Proxy-Normalizing Activations to Match Batch Normalization while Removing Batch Dependence

Antoine Labatie **Dominic Masters** **Zach Eaton-Rosen** **Carlo Luschi**
Graphcore Research, UK
antoine.labatie@centraliens.net
{dominicm, zacher, carlo}@graphcore.ai

Abstract

We investigate the reasons for the performance degradation incurred with batch-independent normalization. We find that the prototypical techniques of layer normalization and instance normalization both induce the appearance of failure modes in the neural network's pre-activations: (i) layer normalization induces a collapse towards channel-wise constant functions; (ii) instance normalization induces a lack of variability in instance statistics, symptomatic of an alteration of the expressivity. To alleviate failure mode (i) without aggravating failure mode (ii), we introduce the technique "Proxy Normalization" that normalizes post-activations using a proxy distribution. When combined with layer normalization or group normalization, this batch-independent normalization emulates batch normalization's behavior and consistently matches or exceeds its performance.

1 Introduction

Normalization plays a critical role in deep learning as it allows the successful scaling to large and deep models. In vision tasks, the most well-established normalization technique is Batch Normalization (BN) [1]. At every layer in the network, BN normalizes the intermediate activations to have zero mean and unit variance in each channel. While indisputably successful when training with large batch size, BN incurs a performance degradation in the regime of small batch size [2, 3, 4, 5, 6, 7]. This performance degradation is commonly attributed to an excessive or simply unwanted regularization stemming from the noise in the approximation of full-batch statistics by mini-batch statistics.

Many techniques have been proposed to avoid this issue while at the same time retaining BN's benefits. Some techniques mimic BN's principle while decoupling the computational batch from the normalization batch [2, 8, 7]. Other techniques are "batch-independent" in that they operate independently of the batch in various modalities: through an explicit normalization either in activation space [9, 10, 11, 12, 13, 3, 5, 14, 15, 6] or in weight space [16, 17, 18, 19, 20]; through the use of an analytic proxy to track activation statistics [21, 22, 23]; through a change of activation function [24, 5, 15].

In this paper, we push the endeavor to replace BN with a batch-independent normalization a step further. Our main contributions are as follows: (i) we introduce a novel framework to finely characterize the various neural network properties affected by the choice of normalization; (ii) using this framework, we show that BN's beneficial properties *are not* retained when using solely the prototypical batch-independent normalization techniques, but *are* retained when using an approach that combines some of these techniques with Proxy Normalization, a novel technique that we hereby introduce; (iii) we demonstrate on an extensive set of experiments that, by reproducing BN's beneficial properties, our batch-independent normalization approach consistently matches or exceeds BN's performance.

As a starting point of our analysis, we need to gain a better understanding of those beneficial properties of BN that we aim at reproducing.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

2 Batch Normalization's beneficial properties

We consider throughout this paper a convolutional neural network with $d = 2$ spatial axes. This neural network receives an input $\mathbf{x} \in \mathbb{R}^{H \times W \times C_0}$ which, unless otherwise stated, is assumed sampled from a finite dataset $\mathcal{D}$. The neural network maps its input $\mathbf{x}$ to intermediate activations $\mathbf{x}^l \in \mathbb{R}^{H \times W \times C_l}$ of *height* H, *width* W and *number of channels* C_l at each layer l. The value of $\mathbf{x}^l$ at spatial position $\alpha \in \{1, \ldots, H\} \times \{1, \ldots, W\}$ and channel $c \in \{1, \ldots, C_l\}$ is denoted as $\mathbf{x}^l_{\alpha,c}$, with the dependency on $\mathbf{x}$ kept *implicit* to avoid overloading notations.

The inclusion of BN at layer l leads in the full-batch setting to adding the following operations $\forall \alpha, c$:

$$\mathbf{y}^l_{\alpha,c} = \frac{\mathbf{x}^l_{\alpha,c} - \mu_c(\mathbf{x}^l)}{\sigma_c(\mathbf{x}^l)}, \qquad \tilde{\mathbf{y}}^l_{\alpha,c} = \boldsymbol{\gamma}^l_c \mathbf{y}^l_{\alpha,c} + \boldsymbol{\beta}^l_c, \tag{1}$$

where $\mu_c(\mathbf{x}^l)$, $\sigma_c(\mathbf{x}^l)$ are the mean and standard deviation of $\mathbf{x}^l$ in channel c, and $\boldsymbol{\gamma}^l_c$, $\boldsymbol{\beta}^l_c$ are channel-wise scale and shift parameters restoring the degrees of freedom lost in the standardization. In the mini-batch setting, the full-batch statistics $\mu_c(\mathbf{x}^l)$, $\sigma_c(\mathbf{x}^l)$ are approximated with mini-batch statistics.

