(w-x)}{y}\right) \cdot \\
&\exp\left(-\frac{(w-\mu_{\mathrm{Z}^{(l)}_i})^2}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}\right) \exp\left(-\frac{(\mathrm{y}-\mu_{\mathrm{Z}^{(l)}_i})^2}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}(|B|-1)\right) y^{|B|-3} \exp\left(-\frac{y^2}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}_i}}\right) \\
&= C \int_0^\infty dy\ y^{|B|-3} e^{-\frac{y^2}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}_i}}}\ y \int_{-\infty}^\infty dx \int_{-\infty}^\infty dw\ \delta\left(\tilde{\mathrm{b}}y - (w-x)\right) \cdot \\
&\exp\left(-\frac{(w-\mu_{\mathrm{Z}^{(l)}_i})^2}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}\right) \exp\left(-\frac{(\mathrm{y}-\mu_{\mathrm{Z}^{(l)}_i})^2}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}(|B|-1)\right) \\
&= C \int_0^\infty dy\ y^{|B|-2} \exp\left(-\frac{-y^2}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}_i}}\right) \int_{-\infty}^\infty dx \int_{-\infty}^\infty dw\ \delta\left(w - (\tilde{\mathrm{b}}y + x)\right) \cdot \\
&\exp\left(-\frac{(w-\mu_{\mathrm{Z}^{(l)}_i})^2}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}\right) \exp\left(-\frac{(\mathrm{y}-\mu_{\mathrm{Z}^{(l)}_i})^2}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}(|B|-1)\right) \\
&= C \int_0^\infty dy\ y^{|B|-2} \exp\left(-\frac{y^2}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}_i}}\right) \cdot \\
&\int_{-\infty}^\infty dx\ \exp\left(-\frac{(\tilde{\mathrm{b}}y + x - \mu_{\mathrm{Z}^{(l)}_i})^2}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}\right) \exp\left(-\frac{(\mathrm{y}-\mu_{\mathrm{Z}^{(l)}_i})^2}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}(|B|-1)\right) \\
&= C \int_0^\infty dy\ y^{|B|-2} \exp\left(-\frac{y^2}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}_i}}\right) \cdot \\
&\int_{-\infty}^\infty dx\ \exp\left(-\frac{1}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}(\tilde{\mathrm{b}}^2 y^2 + x^2 + 2xy\tilde{\mathrm{b}} + (|B|-1)x^2)\right) \\
&= C \int_0^\infty dy\ y^{|B|-2} \exp\left(-\frac{y^2}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}_i}}\right) \cdot \\
&\int_{-\infty}^\infty dx\ \exp\left(-\frac{1}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}\left(\tilde{\mathrm{b}}^2 y^2\left(1 - \frac{1}{|B|}\right)\right)\right) \exp\left(-\frac{1}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}\left(\sqrt{|B|}x + \frac{y\tilde{\mathrm{b}}}{\sqrt{|B|}}\right)^2\right) \\
&= C' \int_0^\infty dy\ y^{|B|-2} \exp\left(-\frac{y^2}{2}\frac{(|B|-1)}{\sigma^2_{\mathrm{Z}^{(l)}_i}}\left(1 + \frac{\tilde{\mathrm{b}}^2}{|B|}\right)\right) \\
&= C' \left(\frac{(|B|-1)}{2\sigma^2_{\mathrm{Z}^{(l)}_i}}\right)^{-\frac{(|B|-1)}{2}} \left(1 + \frac{\tilde{\mathrm{b}}^2}{|B|}\right)^{-\frac{(|B|-1)}{2}} \Gamma\left(\frac{|B|-1}{2}\right)\frac{1}{2} \\
&= \sqrt{\frac{1}{|B|\pi}} \frac{\Gamma\left(\frac{|B|-1}{2}\right)}{\Gamma\left(\frac{|B|-2}{2}\right)} \left(1 + \frac{\tilde{\mathrm{b}}^2}{|B|}\right)^{-\frac{(|B|-1)}{2}}, \qquad (49)
\end{aligned}
$$

where

$$C = \frac{\sqrt{|B|-1}}{(2\pi\sigma^2_{\mathrm{Z}_i^{(l)}})} \frac{1}{2^{\frac{|B|}{2}-2}\Gamma\left(\frac{|B|-2}{2}\right)} \left(\frac{|B|-1}{\sigma^2_{\mathrm{Z}_i^{(l)}}}\right)^{\frac{|B|}{2}-1}$$

$$C' = C\sqrt{\frac{2\pi\sigma^2_{\mathrm{Z}_i^{(l)}}}{|B|}}. \tag{50}$$

The distribution outlined in Eq. (49) is expected to converge to the Gaussian profile observed in the full batch scenario. To rigorously demonstrate this assertion, we explicitly expand the binomial term in Eq. (49) in the limit as the batch size $|B|$ approaches infinity. This expansion allows us to directly compare each power of $\tilde{\mathrm{b}}$ in the binomial series with the corresponding term in the Taylor series expansion of the Gaussian profile (centered around zero) that emerges in the full-batch case. By examining each power individually, we show that, in the full batch limit, every term in the expanded form of Eq. (49) precisely matches its counterpart in the Gaussian Taylor series. This one-to-one correspondence confirms the convergence of the expression to a standard Gaussian as $|B| \to \infty$.

To demonstrate that the expression in Eq. (49) converges to a Gaussian as $|B| \to \infty$, we expand the term

$$\left(1 + \frac{\tilde{\mathrm{b}}^2}{|B|}\right)^{-\frac{|B|-1}{2}}$$

using the binomial series expansion. For any $\alpha \in \mathbb{R}$, the binomial expansion for $(1+x)^\alpha$ is

$$(1+x)^\alpha = \sum_{k=0}^{\infty} \binom{\alpha}{k} x^k,$$

where $\binom{\alpha}{k} = \frac{\alpha(\alpha-1)\cdots(\alpha-k+1)}{k!}$. In our case, we set $x = \frac{\tilde{\mathrm{b}}^2}{|B|}$ and $\alpha = -\frac{|B|-1}{2}$, giving

$$\left(1 + \frac{\tilde{\mathrm{b}}^2}{|B|}\right)^{-\frac{|B|-1}{2}} = \sum_{k=0}^{\infty} \binom{-\frac{|B|-1}{2}}{k} \left(\frac{\tilde{\mathrm{b}}^2}{|B|}\right)^k.$$

The general term in this expansion is

$$\binom{-\frac{|B|-1}{2}}{k} \left(\frac{\tilde{\mathrm{b}}^2}{|B|}\right)^k.$$

Using the definition of the binomial coefficient for negative exponents, we have

$$\binom{-\frac{|B|-1}{2}}{k} = \frac{(-1)^k \left(\frac{|B|-1}{2}\right)\left(\frac{|B|-1}{2}+1\right)\cdots\left(\frac{|B|-1}{2}+k-1\right)}{k!}.$$

For large $|B|$, each term in the product $\left(\frac{|B|-1}{2}+j\right) \approx \frac{|B|}{2}$, so we approximate

$$\binom{-\frac{|B|-1}{2}}{k} \approx \frac{(-1)^k}{k!}\left(\frac{|B|}{2}\right)^k.$$

Substituting this approximation into the general term, we find

$$\binom{-\frac{|B|-1}{2}}{k} \left(\frac{\tilde{\mathrm{b}}^2}{|B|}\right)^k \approx \frac{(-1)^k}{k!} \frac{\left(\frac{|B|}{2}\right)^k \left(\frac{\tilde{\mathrm{b}}^2}{|B|}\right)^k}{|B|^k} = \frac{(-1)^k}{k!} \frac{\tilde{\mathrm{b}}^{2k}}{2^k}.$$

Thus, in the limit $|B| \to \infty$, we recover the series expansion of the Gaussian function:

$$\exp\left(-\frac{\tilde{\mathrm{b}}^2}{2}\right) = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!} \left(\frac{\tilde{\mathrm{b}}^2}{2}\right)^k.$$

This demonstrates that each term in the expansion of Eq. (49) matches the corresponding term in the Gaussian Taylor series in the full batch limit $|B| \to \infty$.

Finally, let us consider the prefactor in Eq. (49),

$$\sqrt{\frac{1}{|B|\pi}} \frac{\Gamma\left(\frac{|B|-1}{2}\right)}{\Gamma\left(\frac{|B|-2}{2}\right)}.$$

Using Stirling's approximation for large $|B|$, we have

$$\Gamma\left(\frac{|B|-1}{2}\right) \approx \sqrt{2\pi} \left(\frac{|B|-1}{2}\right)^{\frac{|B|-1}{2}-\frac{1}{2}} e^{-\frac{|B|-1}{2}},$$

and similarly,

$$\Gamma\left(\frac{|B|-2}{2}\right) \approx \sqrt{2\pi} \left(\frac{|B|-2}{2}\right)^{\frac{|B|-2}{2}-\frac{1}{2}} e^{-\frac{|B|-2}{2}}.$$

Thus,

$$\frac{\Gamma\left(\frac{|B|-1}{2}\right)}{\Gamma\left(\frac{|B|-2}{2}\right)} \approx \sqrt{\frac{|B|}{2}}.$$

Combining this result with the factor $\sqrt{\frac{1}{|B|\pi}}$ gives

$$\sqrt{\frac{1}{|B|\pi}} \cdot \sqrt{\frac{|B|}{2}} = \frac{1}{\sqrt{2\pi}},$$

which is precisely the normalization factor of a standard Gaussian distribution.

Therefore, in the limit $|B| \to \infty$, Eq. (49) converges to

$$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\tilde{\mathrm{b}}^2}{2}\right),$$

confirming that the expression approaches a standard Gaussian distribution as $|B| \to \infty$.

Figure E.1 illustrates a comparison between the empirical distribution of Gaussian random variables after processing through a batch normalization layer and the theoretical p.d.f. as derived from Eq.(49). As expected, increasing the batch size progressively minimizes the differences between these distributions until they become virtually identical. Figure E.2 shows how the disparity between the distributions increases from the center toward the tails. It is critical to recognize that the tails of the distribution cannot be fully explored empirically due to the dataset's finite size. Thus, with a limited dataset, as the batch size increases, a point is reached where the differences between the distributions are no longer observable within the empirically accessible range. This phenomenon is depicted in Figure E.2 (right), where the empirically accessible region covers the interval $[-5, 5]$. In this interval, the distributions are perfectly aligned.

With the p.d.f. $f^{(\chi)}_{\tilde{\mathrm{B}}_i^{(l)}}\left(\tilde{\mathrm{b}}\right)$ established, the following process is undertaken:

I. Begin with datapoints that are normally distributed.

II. Input these datapoints (organized into mini-batches) into a fully connected layer (using Gaussian weights).

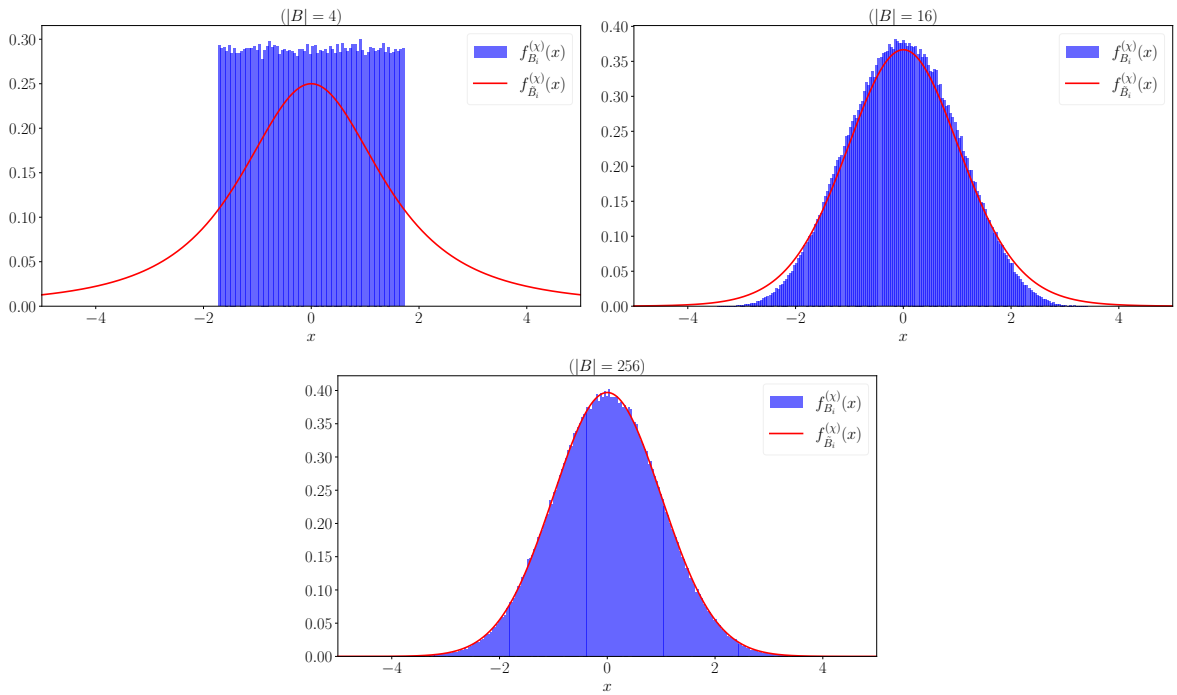

Figure E.1: Comparison, for different batch sizes, between the empirical p.d.f. of $\mathrm{B}_i^{(l)}$ (following the canonical batch norm transformation) and the theoretical p.d.f. of $\tilde{\mathrm{B}}_i^{(l)}$ [Eq.(27)].