Table 1 summarizes the beneficial properties that result from including BN in the neural network. Below, we provide details on each of these properties, and we discuss whether each property is reproduced with batch-independent norms.

Scale invariance. When BN is present, the input-output mapping of the neural network is invariant to the scale of the weights preceding any BN layer. With such scale invariance plus weight decay, the scale of weights during training reaches an equilibrium with an effective learning rate depending on both the learning rate and the weight decay strength [25, 26, 27, 8, 28, 29, 30]. Such mechanism of "auto rate-tuning" has been shown to provide optimization benefits [31, 26, 32].

This property is easy to reproduce. It is already obtained with most existing batch-independent norms.

Control of activation scale in residual networks. To be trainable, residual networks require the scale of activations to remain well-behaved at initialization [33, 34, 35, 20, 36, 37]. While this property naturally arises when BN is present on the residual path, when BN is not present it can also be enforced by a proper scaling decaying with the depth of the residual path. This "dilution" of the residual path with respect to the skip connection path reduces the effective depth of the network and thereby enables to avoid coarse-grained failures modes [38, 39, 40, 41, 42, 35].

This property is easy to reproduce. It is already obtained with most existing batch-independent norms.

Regularizing noise. Due to the stochasticity of the approximation of $\mu_c(\mathbf{x}^l)$, $\sigma_c(\mathbf{x}^l)$ by mini-batch statistics, training a neural network with BN in the mini-batch setting can be seen as equivalent to performing Bayesian inference [43, 44] or as adding a regularizing term to the training of the same network with full-batch statistics [45]. As a result, BN induces a specific form of regularization.

This regularization is not reproduced with batch-independent norms, but we leave it out of the scope of this paper. To help minimize the bias in our analysis and "subtract away" this effect, we perform all our experiments without and with extra degrees of regularization. This procedure can be seen as a coarse disentanglement of normalization's effects from regularization's effects.

Avoidance of collapse. Unnormalized networks with non-saturating nonlinearities are subject to a phenomenon of "collapse" whereby the distribution with respect to $\mathbf{x}$, α of the intermediate activation vectors $(\mathbf{x}^l_{\alpha,1}, \ldots, \mathbf{x}^l_{\alpha,C_l})^{\mathrm{T}}$ becomes close to zero- or one-dimensional in deep layers [39, 46, 41, 47, 48, 49, 37]. This means that deep in an unnormalized network: (i) layers tend to have their channels imbalanced; (ii) nonlinearities tend to become channel-wise linear with respect to $\mathbf{x}$, α and not add any effective capacity [39, 50, 51]. Consequently, unnormalized networks can neither effectively use their whole width (imbalanced channels) nor effectively use their whole depth (channel-wise linearity).

Conversely, when BN is used, the standardization at each layer prevents this collapse from happening. Even deep in the network, layers still have balanced channels and nonlinearities remain nonlinear with respect to $\mathbf{x}$, α. Consequently, networks with BN can effectively use their whole width and depth.

The collapse is, on the other hand, not always avoided with batch-independent norms [39, 18, 49]. Most notably, it is *not* avoided with Layer Normalization (LN) [9] and Group Normalization (GN) [3], as we show both theoretically and experimentally on commonly found networks in Section 4.

To the extent possible, we aim at designing a batch-independent norm that avoids this collapse.

Table 1: **BN's beneficial properties**. We show whether each property is (at least approximately) present (✓) or absent (✗) with different batch-independent norms: Instance Normalization (IN), Layer Normalization (LN), Layer Normalization + Proxy Normalization (LN+PN, cf Section 5). In this categorization, BN is considered in the mini-batch setting but still close to the full-batch setting, such that it approximately preserves expressivity.

	Scale invariance	Control of activation scale	Regularizing noise	Avoidance of collapse	Preservation of expressivity
BN	✓	✓	✓	✓	✓
IN	✓	✓	✗	✓	✗
LN	✓	✓	✗	✗	✓
LN+PN	✓	✓	✗	✓	✓

Preservation of expressivity. We can always express the identity with Eq. (1) by choosing $\boldsymbol{\beta}^l_c = \mu_c(\mathbf{x}^l)$ and $\boldsymbol{\gamma}^l_c = \sigma_c(\mathbf{x}^l)$. Conversely, for any choice of $\boldsymbol{\beta}^l_c$, $\boldsymbol{\gamma}^l_c$, we can always "re-absorb" Eq. (1) into a preceding convolution with bias. This means that BN in the full-batch setting *does not alter the expressivity* compared to an unnormalized network, i.e. it amount to a plain reparameterization of the hypothesis space.

The expressivity is, on the other hand, not always preserved with batch-independent norms. In activation space, the dependence of batch-independent statistics on the input $\mathbf{x}$ turns the standardization into a channel-wise nonlinear operation that cannot be "re-absorbed" into a preceding convolution [18]. This phenomenon is most pronounced when statistics get computed over few components. This means e.g. that Instance Normalization (IN) [10] induces a greater change of expressivity than GN, which itself induces a greater change of expressivity than LN.

In weight space, the expressivity can also be altered, namely by the removal of degrees of freedom. This is the case with Weight Standardization (WS) [18, 20] and Centered Weight Normalization [17] that remove degrees of freedom (one per unit) that *cannot* be restored in a succeeding affine transformation. This reduction of expressivity could explain the ineffectiveness of these techniques in EfficientNets [52], as observed in previous works [20] and as we confirm in Section 6.

To the extent possible, we aim at designing a batch-independent norm that preserves expressivity.

3 Theoretical framework of analysis

We outlined the properties that we wish to retain in our design of batch-independent normalization: (i) scale invariance, (ii) control of activation scale; (iii) avoidance of collapse; (iv) preservation of expressivity. We now introduce a framework to quantify the presence or absence of the specific properties (iii) and (iv) with various choices of normalization.

Propagation. For simplicity, we assume in our theoretical setup that any layer l up to depth L consists of the following three steps: (i) convolution step with weights $\boldsymbol{\omega}^l \in \mathbb{R}^{K_l \times K_l \times C_{l-1} \times C_l}$; (ii) normalization step; (iii) activation step sub-decomposed into an affine transformation with scale and shift parameters $\boldsymbol{\gamma}^l, \boldsymbol{\beta}^l \in \mathbb{R}^{C_l}$ and an activation function ϕ which, unless otherwise stated, is assumed positive homogeneous and nonzero (e.g. $\phi = \mathrm{ReLU}$). If we denote $\mathbf{x}^l, \mathbf{y}^l, \mathbf{z}^l \in \mathbb{R}^{H \times W \times C_l}$ the intermediate activations situated just after (i), (ii), (iii) with the convention $\mathbf{z}^0 \equiv \mathbf{x}$, we may write the propagation through layer l as

$$\mathbf{x}^l = \mathrm{Conv}(\mathbf{z}^{l-1}), \qquad \forall \alpha, c: \quad \mathrm{Conv}(\mathbf{z}^{l-1})_{\alpha,c} = (\boldsymbol{\omega}^l * \mathbf{z}^{l-1})_{\alpha,c}, \tag{2}$$

$$\mathbf{y}^l = \mathrm{Norm}(\mathbf{x}^l), \qquad \forall \alpha, c: \quad \mathrm{Norm}(\mathbf{x}^l)_{\alpha,c} = \frac{\mathbf{x}^l_{\alpha,c} - \mu_{I_{\mathbf{x},c}}(\mathbf{x}^l)}{\sigma_{I_{\mathbf{x},c}}(\mathbf{x}^l)}, \tag{3}$$ [1]

$$\mathbf{z}^l = \mathrm{Act}(\mathbf{y}^l), \qquad \forall \alpha, c: \quad \mathrm{Act}(\mathbf{y}^l)_{\alpha,c} = \phi\big(\tilde{\mathbf{y}}^l_{\alpha,c}\big) = \phi\big(\boldsymbol{\gamma}^l_c \mathbf{y}^l_{\alpha,c} + \boldsymbol{\beta}^l_c\big), \tag{4}$$

where $\mu_{I_{\mathbf{x},c}}(\mathbf{x}^l)$, $\sigma_{I_{\mathbf{x},c}}(\mathbf{x}^l)$ denote the mean and standard deviation of $\mathbf{x}^l$ conditionally on $I_{\mathbf{x},c} \equiv \{c\}$, $\{\mathbf{x}\}$, $\{\mathbf{x}, c\}$, $\{\mathbf{x}, c \bmod G\}$ for the respective cases $\mathrm{Norm} = \mathrm{BN}, \mathrm{LN}, \mathrm{IN}, \mathrm{GN}$ with G groups.