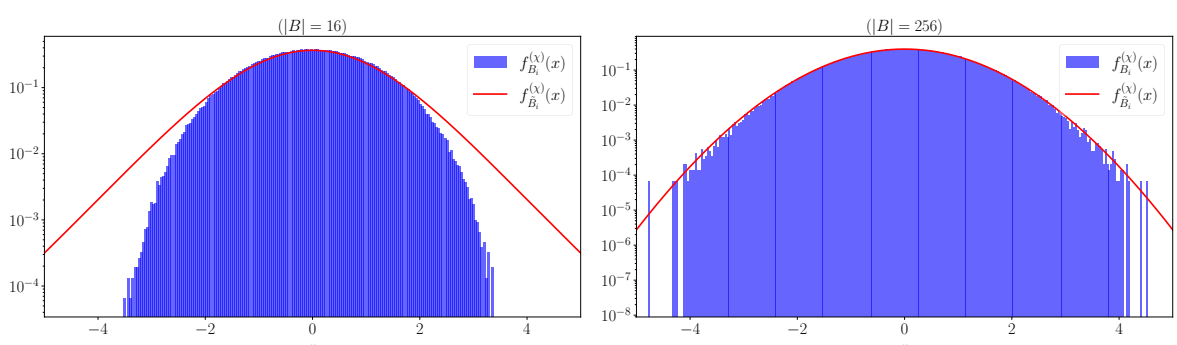

Figure E.2: Same comparison of Fig. E.1 in log scale, to better visualize the tails density.

III. Process these through the batch normalization layer.

IV. Process through the activation function (ReLU).

V. (For deep architectures, iteratively repeat steps II through IV.)

VI. Repeat step II to link the last hidden layer to the output layer.

VII. Obtain the $f_{G_0}(g_0)$.

We will follow now the above propagation process step by step deriving the output nodes distribution. We will show that the distributions of the output nodes do not change with depth, *i.e.* IGB does not amplify with depth. Let us start considering a random vector as input, whose components $\{\mathrm{X}_i^{(0)}\}_{i=1}^{N_0}$ are distribuited according to

$$\mathrm{X}_i^{(0)} \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right) . \tag{51}$$

After passing throught the fully connected layer we will have a set of random variables $\{\mathrm{Z}_i^{(1)}\}_{i=1}^{N_1}$, with

$$\mathrm{Z}_i^{(1)} = \sum_j w_{ij}^{(1)} \mathrm{X}_i^{(0)} , \tag{52}$$

where

$$w_{ij}^{(l)} \sim \mathcal{N}\left(0, \frac{\sigma_w^2}{N_{l-1}}\right) \forall\, l, i, k \tag{53}$$

are random weights fixed with the initialization; this means that for a given initialization they will form a set of random coefficients. Therefore, from Eq. (52), $\mathrm{Z}_i^{(1)}$ is a linear combination of independent Gaussian random variables, meaning that will be itself normally distributed. In particular:

$$\mathrm{Z}_i^{(1)} \sim \mathcal{N}\left(\sum_j w_{ij}^{(1)} \mu_j, \sum_j (w_{ij}^{(1)} \sigma_j)^2\right) . \tag{54}$$

Note that, for our analysis we don't need to compute explicitly the mean and variance of the distribution in Eq. (54). In fact, as a next step, we have to process the set of pre-activated random variables $\{\mathrm{Z}_i^{(1)}\}_{i=1}^{N_1}$ through the BN layer. We derived a proxy for the distribution (Eq. (49)) for the processed variables. This distribution only assumes i.i.d. Gaussian variables for the batches in input in our layer. Remarkably the p.d.f. from Eq. (49) does not depend on the mean and variance of the input random variables but only on the batch size $|B|$. This means that, if we are in the regime where Eq. (49) well approximates the distribution of the hidden layer node after the BN ($|B| \gg 1$), $\mathrm{B}_i^{(1)}$, then

$$f_{\mathrm{B}_i^{(1)}}^{(\chi)}(\mathrm{b}) \approx f_{\tilde{\mathrm{B}}_i^{(1)}}^{(\chi)}(\mathrm{b}) = \sqrt{\frac{1}{|B|\pi}} \frac{\Gamma\left(\frac{|B|-1}{2}\right)}{\Gamma\left(\frac{|B|-2}{2}\right)} \left(1 + \frac{\mathrm{b}^2}{|B|}\right)^{-\frac{(|B|-1)}{2}} . \tag{55}$$

Therefore, after passing through the ReLU, using $\tilde{\mathrm{B}}_i^{(1)}$ and the corresponding distribution as proxy for $\mathrm{B}_i^{(l)}$, we will have

$$f_{\mathrm{Y}_i^{(1)}}^{(\chi)}(\mathrm{y}) = \frac{1}{2}\delta(\mathrm{y}) + \Theta(\mathrm{y}) \sqrt{\frac{1}{|B|\pi}} \frac{\Gamma\left(\frac{|B|-1}{2}\right)}{\Gamma\left(\frac{|B|-2}{2}\right)} \left(1 + \frac{\mathrm{y}^2}{|B|}\right)^{-\frac{(|B|-1)}{2}} . \tag{56}$$

Note that $\{\mathrm{Y}_i^{(1)}\}_{i=1}^{N_1}$ are identically distributed; from Eq. (56) we can compute the average over the dataset

randomness that will be the same for each node; in particular:

$$
\begin{aligned}
\mathbb{E}_\chi\left(\mathrm{Y}_i^{(1)}\right) =& \sqrt{\frac{1}{|B|\pi}}\frac{\Gamma\left(\frac{|B|-1}{2}\right)}{\Gamma\left(\frac{|B|-2}{2}\right)}\int_0^\infty dz\, z\left(1+\frac{z^2}{|B|}\right)^{-\frac{(|B|-1)}{2}} \\
=& \sqrt{\frac{1}{|B|\pi}}\frac{\Gamma\left(\frac{|B|-1}{2}\right)}{\Gamma\left(\frac{|B|-2}{2}\right)}\frac{|B|}{2}\int_0^\infty d\left(1+\frac{z^2}{|B|}\right)\left(1+\frac{z^2}{|B|}\right)^{-\frac{(|B|-1)}{2}} \\
=& \sqrt{\frac{1}{|B|\pi}}\frac{\Gamma\left(\frac{|B|-1}{2}\right)}{\Gamma\left(\frac{|B|-2}{2}\right)}\frac{|B|}{2}\frac{2}{-|B|+3}\left(1+\frac{z^2}{|B|}\right)^{-\frac{(|B|-3)}{2}}\Bigg|_0^\infty \\
=& \sqrt{\frac{1}{|B|\pi}}\frac{\Gamma\left(\frac{|B|-1}{2}\right)}{\Gamma\left(\frac{|B|-2}{2}\right)}\frac{|B|}{|B|-3}\,.
\end{aligned}
\tag{57}
$$

Using Eq. (47) and the fact that, by symmetry,

$$
\mathbb{E}_\chi\left(\left(\mathrm{Y}_i^{(1)}\right)^2\right) = \frac{\mathrm{Var}\left(\mathrm{B}_i^{(1)}\right)}{2} \approx \frac{\mathrm{Var}\left(\tilde{\mathrm{B}}_i^{(1)}\right)}{2}\,, \tag{58}
$$

we have

$$
\mathrm{Var}_\chi\left(\mathrm{Y}_i^{(1)}\right) \approx \frac{|B|}{2(|B|-4)} - \left(\sqrt{\frac{1}{|B|\pi}}\frac{\Gamma\left(\frac{|B|-1}{2}\right)}{\Gamma\left(\frac{|B|-2}{2}\right)}\frac{|B|}{|B|-3}\right)^2\,. \tag{59}
$$

Note that the r.h.s. of Eq. (59) do not dependent from the node index i. We indicate with $\mathrm{Var}_\chi\left(\mathrm{Y}_0^{(1)}\right)$ the value equal for each node, to highlight the absence of dependence from the index. Repeating the analysis analogue to the one outlined in Ref. Francazi et al. (2024) (in particular see App. D) we get, for a single hidden layer architecture:

$$
f_{\mathrm{O}_c}^{(\chi)}(\mathrm{o}) = \mathcal{N}\left(\mathrm{o};\mu_c,\sigma_w^2\mathrm{Var}_\chi\left(\mathrm{Y}_0^{(1)}\right)\right)\,, \tag{60}
$$

$$
f_{\mu_c}(\mathrm{y}) = \mathcal{N}\left(m;0,\sigma_w^2\mathbb{E}_\chi\left(\mathrm{Y}_0^{(1)}\right)^2\right)\,. \tag{61}
$$

Deep architectures We can then consider deeper architectures, increasing the number of hidden layers. Each hidden layer is defined by the same sequence of sub-modules: fully connection, BN, ReLU. Thus we can repeat, for each layer, the analysis of the single hidden layer case using as input the distribution of the output of the previous layer. For example for the second layer the we will repeat the same analysis using Eq. (60) as input distribution. Note that this is also a Gaussian distribution. We derived the p.d.f. of the set of random variables after passing through the BN transformation. In particular from Eq. (49) we can see that, assuming Gaussian i.i.d. random variables in input, the distribution of $\mathrm{B}_i^{(l)}$ do not depend on the mean and variance of the input. Therefore, is easy to proof (*e.g.* by induction) that the distribution after the batch norm will be the same for each layer. This implies that the output nodes will follow Eq. (60) and Eq. (61) irrespective to the depth of the network.

The presence of BN before the activation function keeps the level of IGB constant with the increasing of depth.

E.2 ReLU + BN

We investigate the impact of applying Batch Normalization (BN) after the ReLU (ReLU + BN), in contrast to the more conventional placement before ReLU (BN + ReLU). We show that this modification significantly alters the statistical properties of the network's initial outputs, promoting more balanced and unbiased predictions at initialization. Following the analysis presented in Appendix E.1, we first consider the full-batch regime and subsequently extend our results to the mini-batch setting.

E.2.1 ReLU + BN: Full Batch

We start by analyzing the full-batch setting, deferring the generalization to the mini-batch case to App. E.2.2. We show that applying BN after the activation function suppresses IGB, regardless of network depth. Specifically, we prove that the output nodes are identically distributed and centered at zero, resulting in class-balanced predictions at initialization.

The following is a formal formulation of Th. 5.2.

Theorem E.2 (ReLU + BN)**.** *Consider the setting described in Sec. B with an MLP where BN is applied* after *the ReLU activation function, and assume full-batch normalization statistics. Then, in the limit of infinite width and dataset size, the output distribution $f^{(\chi)}_{\mathrm{O}_c}(\mathrm{o})$ converges to:*

$$f^{(\chi)}_{\mathrm{O}_c}(\mathrm{o}) \xrightarrow{|\mathcal{W}|\to\infty} \mathcal{N}\left(\mathrm{o}; 0, \sigma_w^2\right), \tag{62}$$

with the corresponding mean distribution collapsing to:

$$f_{\mu_c}(m) \xrightarrow{|\mathcal{W}|\to\infty} \delta(m), \tag{63}$$

indicating the suppression of IGB and the emergence of balanced initial predictions.

To prove Thm. E.2 we will follow once again the induction principle. In this case we want to show that the output nodes are identically distributed. In particular we will show that after each Dense layer, nodes are all centered in 0:

$$\begin{cases} f^{(\chi)}_{\mathrm{Z}_i^{(1)}}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z}; 0, \sigma_w^2\right), \\ f^{(\chi)}_{\mathrm{Z}_i^{(l)}}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z}; 0, \sigma_w^2\right) \Longrightarrow f^{(\chi)}_{\mathrm{Z}_i^{(1+1)}}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z}; 0, \sigma_w^2\right). \end{cases} \tag{64}$$

Base step As stated above, our input data have distribution $\mathrm{X}_i^{(0)} \sim \mathcal{N}(\mathrm{z}; 0, 1)$ and they are IID. Also in this case it is straightforward to extend the analysis to arbitrary mean and variance for the input, reaching the same conclusions.
After passing through the weights, the data will undergo the following linear transformation:

$$\mathrm{Z}_i^{(1)} = \sum_{j=1}^{D} w_{ij}^{(0)} \mathrm{X}_j^{(0)}, \tag{65}$$

Now, the mean over the data of $\mathrm{Z}_i^{(1)}$ is

$$\mathbb{E}_\chi\left(\mathrm{Z}_i^{(1)}\right) = \mathbb{E}_\chi\left(\sum_{j=1}^{D} w_{ij}^{(0)} \mathrm{X}_j^{(0)}\right) \overset{\text{Ind.}}{=} \sum_{j=1}^{D} w_{ij}^{(0)} \mathbb{E}_\chi\left(\mathrm{X}_j^{(0)}\right) = 0, \tag{66}$$

while the variance over the data is

$$\mathrm{Var}_\chi\left(\mathrm{Z}_i^{(1)}\right) = \mathrm{Var}_\chi\left(\sum_{j=1}^{D} w_{ij}^{(0)} \mathrm{X}_j^{(0)}\right) = \sum_{j=1}^{D} \left(w_{ij}^{(0)}\right)^2 \mathrm{Var}_\chi\left(\mathrm{X}_j^{(0)}\right) = D\frac{\sigma_w^2}{D} = \sigma_w^2. \tag{67}$$

After passing through the weights, the Dense output will distribute as

$$f^{(\chi)}_{\mathrm{Z}^{(1)}_i}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z}; 0, \sigma_w^2\right). \tag{68}$$

Inductive step I: Dense → ReLU The effect of the ReLU is

$$f^{(\chi)}_{\mathrm{Y}^{(l)}_i}(\mathrm{y}) = \frac{1}{2}\delta(\mathrm{y}) + \Theta(\mathrm{y})\mathcal{N}(\mathrm{y}; 0,\, \sigma_w^2). \tag{69}$$

Inductive step II. ReLU → BN We know that Batch Normalization has the following effect on the outputs:

$$\mathrm{B}^{(l)}_i = \frac{\mathrm{Y}^{(l)}_i - \mathbb{E}_\chi\left(\mathrm{Y}^{(l)}_i\right)}{\mathrm{Var}_\chi\left(\mathrm{Y}^{(l)}_i\right)}, \tag{70}$$

In order to evaluate the distribution of $\mathrm{B}^{(l)}_i$, we need to compute the expected value and the variance of $\mathrm{Y}^{(l)}_i$ as follows:

$$\begin{aligned}\mathbb{E}_\chi\left(\mathrm{Y}^{(l)}_i\right) &= \int_{\mathbb{R}} \left(\frac{1}{2}\delta(\mathrm{y}) + \Theta(\mathrm{y})\mathcal{N}\left(\mathrm{y}; 0, \sigma_w^2\right)\right) \mathrm{y}\, dy = \\ &= \underbrace{\frac{1}{2}\int_{\mathbb{R}} \delta(\mathrm{y})\mathrm{y}\, dy}_{=0} + \frac{1}{\sqrt{2\pi\sigma_w^2}} \int_0^\infty \exp\left(-\frac{1}{2}\left(\frac{\mathrm{y}}{\sigma_w}\right)^2\right) \mathrm{y}\, dy = \frac{\sigma_w}{\sqrt{2\pi}},\end{aligned} \tag{71}$$

$$\begin{aligned}\mathbb{E}_\chi\left(\left(\mathrm{Y}^{(l)}_i\right)^2\right) &= \int_{\mathbb{R}} \left(\frac{1}{2}\delta(\mathrm{y}) + \Theta(\mathrm{y})\mathcal{N}\left(\mathrm{y}; 0, \sigma_w^2\right)\right) \mathrm{y}^2\, dy = \\ &= \underbrace{\frac{1}{2}\int_{\mathbb{R}} \delta(\mathrm{y})\mathrm{y}^2\, dy}_{=0} + \frac{1}{\sqrt{2\pi\sigma_w^2}} \int_0^\infty \exp\left(-\frac{1}{2}\left(\frac{\mathrm{y}}{\sigma_w}\right)^2\right) \mathrm{y}^2\, dy = \frac{\sigma_w^2}{2},\end{aligned} \tag{72}$$

$$\mathrm{Var}_\chi\left(\mathrm{Y}^{(l)}_i\right) = \mathbb{E}_\chi\left(\left(\mathrm{Y}^{(l)}_i\right)^2\right) - \mathbb{E}_\chi\left(\mathrm{Y}^{(i)}_l\right)^2 = \frac{\sigma_w^2}{2} - \frac{\sigma_w^2}{2\pi} = \frac{\sigma_w^2}{2}\left(1 - \frac{1}{\pi}\right). \tag{73}$$

So the expected value and standard deviation are $\mathbb{E}_\chi\left(\mathrm{Y}^{(l)}_i\right) = \frac{\sigma_w}{\sqrt{2\pi}}$ and $\mathrm{Var}_\chi\left(\mathrm{Y}^{(l)}_i\right) = \frac{\sigma_w^2}{2} - \frac{\sigma_w^2}{2\pi}$,

After the normalization, the output will have a distribution with zero mean and standard deviation equal to 1.

For simplicity let's introduce the following notation: we define the parameters of the previous ReLU's output as $\mathbb{E}_\chi\left(\mathrm{Y}^{(l)}_i\right) = a$, $\sigma_{\mathrm{Y}^{(l)}_i} = \alpha$.

The distribution of the BN output is now the following:

$$\begin{aligned}f^{(\chi)}_{\mathrm{B}^{(l)}_i}(\mathrm{b}) &= \frac{1}{2}\delta\left(\frac{\mathrm{b} - a}{\alpha}\right) + \Theta\left(\frac{\mathrm{b} - a}{\alpha}\right)\mathcal{N}\left(\mathrm{b}; a, \frac{\sigma_w^2}{\alpha^2}\right) \\ &= \frac{\sigma_w\sqrt{\pi - 1}}{2}\delta\left(\mathrm{b}\sqrt{2\pi} - \sigma_w\right) + \Theta\left(\mathrm{b}\sqrt{2\pi} - \sigma_w\right)\mathcal{N}\left(\mathrm{b}; \frac{\sigma_w}{\sqrt{2\pi}}, \frac{2\pi}{(\pi - 1)}\right).\end{aligned} \tag{74}$$

From the above formula, one can easily verify that $f^{(\chi)}_{\mathrm{B}_i^{(l)}}(\mathrm{b})$ is a distribution centered at zero with unit variance.

III. BN → Dense
Note that the random variables $\{\mathrm{B}_i^{(l)}\}$'s are IID and have zero mean and unitary variance.
It means that, when we pass the output through the Dense Layer, the final output will have the following form:

$$\mathrm{Y}_i^{(l+1)} = \sum_{j=1}^{N_l} w_{ij}^{(l)} \mathrm{B}_j^{(l)}, \tag{75}$$

therefore, from the CLT,

$$f^{(\chi)}_{\mathrm{Z}_i^{(l+1)}}(\mathrm{z}) \xrightarrow{N_l \to \infty} \mathcal{N}\left(\mathrm{z}; \mathbb{E}_\chi\left(\sum_{j=1}^{N_l} w_{ij}^{(l)} \mathrm{B}_j^{(l)}\right), \mathrm{Var}_\chi\left(\sum_{j=1}^{N_l} w_{ij}^{(l)} \mathrm{B}_j^{(l)}\right)\right). \tag{76}$$

Given that what is stated in Eq. (24) and Eq. (25) applies also to our case, then

$$\mathbb{E}_\chi\left(\mathrm{Z}_j^{(l+1)}\right) = \mathbb{E}_\chi\left(\mathrm{B}_0^{(l)}\right) \sum_{j=1}^{N_1} w_{ij}^{(l)} = 0, \tag{77}$$

$$\mathrm{Var}_\chi\left(\mathrm{Z}_j^{(l+1)}\right) = \underbrace{\mathbb{E}_\chi\left(\left(\mathrm{B}_0^{(l)}\right)^2\right)}_{=1} S_{w^2}^{(l)} = \sigma_w^2. \tag{78}$$

The reason why the expected value above is 1 is that

$$\mathbb{E}_\chi\left(\left(\mathrm{B}_0^{(l)}\right)^2\right) = \mathrm{Var}_\chi\left(\mathrm{B}_0^{(l)}\right) - \mathbb{E}_\chi\left(\left(\mathrm{B}_0^{(l)}\right)\right)^2 = 1 - 0 \tag{79}$$

We can therefore write the distribution of the hidden units at any layer $l+1$ as:

$$f^{(\chi)}_{\mathrm{Z}_i^{(l+1)}}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z}; 0, \sigma_w^2\right). \tag{80}$$

This conclusion extends to the output layer as well, since $\mathrm{O}_c = \mathrm{Z}_c^{(L+1)}$. As a result, all output nodes are identically distributed and centered at zero. This symmetry implies that, at initialization, each class is predicted with equal probability, and therefore *no initial guessing bias* is present. The model makes balanced predictions across the dataset, regardless of its depth.

E.2.2 ReLU + BN: Mini Batch

When BN is placed after the activation function we are in the following settings:

I. Start from normally distributed datapoints.

II. Pass inputs (divided in mini-batches) through a fully connected layer (Gaussian weights).

III. Pass through the activation function (ReLU).

IV. Pass through the BN layer.

V. (Repeat iteratively steps from II. to IV. in case of deep architectures).

VI. Repeat step II. to connect the last hidden layer to the output layer.

VII. Get the $f_{G_0}(g_0)$.

In this case we don't need to explicitly compute the distribution. In fact at the end of each hidden layer–and in particular at the end of the last one– we have the presence of BN that modify the distribution of each batch, centering the hidden representation in 0. The dataset is then given simply by the union of batches. Being the batches all centred in 0, the overall dataset average will be 0 as well. This means that, again according to the analysis of Ref. Francazi et al. (2024) (App. D) we have

$$f_{\mathrm{O}_c}^{(\chi)}(\mathrm{o}) = \mathcal{N}\left(\mathrm{o}; 0, \sigma_w^2 \mathrm{Var}_\chi\left(\mathrm{B}_i^{(L)}\right)\right) , \tag{81}$$

and therefore absence of IGB.

F Layer Normalization

LN is commonly applied in recurrent neural networks and Transformer models, typically positioned immediately before activation functions Ba et al. (2016). Unlike BN, LN computes statistics over individual samples, normalizing across all features within a single data point. Below, we analyze how the placement of LN within hidden layers impacts IGB in deep architectures.
Formally, LN computes normalization across the nodes within a layer, not across batches. Thus, given a layer with outputs from numerous nodes, LN standardizes these outputs individually for each data point. With the assumption that the number of nodes N_l in each layer l is sufficiently large, $N_l \gg 1$, we leverage the CLT in subsequent analyses.

In Fig. 3, we illustrate the critical influence of LN placement. Specifically, applying **ReLU after LN** (LN + ReLU) maintains IGB, whereas applying **LN after ReLU** (ReLU + LN) eliminates IGB entirely.

To quantify fluctuations of layer outputs, we introduce $\gamma(\mathcal{A}, \psi(\chi))$ (Definition C.2):

$$\gamma(\mathcal{A}, \psi(\chi)) \equiv \frac{\mathrm{Var}_{\mathcal{W}}\left(\mu_c\right)}{\mathrm{Var}_{\chi}\left(\mathrm{O}_c\right)}. \tag{82}$$

An increased $\gamma(\mathcal{A}, \psi(\chi))$ signifies greater IGB. Our analysis demonstrates that placing LN before ReLU proportionally scales both variances, preserving IGB, whereas positioning LN after ReLU sets $\gamma(\mathcal{A}, \psi(\chi))$ consistently to zero.

To support subsequent analysis, we introduce the following lemma, elucidating the linearity properties of rectified normal distributions under scaling:

Lemma F.1 (Linearity of Mean and Variance of Rectified Normal Distributions)**.** *Let $X \sim \mathcal{N}(\mu, \sigma^2)$ and define $Y = \alpha X$, with $\alpha \in \mathbb{R}$. The ReLU activation function, defined as $ReLU(z) = \max(0, z)$, satisfies:*

$$\mathbb{E}[ReLU(Y)] = \alpha\, \mathbb{E}[ReLU(X)], \tag{83}$$

$$Var[ReLU(Y)] = \alpha^2\, Var[ReLU(X)]. \tag{84}$$

Proof. Let $X \sim \mathcal{N}(\mu, \sigma^2)$ and $Y = \alpha X \sim \mathcal{N}(\alpha\mu, \alpha^2\sigma^2)$. Consider the cumulative distribution function (CDF) Φ of the standard normal distribution, defined by:

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-t^2/2}\, dt.$$

The distribution of the rectified variable $\mathrm{ReLU}(X)$ is given by:

$$\mathrm{ReLU}(X) \sim \Phi\left(-\frac{\mu}{\sigma}\right)\delta(x) + \Theta(x)\mathcal{N}(x; \mu, \sigma^2),$$

where $\delta(x)$ is the Dirac delta and $\Theta(x)$ the Heaviside step function. Its expectation and variance are:

$$\mathbb{E}[\mathrm{ReLU}(X)] = \mu\left[1 - \Phi\left(-\frac{\mu}{\sigma}\right)\right] + \frac{\sigma}{\sqrt{2\pi}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right), \tag{85}$$

$$\mathrm{Var}[\mathrm{ReLU}(X)] = (\mu^2 + \sigma^2)\left[1 - \Phi\left(-\frac{\mu}{\sigma}\right)\right] + \frac{\mu\sigma}{\sqrt{2\pi}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right) - \mathbb{E}[\mathrm{ReLU}(X)]^2. \tag{86}$$

Similarly, the rectified distribution $\mathrm{ReLU}(Y)$ is:

$$\mathrm{ReLU}(Y) \sim \Phi\left(-\frac{\alpha\mu}{|\alpha|\sigma}\right)\delta(y) + \Theta(y)\mathcal{N}(y; \alpha\mu, \alpha^2\sigma^2).$$

Evaluating expectation and variance, we have:

$$\mathbb{E}[\mathrm{ReLU}(Y)] = |\alpha| \left[\frac{\alpha}{|\alpha|}\mu \left(1 - \Phi\left(-\frac{\alpha\mu}{|\alpha|\sigma}\right)\right) + \frac{|\alpha|\sigma}{\sqrt{2\pi}} \exp\left(-\frac{\alpha^2\mu^2}{2\alpha^2\sigma^2}\right)\right] \tag{87}$$

$$= \alpha \left[\mu \left(1 - \Phi\left(-\frac{\mu}{\sigma}\right)\right) + \frac{\sigma}{\sqrt{2\pi}} \exp\left(-\frac{\mu^2}{2\sigma^2}\right)\right] \tag{88}$$

$$= \alpha\, \mathbb{E}[\mathrm{ReLU}(X)], \tag{89}$$

$$\mathrm{Var}[\mathrm{ReLU}(Y)] = \alpha^2 \mathrm{Var}[\mathrm{ReLU}(X)], \tag{90}$$

thus confirming linearity and completing the proof. □

F.1 LN + ReLU

We now analyze the impact on IGB of applying LN immediately before the ReLU activation function within settings depicted in Fig. A.1. A fundamental assumption throughout our analysis is that the width of each hidden layer is infinitely large, i.e., $N_l \to \infty$ for every layer l, with N_l indicating the number of nodes in layer l. Additionally, consistent with typical IGB analysis (Francazi et al., 2023), we assume the initial input data distribution is standardized normal:

$$\mathrm{X}_i^{(0)} \sim \mathcal{N}(0, 1). \tag{91}$$

The following is a formal formulation of Th. 5.3.

Theorem F.2 (LN + ReLU)**.** *Consider the same setting as in Thm. C.1, but with LN applied before the activation function in each hidden layer. In the limit of infinite width and dataset size, the distribution of the output node* O_c *at initialization converges to:*

$$f_{\mathrm{O}_c}^{(\chi)}(\mathrm{o}) \xrightarrow{|\mathcal{W}|\to\infty} \mathcal{N}\left(\mathrm{o}; \mu_c, \left(\hat{\sigma}^{(1)}\right)^2 Var_\chi(\mathrm{O}_c)\right), \tag{92}$$

$$f_{\mu_c}(m) \xrightarrow{|\mathcal{W}|\to\infty} \mathcal{N}\left(m; 0, \left(\hat{\sigma}^{(1)}\right)^2 Var_{\mathcal{W}}(\mu_c)\right), \tag{93}$$

where $Var_\chi(\mathrm{O}_c)$ *and* $Var_{\mathcal{W}}(\mu_c)$ *denote variances of the output distribution and output means in the baseline scenario (ReLU-only, without normalization). Here,* $\left(\hat{\sigma}^{(1)}\right)^2$ *represents the variance of pre-activation values across the first layer prior to normalization.*

This theorem shows explicitly how Layer Normalization, when placed immediately before the ReLU activation, affects the distribution of layer outputs. Crucially, the distribution retains the same functional form as the baseline scenario described in Thm. C.1, with the only difference being a scaling of both variances by the same factor $\left(\hat{\sigma}^{(1)}\right)^2$. This scaling is constant and does not vary with network depth or across initialization instances. Consequently, when we compute $\gamma(\mathcal{A}, \psi(\chi))$ (see Eq. (82)), which measures IGB as the ratio between variance of centers and variance of distributions around those centers, the scaling factor cancels out. Thus, the level of IGB remains exactly the same as in the unnormalized baseline scenario.