[1] We omit the numerical stability constant and adopt the convention $\mathrm{Norm}(\mathbf{x}^l)_{\alpha,c} = 0, \forall \alpha$ if $\sigma_{I_{\mathbf{x},c}}(\mathbf{x}^l) = 0$.

Moments. Extending the previous notations, we use μ, σ, $\mathcal{P}$ indexed with a (possibly empty) subset of variables to denote the operators of conditional mean, standard deviation and power. If we notably apply these operators to the intermediate activations $\mathbf{y}^l$ that implicitly depend on the input $\mathbf{x}$ and that depend on the spatial position α and the channel c, we get e.g.

$$\mu_c(\mathbf{y}^l) = \mathbb{E}_{\mathbf{x},\alpha}\left[\mathbf{y}^l_{\alpha,c}\right], \qquad \sigma_c(\mathbf{y}^l) = \sqrt{\mathrm{Var}_{\mathbf{x},\alpha}\left[\mathbf{y}^l_{\alpha,c}\right]}, \qquad \mathcal{P}_c(\mathbf{y}^l) = \underbrace{\mathbb{E}_{\mathbf{x},\alpha}\left[(\mathbf{y}^l_{\alpha,c})^2\right]}_{\sigma_c(\mathbf{y}^l)^2+\mu_c(\mathbf{y}^l)^2},$$

$$\mu_{\mathbf{x},c}(\mathbf{y}^l) = \mathbb{E}_{\alpha}\left[\mathbf{y}^l_{\alpha,c}\right], \qquad \sigma_{\mathbf{x},c}(\mathbf{y}^l) = \sqrt{\mathrm{Var}_{\alpha}\left[\mathbf{y}^l_{\alpha,c}\right]}, \qquad \mathcal{P}_{\mathbf{x},c}(\mathbf{y}^l) = \underbrace{\mathbb{E}_{\alpha}\left[(\mathbf{y}^l_{\alpha,c})^2\right]}_{\sigma_{\mathbf{x},c}(\mathbf{y}^l)^2+\mu_{\mathbf{x},c}(\mathbf{y}^l)^2},$$

where, by convention, $\mathbf{x}$, α, c are considered uniformly sampled among inputs of $\mathcal{D}$, spatial positions and channels whenever they are considered as random

Power decomposition. Using these notations, we may gain important insights by decomposing the power in channel c of $\mathbf{y}^l$, just after the normalization, as

$$\mathcal{P}_c(\mathbf{y}^l) = \underbrace{\mathbb{E}_{\mathbf{x}}\left[\mu_{\mathbf{x},c}(\mathbf{y}^l)\right]^2}_{\mathcal{P}_c^{(1)}(\mathbf{y}^l)} + \underbrace{\mathrm{Var}_{\mathbf{x}}\left[\mu_{\mathbf{x},c}(\mathbf{y}^l)\right]}_{\mathcal{P}_c^{(2)}(\mathbf{y}^l)} + \underbrace{\mathbb{E}_{\mathbf{x}}\left[\sigma_{\mathbf{x},c}(\mathbf{y}^l)\right]^2}_{\mathcal{P}_c^{(3)}(\mathbf{y}^l)} + \underbrace{\mathrm{Var}_{\mathbf{x}}\left[\sigma_{\mathbf{x},c}(\mathbf{y}^l)\right]}_{\mathcal{P}_c^{(4)}(\mathbf{y}^l)}. \tag{5}$$

Since this four-terms *power decomposition* will be at the core of our analysis, we detail two useful views of it. The first view is that of a *hierarchy of scales*: $\mathcal{P}_c^{(1)}(\mathbf{y}^l)$ measures the power of $\mu_c(\mathbf{y}^l)$ at the dataset scale; $\mathcal{P}_c^{(2)}(\mathbf{y}^l)$ measures the power of $\mu_{\mathbf{x},c}(\mathbf{y}^l) - \mu_c(\mathbf{y}^l)$ at the instance scale; the sum of $\mathcal{P}_c^{(3)}(\mathbf{y}^l)$, $\mathcal{P}_c^{(4)}(\mathbf{y}^l)$ measures the power of $\mathbf{y}^l_{\alpha,c} - \mu_{\mathbf{x},c}(\mathbf{y}^l)$ at the pixel scale. A particular situation where the power would be concentrated at the dataset scale with $\mathcal{P}_c^{(1)}(\mathbf{y}^l)$ equal to $\mathcal{P}_c(\mathbf{y}^l)$ would thus imply that $\mathbf{y}^l$ has its distribution fully "collapsed" in channel c, i.e. that $\mathbf{y}^l$ is constant in channel c.