To formally demonstrate this equivalence, we now derive the distribution of the intermediate node $\mathrm{Z}_i^{(l)}$, representing the output of a deep neural network after l layers (Eq. 3). By comparing the resultant distributions and variances with and without LN, we verify explicitly the invariance of the IGB measure $\gamma(\mathcal{A}, \psi(\chi))$.

Input → Dense

After passing through the Dense Layer, the input is mapped following the same linear transformation as in Eq. (65) Now, averaging over the data distribution gives

$$\mathbb{E}_\chi\left(\mathrm{Z}_i^{(1)}\right) = \mathbb{E}_\chi\left(\sum_{i=1}^{D} w_{ij}^{(0)} \mathrm{X}_j^{(0)}\right) = \sum_{i=1}^{D} w_{ij}^{(0)} \mathbb{E}_\chi\left(\mathrm{X}_j^{(0)}\right) = 0, \tag{94}$$

while the variance over the data is

$$\mathrm{Var}_\chi\left(\mathrm{Z}_i^{(1)}\right) = \mathrm{Var}_\chi\left(\sum_{i=1}^{D} w_{ij}^{(0)}\mathrm{X}_j^{(0)}\right) = \sum_{i=1}^{D}\left(w_{ij}^{(0)}\right)^2 \mathrm{Var}\left(\mathrm{X}_j^{(0)}\right) = D\frac{\sigma_w^2}{D} = \sigma_w^2\,, \tag{95}$$

Where we used the law of large numbers,

$$\frac{1}{d}\sum_{j=1}^{d}\left(w_{ij}^{(0)}\right)^2 \overset{d\to\infty}{\longrightarrow} \mathbb{E}_{\mathcal{W}}\left(\left(w_{ij}^{(0)}\right)^2\right) = \frac{\sigma_w^2}{d}\,.$$

After passing through the weights, the Dense output will distribute as

$$f^{(\chi)}_{\mathrm{Z}_i^{(1)}}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z}; 0, \sigma_w^2\right). \tag{96}$$

Dense → LN
The second step is to normalize the Dense output along the layer.
The transformation performed in LN standardizes $\mathrm{Z}_{i;a}^{(1)}$ over all the elements of the respective layer as follows:

$$\mathrm{L}_{i;a}^{(1)} = \frac{\mathrm{Z}_{i;a}^{(1)} - \hat{\mu}_a^{(1)}}{\hat{\sigma}_a^{(1)}}, \tag{97}$$

where

- $\mathrm{L}_{i;a}^{(1)}$ represents the ith output (node) of the input $\mathbf{X}_a^{(0)}$ after passing through LN in the first block
- $\hat{\mu}_a^{(1)} = \sum_i^{N_1} \mathrm{Z}_{i;a}^{(1)}/N_1$ is the average along the layer. $\hat{\mu}_a^{(1)} \xrightarrow{N_1\to\infty} \mathbb{E}_{\mathcal{W}}\left(\mathrm{Z}_{i;a}^{(1)}\right)$.
- $\left(\hat{\sigma}_a^{(1)}\right)^2 = \sum_j^{N_1}\left(\mathrm{Z}_j^{(1)} - \hat{\mu}_a^{(1)}\right)^2/N_1$ is the variance along the layer.

Note the dependence on the sample index a in $\hat{\mu}_a^{(1)}$ and $\hat{\sigma}_a^{(1)}$ as the two quantity are estimated over the sample node components. In the limit of $N_1 \to \infty$, however, these two random quantities concentrate around their mean value, according to the CLT, *i.e.*:

$$\hat{\mu}_a^{(1)} \to 0\,, \tag{98}$$

$$\hat{\sigma}_a^{(1)} \to \hat{\sigma}^{(1)}\,, \tag{99}$$

where $\left(\hat{\sigma}^{(1)}\right)^2$ indicate the variance over the whole layer l, not anymore of a single node. We will assume this wide network regime in the following, avoiding the sample dependece on the estimators. Then we can re-write the transformation as

$$\mathrm{L}_i^{(1)} = \frac{\mathrm{Z}_i^{(1)}}{\hat{\sigma}^{(1)}}. \tag{100}$$

The distribution can now be written as

$$f^{(\chi)}_{\mathrm{L}_i^{(1)}}(\mathrm{l}) = \mathcal{N}\left(\mathrm{l}; 0, \frac{\sigma_w^2}{\left(\hat{\sigma}^{(1)}\right)^2}\right). \tag{101}$$

Note that we are applying just a scaling, and not a shifting to the normal distribution. This means that the mean does not vary, but the variance gets scaled.

LN → ReLU
We then observe what happens after passing the LN output through the ReLU: we refer to the ith output of the

ReLU activation of the 1st layer as $\mathrm{Y}_i^{(1)}$.
The output of the ReLU will then follow the distribution below:

$$f^{(\chi)}_{\mathrm{Y}_i^{(1)}}(\mathrm{y}) = \frac{1}{2}\delta(\mathrm{y}) + \Theta(\mathrm{y})\mathcal{N}\left(\mathrm{y};\, 0,\, \frac{\sigma_w^2}{\left(\hat{\sigma}^{(1)}\right)^2}\right). \tag{102}$$

We will show in the following that this setting generates the same level of IGB as in the absence of LN, *i.e.*, we get the same value for $\gamma(\mathcal{A}, \psi(\chi))$ at each depth. Note that, in the absence of LN, we have an MLP with a ReLU activation function. As shown in Francazi et al. (2024), in this setting we have amplifying IGB, *i.e.* IGB emerges and gets amplified with the depth. To show the equivalence, in terms of IGB, between the two cases we will compare the two settings. We will call from now on with $\overline{\mathrm{Y}}_i^{(1)}$ the node after passing through the ReLU in a MLP without LN. Applying the ReLU to the distribution in Eq. (96) we will have

$$f^{(\chi)}_{\overline{\mathrm{Y}}_i^{(1)}}(\mathrm{y}) = \frac{1}{2}\delta(\mathrm{y}) + \Theta(\mathrm{y})\mathcal{N}\left(\mathrm{y}; 0, \sigma_w^2\right). \tag{103}$$

As demonstrated in Theorem F.1, the expected value and the variance of the two distributions are

$$\mathbb{E}_\chi\left(\mathrm{Y}_i^{(1)}\right) = \frac{\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_i^{(1)}\right)}{\hat{\sigma}^{(1)}}, \tag{104}$$

$$\mathrm{Var}_\chi\left(\mathrm{Y}_i^{(1)}\right) = \frac{\mathrm{Var}_\chi\left(\overline{\mathrm{Y}}_i^{(1)}\right)}{\left(\hat{\sigma}^{(1)}\right)^2}. \tag{105}$$

The latter will be useful in the next steps.

ReLU → Dense
The transformation after the Dense Layer is

$$\mathrm{Z}_i^{(2)} = \sum_{j=1}^{N_1} w_{ij}^{(1)} \mathrm{Y}_j^{(1)}. \tag{106}$$

Then the distribution is

$$f^{(\chi)}_{\mathrm{Z}_i^{(2)}}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z}; \sum_j^{N_1} w_{ij}^{(1)} \mathbb{E}_\chi\left(\mathrm{Y}_j^{(1)}\right), \sum_j^{N_1} \left(w_{ij}^{(1)}\right)^2 \mathrm{Var}_\chi\left(\mathrm{Y}_j^{(1)}\right)\right) \tag{107}$$

We just saw what happens to the distribution of the output after the first set of layers (Dense → LN → ReLU → Dense).
The next step is to study the behavior of the variance of the centers and the behavior of the variance around the centers: if the ratio of the latter increases with the number of layers, it means that there is strong IGB.

First of all, we evaluate the probability distribution of the centers after the first layer.

$$f_{\mathbb{E}_\chi\left(\mathrm{Z}_i^{(2)}\right)}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z}; \sum_j^{N_1} \mathbb{E}_\mathcal{W}\left(w_{ij}^{(1)}\right) \mathbb{E}_\chi\left(\mathrm{Y}_j^{(1)}\right), \sum_j^{N_1} \mathrm{Var}_\mathcal{W}\left(w_{ij}^{(1)}\right) \mathbb{E}_\chi\left(\mathrm{Y}_j^{(1)}\right)^2\right), \tag{108}$$

Where $\mathbb{E}_{\mathcal{W}}(\cdot)$ is the expected value of the weights randomness.
The expected value of the weights is zero, so the expected value of $\mathbb{E}_\chi\left(\mathrm{Z}_i^{(2)}\right)$ becomes zero, since the weights are centered in zero:

$$\mathbb{E}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\mathrm{Z}_i^{(2)}\right)\right) = \sum_j^{N_1} \mathbb{E}_{\mathcal{W}}\left(w_{ij}^{(1)}\right)\mathbb{E}_\chi\left(\mathrm{Y}_j^{(1)}\right) = 0, \tag{109}$$

The variance is instead

$$\begin{aligned}\mathrm{Var}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\mathrm{Z}_i^{(2)}\right)\right) &= \sum_j^{N_1} \mathrm{Var}_{\mathcal{W}}\left(w_{ij}^{(1)}\right)\mathbb{E}_\chi\left(\mathrm{Y}_j^{(1)}\right)^2 = \sigma_w^2 \frac{1}{N_1}\sum_j^{N_1}\mathbb{E}_\chi\left(\mathrm{Y}_j^{(1)}\right)^2 \\ &= \sigma_w^2 \frac{1}{N_1}\sum_j^{N_1}\frac{\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)^2}{\left(\hat{\sigma}^{(1)}\right)^2} \xrightarrow{N_1\to\infty} \sigma_w^2 \frac{\mathbb{E}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)^2\right)}{\left(\hat{\sigma}^{(1)}\right)^2},\end{aligned} \tag{110}$$

where $\mathbb{E}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)^2\right)$ is the average of $\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)^2$ over the weights randomness.
Substituting in Eq. (108) we have:

$$f_{\mathbb{E}_\chi\left(\mathrm{Z}_i^{(2)}\right)}(\mathrm{z}) \xrightarrow{N_1\to\infty} \mathcal{N}\left(\mathrm{z};\, 0, \sigma_w^2 \frac{\mathbb{E}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)^2\right)}{\left(\hat{\sigma}^{(1)}\right)^2}\right). \tag{111}$$

We now pass to the denominator of the r.h.s. of Eq. (82). We saw in Eq. (107) that the output of the dense layer $\mathrm{Z}_i^{(2)}$ distributes as

$$f^{(\chi)}_{\mathrm{Z}_i^{(2)}}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z}; \sum_j^{N_1} w_{ij}^{(1)}\mathbb{E}_\chi\left(\mathrm{Y}_j^{(1)}\right), \sum_j^{N_1}\left(w_{ij}^{(1)}\right)^2 \mathrm{Var}_\chi\left(\mathrm{Y}_j^{(1)}\right)\right). \tag{112}$$

Now we need to study the expected value of the random variable $\mathrm{Var}_\chi\left(\mathrm{Z}_i^{(2)}\right)$ to compare it with the ones of the centers from Eq. (111):

$$\begin{aligned}\mathbb{E}_{\mathcal{W}}\left(\mathrm{Var}_\chi\left(\mathrm{Z}_i^{(2)}\right)\right) =& \mathbb{E}_{\mathcal{W}}\left(\sum_j^{N_1}\left(w_{ij}^{(1)}\right)^2 \mathrm{Var}_\chi\left(\mathrm{Y}_j^{(1)}\right)\right) = \sum_j^{N_1}\mathbb{E}_{\mathcal{W}}\left(\left(w_{ij}^{(1)}\right)^2\right)\mathbb{E}_{\mathcal{W}}\left(\mathrm{Var}_\chi\left(\mathrm{Y}_j^{(1)}\right)\right) \\ =& \sigma_w^2\frac{1}{N_1}\sum_j^{N_1}\mathbb{E}_{\mathcal{W}}\left(\mathrm{Var}_\chi\left(\mathrm{Y}_j^{(1)}\right)\right) = \sigma_w^2\frac{1}{N_1}\sum_j^{N_1}\frac{\mathbb{E}_{\mathcal{W}}\left(\mathrm{Var}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)\right)}{\left(\hat{\sigma}^{(l)}\right)^2} \\ =& \sigma_w^2\frac{\mathbb{E}_{\mathcal{W}}\left(\mathrm{Var}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)\right)}{\left(\hat{\sigma}^{(l)}\right)^2}.\end{aligned} \tag{113}$$

Finally, from Eq. (110) and Eq. (113) we get

$$\frac{\mathrm{Var}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\mathrm{Z}_0^{(2)}\right)\right)}{\mathrm{Var}_\chi\left(\mathrm{Z}_0^{(2)}\right)} = \frac{\sigma_w^2\frac{\mathbb{E}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)^2\right)}{\left(\hat{\sigma}^{(l)}\right)^2}}{\sigma_w^2\frac{\mathbb{E}_{\mathcal{W}}\left(\mathrm{Var}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)\right)}{\left(\hat{\sigma}^{(l)}\right)^2}} = \frac{\mathbb{E}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)^2\right)}{\mathbb{E}_{\mathcal{W}}\left(\mathrm{Var}_\chi\left(\overline{\mathrm{Y}}_j^{(1)}\right)\right)}. \tag{114}$$

Note that $\mathrm{Z}_c^{(L)} = \mathrm{O}_c$ for a DNN with L hidden layers (see Eq. (3)). The ratio in the last term of Eq. (114) is formally equal to the one we would get in a ReLU MLP, as shown in Ref. Francazi et al. (2024).

We now proceed with the same computation for the following layers.

Dense → LN

In order to evaluate the expected value and the variance over the output layers, we need to study the probability distribution over the entire layer, not just over the single node's output. To do so, we can compute the distribution of the layer preactivation as a convolution over the generic single node distribution. To distinguish the random variable associated with the specific node i, *i.e.*, $\mathrm{Z}_i^{(2)}$, from the generalized layer pre-activation (whose statistics are computed over different layer nodes), we indicate the latter with $\mathrm{Z}^{(2)}$. Note that there is a crucial difference between $\mathrm{Z}_i^{(2)}$ and $\mathrm{Z}^{(2)}$ as, in the presence of IGB, nodes of the same layer are not identically distributed. In particular, since the difference between the nodes' distributions lies in the mean, we can derive the distribution of the layer by expressing the node distribution as a conditional probability (conditioned on the mean value) and marginalizing over the mean distribution, *i.e.*

$$f_{\mathrm{Z}^{(2)}}(\mathrm{z}) = \int_{\mathbb{R}} f_{\mathrm{Z}_i^{(2)}}\left(\mathrm{z} \mid \mathbb{E}_\chi\left(\mathrm{Z}_i^{(2)}\right) = y\right) f_{\mathbb{E}_\chi\left(\mathrm{Z}_i^{(2)}\right)}(y)\, dy\,. \tag{115}$$

The left-hand side of Eq. (115) is the marginal distribution of the second-layer preactivations (not conditioned on a specific node). Conceptually, nodes in layer 2 are independent but not identically distributed: each node has a center that depends on the fixed weights. The integral builds the layerwise distribution as a mixture over these node distributions, so we are effectively marginalizing over the weight initialization through the distribution of (note that is an expectation over dataset randomness but still a random variable with respect to the weights).

In the following we refer to the variances of the two distributions respectively as $\mathrm{Var}_\chi\left(\mathrm{Z}_i^{(2)}\right) = \sigma_{\mathrm{z}}^2$ and $\mathrm{Var}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\mathrm{Z}_i^{(2)}\right)\right) = \sigma^2_{\mathbb{E}_\chi(\mathrm{z})}$. We will have then:

$$\begin{aligned} f_{\mathrm{Z}^{(2)}}(\mathrm{z}) &= \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma_{\mathrm{z}}^2}} \exp\left(-\frac{1}{2}\frac{(x-y)^2}{\sigma_{\mathrm{z}}^2}\right) \cdot \frac{1}{\sqrt{2\pi\sigma^2_{\mathbb{E}_\chi(\mathrm{z})}}} \exp\left(-\frac{1}{2}\frac{(y)^2}{\sigma^2_{\mathbb{E}_\chi(\mathrm{z})}}\right) dy \\ &= \int_{\mathbb{R}} \frac{1}{2\pi\sigma_{\mathrm{z}}\sigma_{\mathbb{E}_\chi(\mathrm{z})}} \exp\left(-\frac{1}{2}\left(\frac{(x-y)^2}{\sigma_{\mathrm{z}}^2} + \frac{y^2}{\sigma^2_{\mathbb{E}_\chi(\mathrm{z})}}\right)\right) dy \\ &= \frac{1}{\sqrt{2\pi(\sigma_{\mathrm{z}}^2 + \sigma^2_{\mathbb{E}_\chi(\mathrm{z})})}} \exp\left(-\frac{x^2}{2(\sigma_{\mathrm{z}}^2 + \sigma^2_{\mathbb{E}_\chi(\mathrm{z})})}\right) = \mathcal{N}\left(x; 0, \sigma_{\mathrm{z}}^2 + \sigma^2_{\mathbb{E}_\chi(\mathrm{z})}\right) \end{aligned} \tag{116}$$

We now need to normalize our output over the Layer's expected value and standard deviation.
We already know the distribution of the layer's output, so in our case the normalization will have the form:

$$\mathrm{L}_i^{(2)} = \frac{\mathrm{Z}_i^{(2)} - 0}{\sqrt{\sigma_{\mathrm{z}}^2 + \sigma^2_{\mathbb{E}_\chi(\mathrm{z})}}} = \frac{\mathrm{Z}_i^{(2)}}{\sqrt{\sigma_{\mathrm{z}}^2 + \sigma^2_{\mathbb{E}_\chi(\mathrm{z})}}} \sim \mathcal{N}\left(\frac{\sum_j^{N_1} w_{ij}^{(1)} \mathbb{E}_\chi\left(\mathrm{Y}_j^{(1)}\right)}{\sqrt{\sigma_{\mathrm{z}}^2 + \sigma^2_{\mathbb{E}_\chi(\mathrm{z})}}}, \frac{\sum_j^{N_1}\left(w_{ij}^{(1)}\right)^2 \mathrm{Var}_\chi\left(\mathrm{Y}_j^{(1)}\right)}{\sigma_{\mathrm{z}}^2 + \sigma^2_{\mathbb{E}_\chi(\mathrm{z})}}\right) \tag{117}$$

LN → ReLU
Now we will demonstrate that, even if the normally-distributed input is not centered in zero, the effect of LN is vain on IGB.
Also in this case, we will compare the parameters of the ReLU when LN is not applied with our actual parameters.

In order to simplify the notation, let us substitute the terms as follows:

- $\sum_j^{N_1} w_{ij}^{(1)} \mathbb{E}_\chi\left(\mathrm{Y}_j^{(1)}\right) = \mu$ is the numerator of the mean of $f^{(\chi)}_{\mathrm{L}_i^{(2)}}(l)$ in Eq. (117)
- $\sum_j^{N_1}\left(w_{ij}^{(1)}\right)^2 \mathrm{Var}_\chi\left(\mathrm{Y}_j^{(1)}\right) = \sigma^2$ is the numerator of the variance of $f^{(\chi)}_{\mathrm{L}_i^{(2)}}(l)$ in Eq. (117)

- $\sqrt{\sigma_{\mathrm{z}}^2+\sigma_{\langle \mathrm{z}\rangle}^2}=\alpha$ is the denominator of the mean (and the standard deviation's denominator) of $f^{(\chi)}_{\mathrm{L}_i^{(2)}}(l)$ in Eq. (117)

We can now re-write the distribution of $\mathrm{L}_i^{(2)}|\mathcal{W}$ as follows:

$$f^{(\chi)}_{\mathrm{L}_i^{(2)}}(l)=\mathcal{N}\left(l;\,\frac{\mu}{\alpha},\,\frac{\sigma^2}{\alpha^2}\right). \tag{118}$$

while - if the LN is not applied - the distribution would remain

$$f^{(\chi)}_{\mathrm{Z}_i^{(2)}}(\mathrm{z})=\mathcal{N}\left(\mathrm{z};\,\mu,\,\sigma^2\right). \tag{119}$$

The output of the rectified $\overline{\mathrm{Y}}_i^{(2)}$ would be

$$f^{(\chi)}_{\overline{\mathrm{Y}}_i^{(2)}}(\mathrm{y})=\Phi\left(\frac{\mu}{\sigma}\right)\delta(\mathrm{y})+\Theta(\mathrm{y})\mathcal{N}\left(\mathrm{y};\,\mu,\,\sigma^2\right), \tag{120}$$

while the distribution of the rectified $\mathrm{L}_i^{(2)}$ will be

$$f^{(\chi)}_{\mathrm{Y}_i^{(2)}}(\mathrm{y})=\Phi\left(\frac{\mu/\alpha}{\sigma/\alpha}\right)\delta(\mathrm{y})+\Theta(\mathrm{y})\mathcal{N}\left(\mathrm{y};\,\frac{\mu}{\alpha},\,\frac{\sigma^2}{\alpha^2}\right). \tag{121}$$

From Theorem F.1 about linearity of expected value and variance of rectified normal distributions, we can see that - if LN is applied - the mean and the variance of $\mathrm{Y}_i^{(2)}$ will be proportional to the ones of $\overline{\mathrm{Y}}_i^{(2)}$:

$$\mathbb{E}_\chi\left(\mathrm{Y}_i^{(2)}\right)=\frac{1}{\sqrt{\sigma_{\mathrm{z}}^2+\sigma_{\langle \mathrm{z}\rangle}^2}}\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_i^{(2)}\right), \tag{122}$$

$$\mathrm{Var}_\chi\left(\mathrm{Y}_i^{(2)}\right)=\frac{1}{\sigma_{\mathrm{z}}^2+\sigma_{\langle \mathrm{z}\rangle}^2}\mathrm{Var}_\chi\left(\overline{\mathrm{Y}}_i^{(2)}\right). \tag{123}$$

This means that the procedure to find the ratio of the variance of the centers over the variance around the centers is exactly the same as the one starting from Eq. 107.
After the next Dense layer, the ratio of the two variances of interest will be

$$\frac{\mathrm{Var}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\mathrm{Z}_0^{(3)}\right)\right)}{\mathrm{Var}_\chi\left(\mathrm{Z}_0^{(3)}\right)}=\frac{\sigma_w^2\frac{\mathbb{E}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_0^{(2)}\right)^2\right)}{(\sigma_z^2+\sigma_{\langle z\rangle}^2)}}{\sigma_w^2\frac{\mathbb{E}_{\mathcal{W}}\left(\mathrm{Var}_\chi\left(\overline{\mathrm{Y}}_0^{(2)}\right)\right)}{(\sigma_z^2+\sigma_{\langle z\rangle}^2)}}=\frac{\mathbb{E}_{\mathcal{W}}\left(\mathbb{E}_\chi\left(\overline{\mathrm{Y}}_0^{(2)}\right)^2\right)}{\mathbb{E}_{\mathcal{W}}\left(\mathrm{Var}_\chi\left(\overline{\mathrm{Y}}_0^{(2)}\right)\right)}, \tag{124}$$

which again is formally equal to the expression of $\gamma(\mathcal{A},\psi(\chi))$ in absence of LN, computed in Francazi et al. (2024).
We can proceed in the same way for the generic layer l, arriving to the same conclusion.
We have now demonstrated that, the presence of LN before ReLU leave the level of IGB unchanged. Therefore, as in the ReLU MLP settings, we will have a level of IGB increasing with the depth of the DNN.
The same procedure can be iterated for all the following layers.

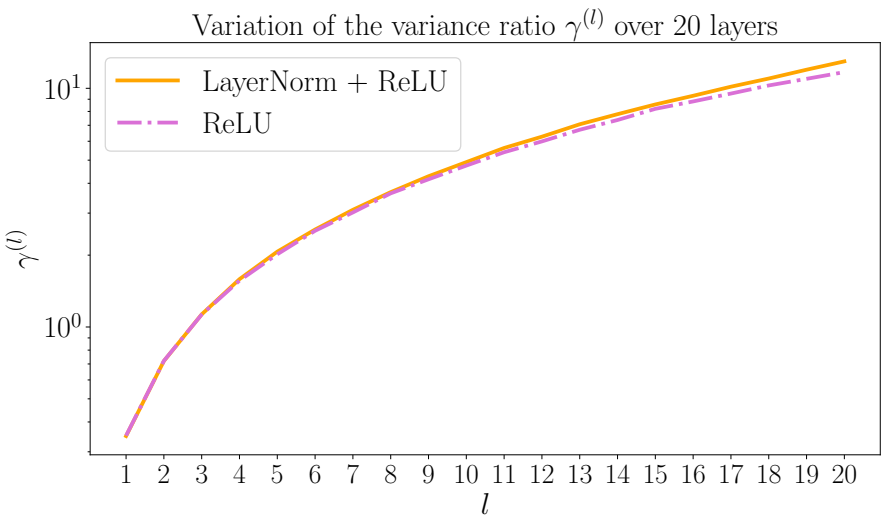

Figure F.1: Empirical comparison of the variance ratio $\frac{\mathrm{Var}_{\mathcal{W}}\left(\mathbb{E}_{\chi}\left(\mathrm{Z}_0^{(l)}\right)\right)}{\mathrm{Var}_{\chi}\left(\mathrm{Z}_0^{(l)}\right)}$ between the ReLU case and the LN+ReLU case for 20 consecutive layers. The two curves are overlapped, meaning that the effect of LN before ReLU is vain on IGB.

F.2 ReLU + LN

We now study what happens when LN is applied after ReLU, with a neural network within the setting illustrated in Fig. A.1. We again assume that our input data distributes as $\mathcal{N}(0,1)$, as the typical setting employed in IGB analysis (Francazi et al., 2023).
The input data will then pass through the first dense layer: the weights are initialized with **Kaiming Initialization**. An important assumption that will be valid for all the passages, is that $N_l \to \infty$ for every l.

The following is a formal formulation of Th. 5.4.

Theorem F.3 (ReLU + LN)**.** *Consider the setting described in Sec. B with an MLP where LN is applied* after *the ReLU activation function in each hidden layer. Then, in the limit of infinite width and dataset size, the output distribution* $f_{\mathrm{O}_c}^{(\chi)}(\mathrm{o})$ *converges to:*

$$f_{\mathrm{O}_c}^{(\chi)}(\mathrm{o}) \xrightarrow{|\mathcal{W}|\to\infty} \mathcal{N}\left(\mathrm{o}; 0, \sigma_w^2\right), \tag{125}$$

with the corresponding mean distribution collapsing to:

$$f_{\mu_c}(m) \xrightarrow{|\mathcal{W}|\to\infty} \delta(m), \tag{126}$$

indicating the suppression of Initial Guessing Bias and the emergence of fully balanced predictions at initialization.

Input → Dense
The first step is to observe the distribution of the output after passing through the Dense Layer.
The transformation performed is the following:

$$\mathrm{Z}_i^{(1)} = \sum_{j=1}^{N_0} w_{ij}^{(0)} \mathrm{X}_j^{(0)}. \tag{127}$$

where $\mathrm{Z}_i^{(1)}$ represents the ith output (node) of the Dense Layer in the first block.