The second view is that of a *two-level binary tree*: on one half of the tree, the sum of $\mathcal{P}_c^{(1)}(\mathbf{y}^l)$ and $\mathcal{P}_c^{(2)}(\mathbf{y}^l)$ measures the power coming from $\mu_{\mathbf{x},c}(\mathbf{y}^l)$, with the relative proportions of $\mathcal{P}_c^{(1)}(\mathbf{y}^l)$ and $\mathcal{P}_c^{(2)}(\mathbf{y}^l)$ functions of the inter-$\mathbf{x}$ similarity and inter-$\mathbf{x}$ variability of $\mu_{\mathbf{x},c}(\mathbf{y}^l)$; on the other half of the tree, the sum of $\mathcal{P}_c^{(3)}(\mathbf{y}^l)$ and $\mathcal{P}_c^{(4)}(\mathbf{y}^l)$ measures the power coming from $\sigma_{\mathbf{x},c}(\mathbf{y}^l)$, with the relative proportions of $\mathcal{P}_c^{(3)}(\mathbf{y}^l)$ and $\mathcal{P}_c^{(4)}(\mathbf{y}^l)$ functions of the inter-$\mathbf{x}$ similarity and inter-$\mathbf{x}$ variability of $\sigma_{\mathbf{x},c}(\mathbf{y}^l)$. A particular situation where $\mathcal{P}_c^{(2)}(\mathbf{y}^l)$, $\mathcal{P}_c^{(4)}(\mathbf{y}^l)$ would be equal to zero would thus imply that $\mu_{\mathbf{x},c}(\mathbf{y}^l)$, $\sigma_{\mathbf{x},c}(\mathbf{y}^l)$ have zero inter-$\mathbf{x}$ variability, i.e. that $\mu_{\mathbf{x},c}(\mathbf{y}^l)$, $\sigma_{\mathbf{x},c}(\mathbf{y}^l)$ are constant for all $\mathbf{x}$.

A version of Eq. (5) at the layer level instead of channel level will be easier to work with. Defining $\mathcal{P}^{(i)}(\mathbf{y}^l)$ as the averages of $\mathcal{P}_c^{(i)}(\mathbf{y}^l)$ over c for $i \in \{1, 2, 3, 4\}$, we obtain

$$\mathcal{P}(\mathbf{y}^l) = \mathcal{P}^{(1)}(\mathbf{y}^l) + \mathcal{P}^{(2)}(\mathbf{y}^l) + \mathcal{P}^{(3)}(\mathbf{y}^l) + \mathcal{P}^{(4)}(\mathbf{y}^l).$$

It should be noted that $\mathcal{P}(\mathbf{y}^l) = 1$ for any choice of $\mathrm{Norm} \in \{\mathrm{BN}, \mathrm{LN}, \mathrm{IN}, \mathrm{GN}\}$, as long as the denominator of Eq. (3) is nonzero for all $\mathbf{x}, c$ [C.2]. Consequently, the terms $\mathcal{P}^{(i)}(\mathbf{y}^l)$ sum to one, meaning they can be conveniently seen as the proportion of each term $i \in \{1, 2, 3, 4\}$ into $\mathcal{P}(\mathbf{y}^l)$.

Revisiting BN's avoidance of collapse. When BN is used, $\mathbf{y}^l$ is normalized not only layer-wise but also channel-wise with $\mathcal{P}_c^{(1)}(\mathbf{y}^l) = 0$ and $\mathcal{P}_c(\mathbf{y}^l) = 1$. As a first consequence, $\tilde{\mathbf{y}}^l$ (that is only one affine transformation away from $\mathbf{y}^l$) is *unlikely* to have its channel-wise distributions collapsed. This means that the nonlinearity ϕ acting on $\tilde{\mathbf{y}}^l$ is *likely* to be effectively nonlinear with respect to $\tilde{\mathbf{y}}^l$'s channel-wise distributions.[2] As a result, each layer adds capacity and the network effectively uses its whole *depth*. This is opposite to the situation where $\tilde{\mathbf{y}}^l$ has its channel-wise distributions collapsed with $\mathcal{P}_c(\tilde{\mathbf{y}}^l) - \mathcal{P}_c^{(1)}(\tilde{\mathbf{y}}^l) \ll \mathcal{P}_c(\tilde{\mathbf{y}}^l)$ for all c, which results in ϕ being close to linear with respect to $\tilde{\mathbf{y}}^l$'s channel-wise distributions. This is illustrated in Figure 1 and formalized in Appendix C.3.