Now, the mean over the data of $\mathrm{Z}_i^{(1)}$ is

$$\mathbb{E}_\chi\left(\mathrm{Z}_i^{(1)}\right) = \mathbb{E}_\chi\left(\sum_{i=1}^{N_0} w_{ij}^{(0)}\mathrm{X}_j^{(0)}\right) = \sum_{i=1}^{N_0} w_{ij}^{(0)}\mathbb{E}_\chi\left(\mathrm{X}_j^{(0)}\right) = 0, \tag{128}$$

While the variance over the data is

$$\mathrm{Var}\left(\mathrm{Z}_i^{(1)}\right) = \mathrm{Var}\left(\sum_{i=1}^{N_1} w_{ij}^{(0)}\mathrm{X}_j^{(0)}\right) \overset{\text{Ind.}}{=} \sum_{i=1}^{N_1}\left(w_{ij}^{(0)} - 0\right)^2 \mathrm{Var}\left(\mathrm{X}_j^{(0)}\right) = N_1\frac{\sigma_w^2}{N_1} = \sigma_w^2. \tag{129}$$

After passing through the weights, the Dense output of every node will distribute, independently, as

$$f_{\mathrm{Z}_i^{(1)}}^{(\chi)}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z};\, 0, \sigma_{w_i}^2\right). \tag{130}$$

Dense → ReLU
After passing every node's output through the ReLU activation function, we will have the same distribution for every node, defined as

$$f_{\mathrm{Y}_i^{(1)}}^{(\chi)}(\mathrm{y}) = \frac{1}{2}\delta(\mathrm{y}) + \Theta(\mathrm{y})\mathcal{N}\left(\mathrm{y}; 0,\, \sigma_w^2\right), \tag{131}$$

Given that all the $\mathrm{Y}_i^{(1)}$ are identically distributed, it means that

$$\mathbb{E}_\chi\left(\mathrm{Y}_i^{(1)}\right) = \mathbb{E}_\chi\left(\mathrm{Y}_j^{(1)}\right)\ \forall\, i, j \in \mathbb{N}, \tag{132}$$

$$\mathrm{Var}_\chi\left(\mathrm{Y}_i^{(1)}\right) = \mathrm{Var}_\chi\left(\mathrm{Y}_j^{(1)}\right)\ \forall\, i, j \in \mathbb{N}. \tag{133}$$

ReLU → LN
We now normalize the ReLU output over the layer. The elements needed for LN have already been defined as:

- $\mathrm{L}_{i;a}^{(1)}$ represents the ith output (node) of the LN in the first block
- $\hat{\mu}_a^{(1)} = \sum_i^{N_1} \mathrm{Y}_i^{(1)}/N_1$ is the average along the layer. $\hat{\mu}_a^{(1)} \xrightarrow{N_1\to\infty} \mathbb{E}_{\mathcal{W}}\left(\mathrm{Y}_i^{(1)}\right)$
- $\left(\hat{\sigma}_a^{(1)}\right)^2 = \sum_i^{N_1}\left(\mathrm{Y}_i^{(1)} - \hat{\mu}_a^{(1)}\right)/N_1$ is the variance along the layer.

As in Sec. F.1 we work in the limit of $N_l \to \infty\ \forall l$ where the two estimators $\hat{\mu}_a^{(1)}$ and $\left(\hat{\sigma}_a^{(1)}\right)^2$ converge, according to the CLT, to their expectation value. Note that in the first layer the nodes $\left\{\mathrm{Y}_i^{(1)}\right\}$ are independent and identically distributed (Francazi et al., 2024). Therefore, we can see the LN transformation as equivalent to a BN transformation:

$$\mathrm{L}_{i;a}^{(1)} = \frac{\mathrm{Y}_i^{(1)} - \mathbb{E}_{\mathcal{W}}\left(\mathrm{Y}_i^{(1)}\right)}{\sqrt{\mathrm{Var}_{\mathcal{W}}\left(\mathrm{Y}_i^{(1)}\right)}} = \frac{\mathrm{Y}_i^{(1)} - \mathbb{E}_\chi\left(\mathrm{Y}_i^{(1)}\right)}{\sqrt{\mathrm{Var}_\chi\left(\mathrm{Y}_i^{(1)}\right)}}, \tag{134}$$

This means that the distribution of every node of $l^{(1)}$ has zero mean and unitary variance; the expected value and the variance over the dataset are standardized:

$$\mathbb{E}_\chi\left(\mathrm{L}_i^{(1)}\right) = 0,\ \forall i \tag{135}$$

$$\mathrm{Var}_\chi\left(\mathrm{L}_i^{(1)}\right) = 1,\ \forall i. \tag{136}$$

LN → Dense
The transformation performed by the fully connected layer is the following:

$$\mathrm{Z}_i^{(2)} = \sum_{j=1}^{N_1} w_{ij}^{(1)}\mathrm{L}_j^{(1)}. \tag{137}$$

So the distribution of $\mathrm{Z}_i^{(2)}$ will be

$$f^{(\chi)}_{\mathrm{Z}_i^{(1)}}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z};\, \sum_j^{N_1} w_j^{(1)} \mathbb{E}_\chi\left(\mathrm{L}_j^{(1)}\right), \sum_j^{N_1} \left(w_j^{(1)}\right)^2 \mathrm{Var}_\chi\left(\mathrm{L}_j^{(1)}\right)\right) \tag{138}$$

Given that we demonstrated that $\mathbb{E}_\chi\left(\mathrm{L}_i^{(1)}\right) = 0$, $\forall i$ and $\mathrm{Var}_\chi\left(\mathrm{L}_i^{(1)}\right) = 1$, $\forall i$ the distribution will be

$$f^{(\chi)}_{\mathrm{Z}_i^{(1)}}(\mathrm{z}) = \mathcal{N}\left(\mathrm{z};\, 0, \sum_j^{N_1} \left(w_j^{(1)}\right)^2\right) = \mathcal{N}\left(\mathrm{z};\, 0, \sum_j^{N_1} \left(w_j^{(1)} - 0\right)^2\right) \xrightarrow{N_1 \to \infty} \mathcal{N}\left(\mathrm{z};\, 0, \sigma_w^2\right) \tag{139}$$

We are, again, in the same situation described for $h_i^{(1)}$ in Eq. 130.
This means that the null amount of IGB is the same as $\mathrm{Z}_i^{(1)}$ and it will not increase in the next blocks' output.

G Experiments

In this section, we present complementary experiments supporting our analysis; implementation details, including model specifications and datasets used, are provided in App. G.1.

G.1 Reproducibility

Here we provide technical details about the experiments, to allow for reproducibility. The code used for the experiments presented in this work is available at `https://github.com/EmanueleFrancazi/IGB-and-Normalization`

Datasets

- **Gaussian Blob** (`GB`): Consistent with the presented analysis, for most of the experiments shown we used a Gaussian blob as input. Specifically, all elements of the dataset are *i.i.d.*, and each individual element consists of a random vector of dimension $d = 1000$ with normally distributed components. For binary experiments, we generate two Gaussian blobs centered at $\pm\mu$, where the mean μ scales with the input dimension to ensure a non-vanishing class overlap even in the high-dimensional limit. In particular, we set $\mu = 1/\sqrt{d}$, such that:

$$\mathrm{X}_i^{(0)} \sim \begin{cases} \mathcal{N}(-\mu, \mathbf{I}) & \text{if } y_i = 0\,, \\ \mathcal{N}(+\mu, \mathbf{I}) & \text{if } y_i = 1\,. \end{cases}$$

  This setup provides a controlled and analytically tractable scenario for evaluating binary classification in high dimensions.

- **CIFAR10** (`C10`): We use `CIFAR10` (`https://www.cs.toronto.edu/~kriz/cifar.html`) (Krizhevsky et al., 2009) as an example of a real multi-class dataset. Before the start of the simulation, we perform standardization: pixel values are rescaled to the interval $[0, 1]$, followed by channel-wise mean subtraction and standard deviation normalization.

  While we present multi-class experiments on more advanced architectures, for MLP-based experiments we focus on binary classification tasks for greater interpretability. To this end, we define two different binary versions of CIFAR10:

  - **Cat vs Dog** (`C2-A`): A binary classification task where we extract only the `cat` and `dog` classes from the full dataset.
  - **Vehicles vs Animals** (`C2-B`): A broader binary task where we aggregate multiple classes into two macro categories: vehicles (`airplane`, `automobile`, `truck`) and animals (`cat`, `deer`, `dog`, `horse`).

  These setups allow us to assess the behavior of the models under both narrowly and broadly defined binary distinctions.

- **Tiny ImageNet** (`TI`): A subset of the ImageNet dataset, consisting of 200 classes with 500 training images, 50 validation images, and 50 test images per class, each downsampled to 64×64 pixels. The dataset is publicly available at `https://www.kaggle.com/c/tiny-imagenet`.

- **MNIST** (`E&O`): We use `MNIST` (`http://yann.lecun.com/exdb/mnist/`) (Deng, 2012) to reproduce binary experiments on real data. The binary dataset is defined by merging the digit classes into two macro groups based on parity: the `even` class $\{0, 2, 4, 6, 8\}$ and the `odd` class $\{1, 3, 5, 7, 9\}$.

Models We provide details on the architectures used in our experiments. Our description does not include loss functions, as they are irrelevant in the untrained networks analyzed.

- **MLP**: Our analysis provides theoretical predictions for MLPs. To support our study, we consider two different MLP configurations:

  - **MLP with a single hidden layer** (`SHLP`): The hidden layer contains $N_1 = 100$ nodes.
  - **MLP with multiple hidden layers** (`MHLP`): This network has $L = 20$ hidden layers, each with $N = 100$ nodes.

Each MLP model employs ReLU activation, which is a widespread example of an activation function known to induce IGB and to amplify it with network depth (Francazi et al., 2024). This setting serves as our baseline, previously analyzed in (Francazi et al., 2024). In our experiments, we compare this baseline to variants that incorporate normalization either before or after the activation, as discussed in Appendix A.

- **MLP-mixer**: We include the MLP-mixer (`MLP-mixer`), introduced in Tolstikhin et al. (2021), as an example of a more advanced MLP architecture. We adopt the design described at `https://github.com/omihub777/MLP-Mixer-CIFAR/blob/main/README.md`.
- **ResNet34**: We consider ResNet34 (`ResNet34`), proposed in He et al. (2016), as an example of an architecture with skip connections. We use the implementation available at `https://www.kaggle.com/code/kmldas/cifar10-resnet-90-accuracy-less-than-5-min`.
- **ResNet101** (`ResNet101`): As a deeper residual network, we also consider ResNet101, following the implementation available at `https://github.com/mbk2103/ResNet101-Implementation/blob/main/resnet_model.py`.
- **Swin Transformer (Large)** (`Swin-L`): Finally, we include the Swin Transformer (Large) model (Liu et al., 2021), representative of modern transformer-based architectures. We rely on the implementation available at `https://github.com/WangFeng18/Swin-Transformer/blob/master/SwinTransformer.py`.

We emphasize that these architectures were not chosen to represent the state of the art, but rather as well-documented, accessible examples capable of achieving good performance with relatively low computational cost. This makes our findings regarding IGB more broadly applicable and facilitates reproducibility.

G.2 Experiments on MLPs

We begin by analyzing MLP architectures, where theoretical predictions regarding IGB are most directly applicable. In all experiments, random initializations are filtered based on the initial class bias G_0, allowing us to compare *neutral* initializations ($G_0 \approx 0.5$) to *prejudiced* ones ($G_0 \approx 1$). This filtering enables us to isolate how class bias at initialization affects training.
A central aim of our experimental analysis is to examine how well the qualitative behavior predicted by our theory—formulated under unstructured data assumptions—translates to real-world scenarios. While our theoretical framework is developed for random, high-dimensional inputs, we empirically assess whether similar dynamics are observed in practical settings, and identify where significant deviations emerge. This comparison offers insight into which behaviors are driven primarily by architectural design and which arise from interactions with data structure. Although a full analysis of these effects is beyond the scope of this work, our findings highlight promising directions for future study into the complex interplay between architecture-induced and data-induced biases. To this end, we extend our analysis beyond the Gaussian blob setting and consider `C2-B` and `E&O` as two representative examples of structured real datasets. The comparison allows us to underline both the consistency and the divergence between theoretical predictions and empirical behavior in more realistic conditions. All dynamic experiments are repeated over multiple random seeds; plots report the mean and standard error across runs to account for variability.

Gaussian Blob – random data. We begin with `GB`, a high-dimensional synthetic dataset without meaningful correlations. Results (Figs. G.1, G.2) replicate theoretical predictions: normalization placement critically determines the level of IGB, which in turn shapes both convergence speed and per-class imbalance. Models with normalization after ReLU exhibit faster reabsorption of initial bias and improved convergence, especially under LN.

CIFAR10 – empirical evaluation on structured inputs. Next, we evaluate `C2-B`, a binary subset of CIFAR10. Despite being a structured real dataset, results (Figs. G.3, G.4) remain qualitatively consistent with those on `GB`. This suggests that our theory captures dominant effects even outside the regime of its assumptions. Normalization placement still controls IGB level, and bias-driven convergence effects remain visible.

MNIST – highly correlated data. To explore the effect of data structure more explicitly, we include experiments on `E&O`. We select MNIST as a prototype of strongly correlated inputs, since prior work on IGB (Francazi et al., 2024) argued that datasets, where the correlations between different instances are strong (such as MNIST, where

pixels at the corners are always white), should exhibit more IGB than what is predicted by a theory based on random uncorrelated inputs. Here (Figs. G.5, G.6), we observe a qualitative departure: LN-after-ReLU reduces IGB without completely eliminating it—contrary to the behavior on `GB` and `C2-B`—as predicted by the arguments in Ref. (Francazi et al., 2024). As we can see by comparing the inset of LN plots in Figs. G.5, G.6 with the corresponding insets in Figs. G.1, G.2, the distribution profiles differ in detail; nevertheless, the behaviour induced by the normalization placement remains consistent with our theory: placing the normalization *before* the activation leads to more biased behaviour than placing it *after*.
Interestingly, BN-after-ReLU continues to exhibit near-neutral initial states, consistent with our theory and highlighting BN's stronger ability to reduce IGB, as our theorems also indicate. We stress that one does not necessarily always want to completely eliminate IGB (Bassi et al., 2026), so the optimal configuration depends on the details of the problem. Nevertheless, our analysis provides a clearer understanding of the mechanisms underlying the initial bias and identifies practical design levers—most notably the placement of normalization relative to activation—that allow one to tune and shape the initial predictive behaviour through principled network design, rather than merely mitigate it.

Robustness: Deep MLPs. Finally, we repeat the `GB` experiments on a deep `MHLP` architecture to test the robustness of our findings (Figs. G.7, G.8). While per-class accuracy still reflects IGB differences, global accuracy curves largely overlap across normalization placements. This suggests that, in deep models, early-stage bias may still shape local dynamics but not necessarily affect global convergence speed.

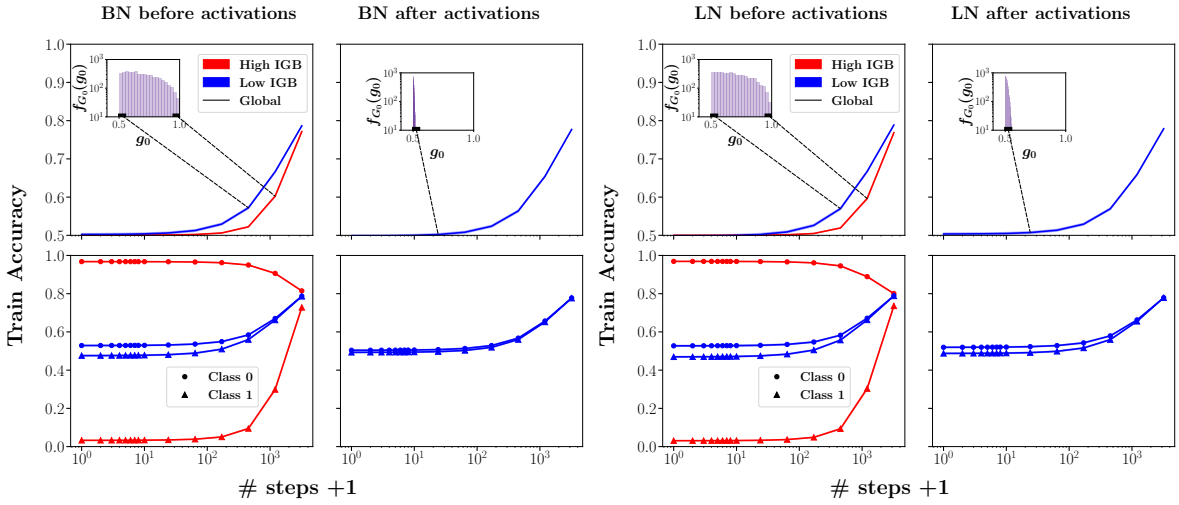

Figure G.1: Training accuracy on `GB` using the shallow `SHLP` model. Left: BN; Right: LN. Each panel compares normalization applied *before* vs *after* ReLU. Within each configuration, curves are averaged over multiple runs and grouped by initialization bias (*neutral*: $G_0 \approx 0.5$, *prejudiced*: $G_0 \approx 1$). Top: global accuracy; bottom: per-class accuracy. See App. G.1 for model `SHLP` and dataset `GB` details. Learning rate: `1e-5`, batch size: `512`.

G.3 Experiments on Other Architectures

While our theory focuses on MLPs, we now investigate how its insights transfer to more complex, widely used architectures. A comprehensive study across all architectures and datasets is outside the scope of this paper; instead, we provide complementary evidence in realistic settings. We report initialization results on large, multi-class settings (`ResNet101`, `Swin-L`) and training-dynamics experiments on (`ResNet34`, `MLP-mixer`). This choice of settings allows us to present results on advanced models and realistic tasks while avoiding massive compute requirements, thereby supporting reproducibility. Across all experiments, we rely on established implementations, using the documented tuned choices of hyperparameters as baselines; the only *variant* introduced for comparison is the placement of the normalization layer.

Normalization placement and initialization bias. For regimes with many classes (N_C large), the neutral fraction $1/N_C$ is small, making the maximum per-class guess fraction $G_{\tilde{0}}$ a sensitive summary of bias at initialization. We compare the distribution of $G_{\tilde{0}}$ between the baseline configuration (*normalization + activation*)

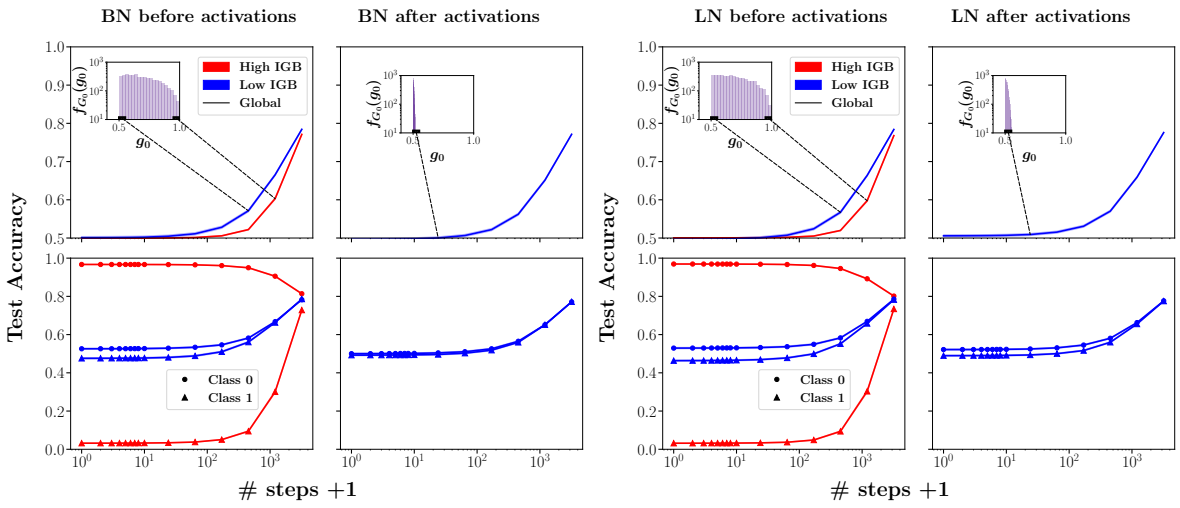

Figure G.2: Test accuracy on `GB` using the shallow `SHLP` model. Left: BN; Right: LN. Each panel compares normalization applied *before* vs *after* ReLU. Top: global accuracy; bottom: per-class accuracy. Curves are grouped by initialization bias. See App. G.1 for model `SHLP` and dataset `GB` details. Learning rate: `1e-5`, batch size: `512`.

and a variant with normalization moved *after* the activation ((*activation* + *normalization*)), for `ResNet101` and `Swin-L`. In both models, placing normalization after the activation yields a right-shift of the $G_{\tilde{0}}$ distribution, indicating an increased typical fraction of datapoints absorbed by the most-guessed class at initialization. This shift signals a higher initial IGB level and is coherent with our theoretical expectations (Fig. G.9; additional details in App. G.1).

Normalization placement and training dynamics. We next study the impact that the normalization placement has on the learning process, using `ResNet34` and `MLP-mixer`. We emphasize that across all experiments the architecture and hyperparameters follow well-documented baselines; the only intervention on the model is the positioning of the normalization layer to define a variant for the comparison with the baseline. To tune the IGB level without altering architectural details, we apply a simple pre-processing step to the input data, adding a constant value to all pixels, as prescribed in Ref. (Francazi et al., 2024). We compare the default setting with a variant in which normalization layers are moved *after* the activation functions (see Figs. G.10, G.11).

Our experiments show consistent results with theoretical expectations. Moving normalization after the nonlinearity reduces IGB and accelerates convergence. For the MLP-Mixer (Fig. G.11), we further vary the IGB level by adjusting the data standardization procedure. We observe that as IGB increases, overall training slows in both configurations, but the gap in convergence speed between the two normalization placements widens. This suggests a compounding interaction between IGB strength and normalization structure, reinforcing the role of architectural choices in determining early dynamics. All experiments on the dynamics are repeated over multiple random seeds; plots report the mean and standard error across runs to account for variability.

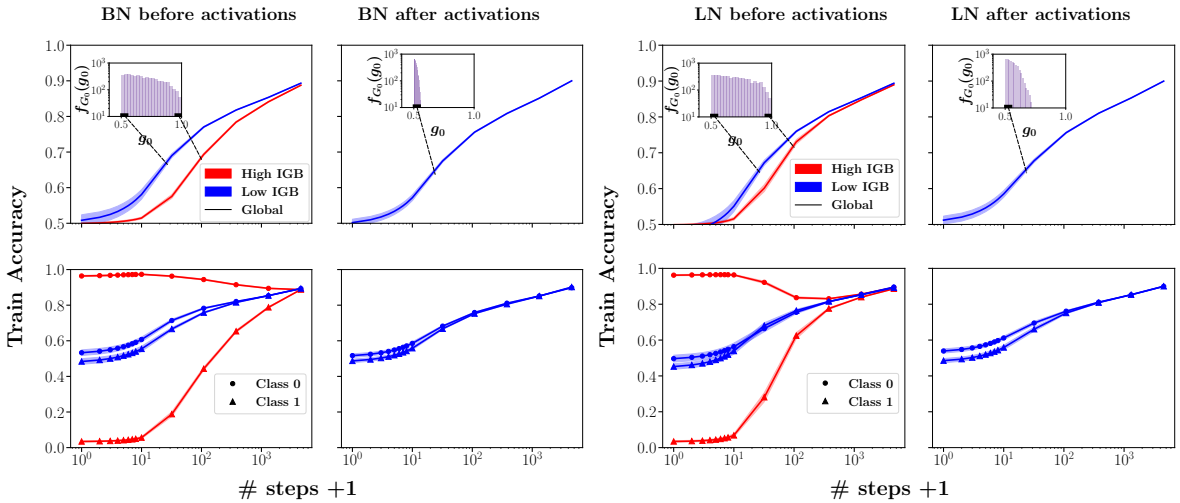

Figure G.3: Training accuracy on `C2-B` using the shallow `SHLP` model. Left: BN; Right: LN. Each panel compares normalization *before* vs *after* ReLU. Top: global accuracy; bottom: per-class accuracy. See App. G.1 for model `SHLP` and dataset `C2-B` details. Learning rate: `1e-5`, batch size: `512`.

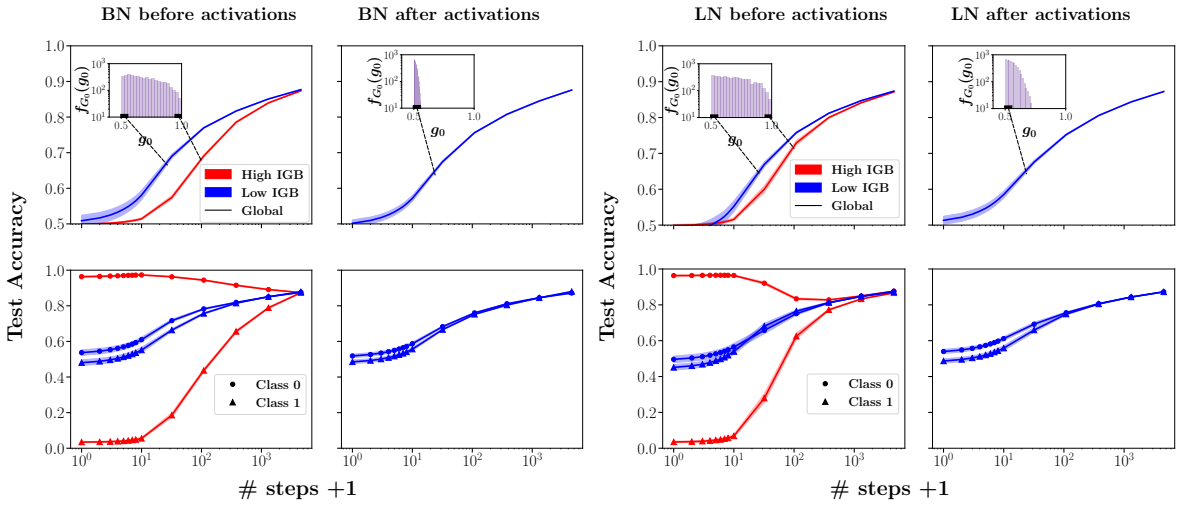

Figure G.4: Test accuracy on `C2-B` using the shallow `SHLP` model. Left: BN; Right: LN. Global (top) and per-class (bottom) performance trends mirror those observed on random data. See App. G.1 for model `SHLP` and dataset `C2-B` details. Learning rate: `1e-5`, batch size: `512`.

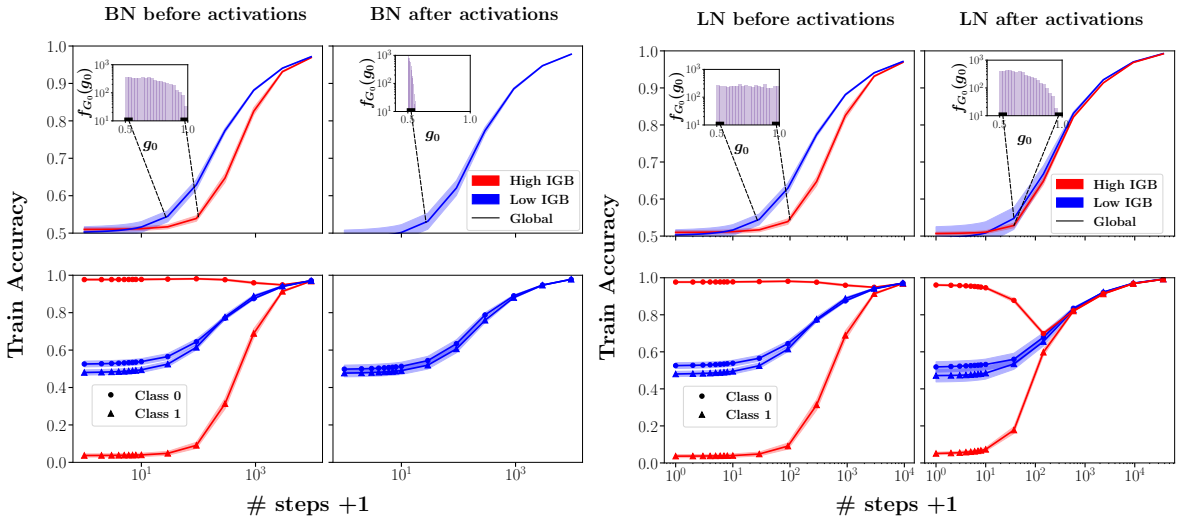

Figure G.5: Training accuracy on `E&O` with the shallow `SHLP` model. Left: BN; Right: LN. LN-after-ReLU reduces IGB (as can be seen from the insets), though it does not eliminate it. Top: global accuracy; bottom: per-class accuracy. See App. G.1 for model `SHLP` and dataset `E&O` details. Learning rate: `1e-5`, batch size: `512`.

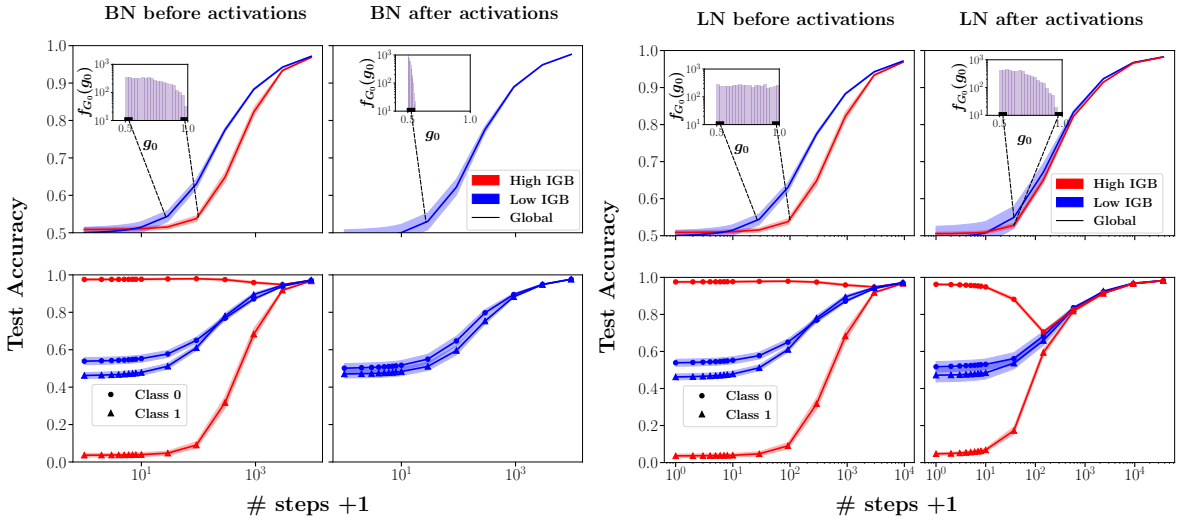

Figure G.6: Test accuracy on `E&O` with the shallow `SHLP` model. Left: BN; Right: LN. LN-after-ReLU results in broader bias distributions and persistent asymmetries. See App. G.1 for model `SHLP` and dataset `E&O` details. Learning rate: `1e-5`, batch size: `512`.

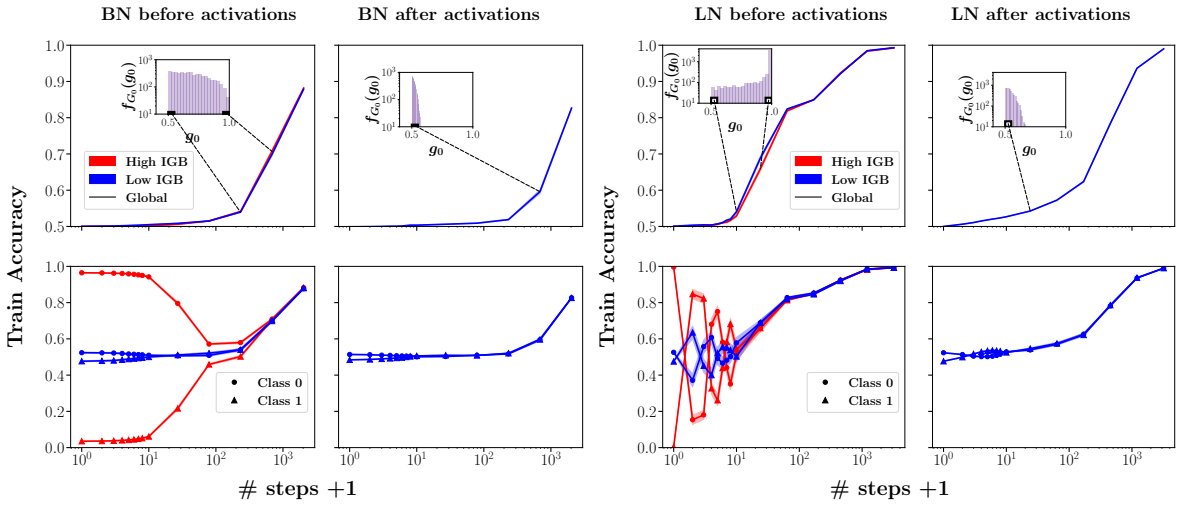

Figure G.7: Training accuracy on GB using the deep MHLP model. Left: BN; Right: LN. Per-class accuracy reflects IGB differences, but global convergence speeds remain similar. See App. G.1 for model MHLP and dataset GB details. Learning rate: 1e-3 for LN, 5e-4 for BN. Batch size: 512.

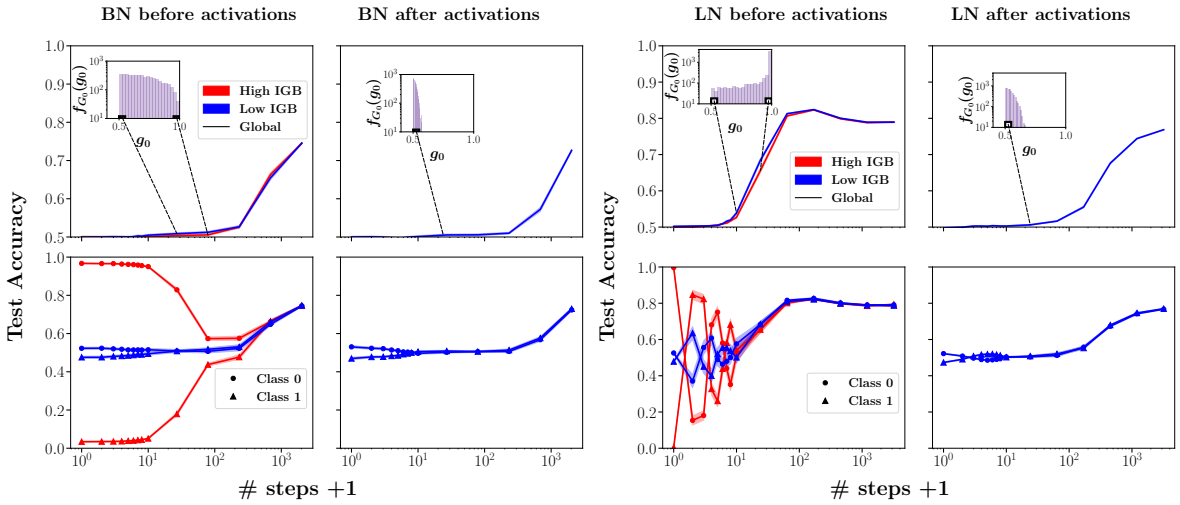

Figure G.8: Test accuracy on GB with the deep MHLP model. Left: BN; Right: LN. Despite clear differences in per-class behavior, global accuracy curves largely overlap. See App. G.1 for model MHLP and dataset GB details. Learning rate: 1e-3 for LN, 5e-4 for BN. Batch size: 512.

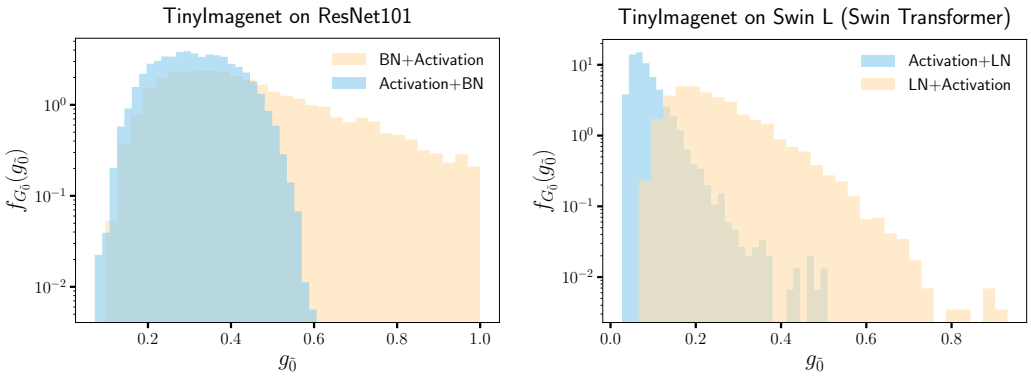

Figure G.9: **Initialization bias at scale.** Distributions of the maximum per-class guess fraction $G_{\tilde{0}}$ under baseline (*norm + act.*) vs. *act. + norm* for `ResNet101` (left) and `Swin-L` (right) on high-cardinality datasets (`TI`). In both cases, moving normalization after the activation produces a right-shift, i.e., larger typical $G_{\tilde{0}}$, indicating increased initial IGB.

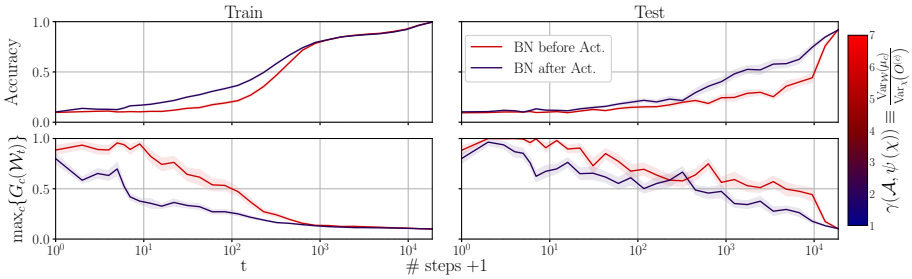

Figure G.10: Training dynamics of `ResNet34` on `C10`, comparing the standard configuration with BN placed before the activation (BN + Act.) and a modified variant with BN applied after the activation (Act. + BN). The level of Initial Guessing Bias (IGB) is measured using the variance ratio $\gamma(\mathcal{A}, \psi(\chi))$, computed from the network's initial outputs. As predicted by our theoretical analysis, applying BN after the activation reduces $\gamma(\mathcal{A}, \psi(\chi))$ in this setting, yielding a more balanced initial predictive state. Results are consistent with trends observed in MLP-based architectures.

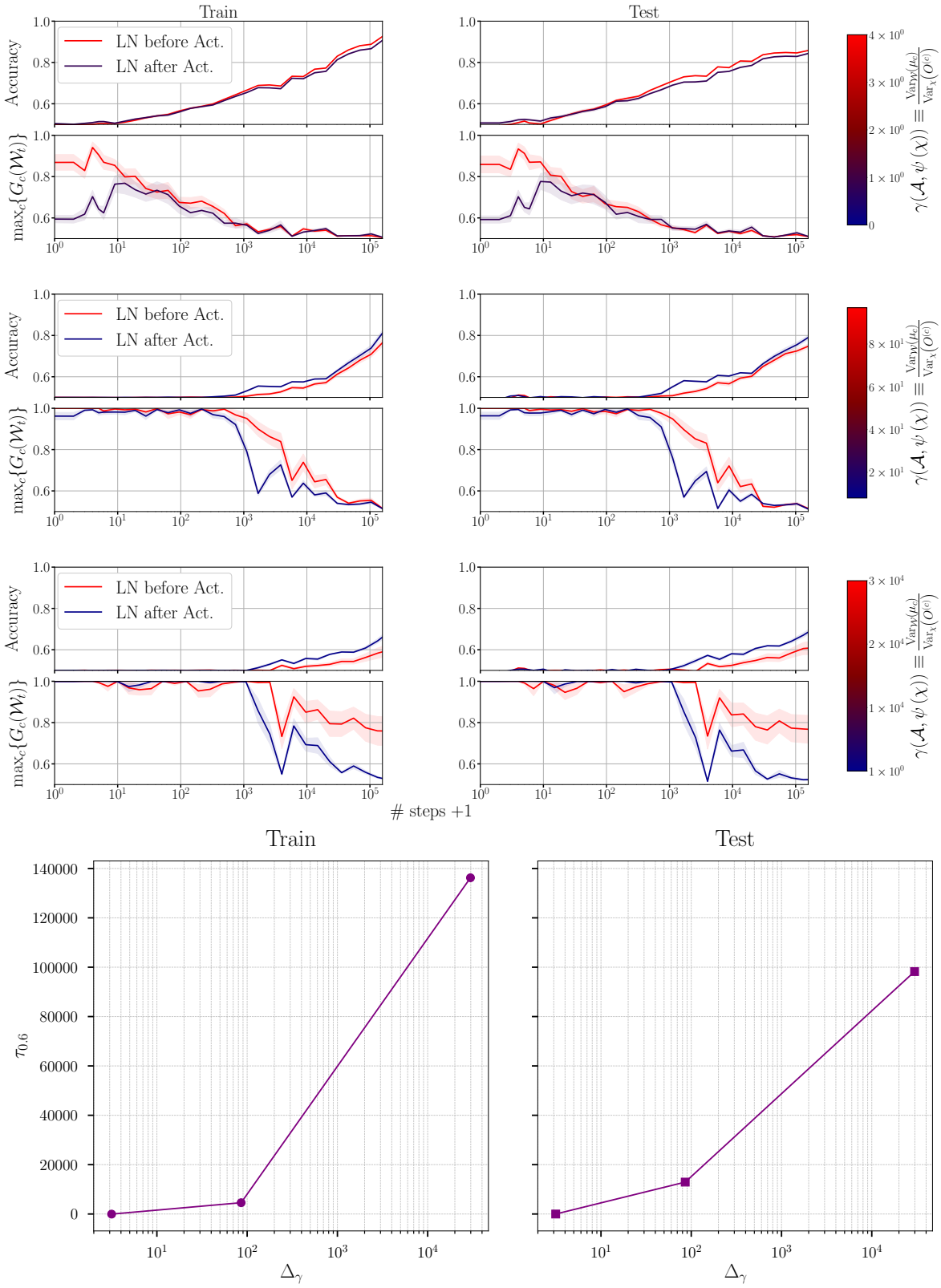

Figure G.11: Learning dynamics of `MLP-mixer` on `C2-A` with increasing levels of IGB (top to bottom), induced by progressively shifting input distributions via standardization (adding a constant to all pixels). Each panel compares the canonical configuration with LN applied *before* the activation (LN + Act.) to a variant with LN applied *after* the activation (Act. + LN). For each setting, we report accuracy (left) and the maximum class prediction bias $\max_c\{G_c\}$ (right), averaged over multiple runs (mean ± standard error). Left column: training data; right column: test data. In the baseline setting (top), both configurations perform similarly, but as IGB increases, LN-before-activation results in slower convergence, consistent with theoretical predictions. Panel (d) summarizes the convergence delay ($\tau_{0.6}$) versus Δ_γ, the difference in $\gamma(\mathcal{A},\psi(\chi))$ between the two configurations.