[2]Note that: (i) this likeliness could be quantified in the context of random nets of Definition 1 with "reasonable" choices of β, γ; (ii) BN still only guarantees an intra-task nonlinearity and not an intra-mode nonlinearity in contexts such as adversarial training [53, 54] or conditional GANs [55], unless the modes are decoupled in BN's computation [56, 57, 58, 59, 60].

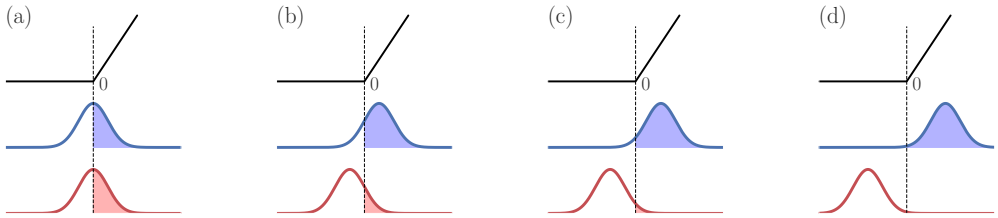

Figure 1: **Channel-wise collapse implies channel-wise linearity**. Each subplot shows $\phi = \mathrm{ReLU}$ (black activation function) as well as two channel-wise distributions (red and blue distributions) positioned symmetrically around 0 with $\frac{\mathcal{P}_c(\tilde{\mathbf{y}}^l) - \mathcal{P}_c^{(1)}(\tilde{\mathbf{y}}^l)}{\mathcal{P}_c(\tilde{\mathbf{y}}^l)} = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ for (a), (b), (c), (d), respectively. When progressing from (a) to (d), the part of the distribution corresponding to active ReLU (shaded region) becomes either overly dominant (blue distribution) or negligible (red distribution). In either case, the channel-wise distribution ends up concentrated on only one side of piece-wise linearity.

As an additional consequence, $\mathbf{y}^l$ is guaranteed to have its channels well balanced with equal power $\mathcal{P}_c(\mathbf{y}^l)$ for all c. As a result, the network effectively uses its whole *width*. This is opposite to the situation where a single channel c becomes overly dominant over the others with $\mathcal{P}_c(\mathbf{y}^l) \gg \mathcal{P}_{c'}(\mathbf{y}^l)$ for $c' \neq c$, which results in downstream layers only "seeing" this channel c and the network behaving as if it had a width equal to one at layer l.

Revisiting BN's preservation of expressivity. When BN is used, $\mathcal{P}_c^{(1)}(\mathbf{y}^l) = 0$ implies for all c that the terms $\mathcal{P}_c^{(2)}(\mathbf{y}^l), \mathcal{P}_c^{(3)}(\mathbf{y}^l), \mathcal{P}_c^{(4)}(\mathbf{y}^l)$ sum to one. Apart from that, BN does not impose any particular constraints on the relative proportions of each term into the sum. This means that the relative proportions of $\mathcal{P}_c^{(i)}(\mathbf{y}^l)$ and $\mathcal{P}^{(i)}(\mathbf{y}^l)$ for $i \in \{2, 3, 4\}$ are free to evolve as naturally dictated by the task and the optimizer during learning.

This absence of constraints seems sensible. Indeed, imposing constraints on these relative proportions would *alter the expressivity*, which would not have any obvious justification in general and could even be detrimental in some cases, as we discuss in Section 4.

4 Failure modes with batch-independent normalization

With our theoretical framework in hand, we now turn to showing that the prototypical batch-independent norms are subject to failures modes opposite to BN's beneficial properties.

In the case of LN, the failure mode does not manifest in an absolute sense but rather as a "soft" inductive bias, i.e. as a preference or a favoring in the hypothesis space. This "soft" inductive bias is quantified by Theorem 1 in the context of networks with random model parameters.

Definition 1 (random net)**.** *We define a "random net" as a neural network having an input* $\mathbf{x}$ *sampled from the dataset* $\mathcal{D}$ *and implementing Eq. (2), (3), (4) in every layer up to depth* L*, with the components of* $\boldsymbol{\omega}^l$*,*$\boldsymbol{\gamma}^l$*,*$\boldsymbol{\beta}^l$ *at each layer* $l \in \{1, \ldots, L\}$ *sampled i.i.d. from the fixed distributions* $\nu_{\boldsymbol{\omega}}$*,*$\nu_{\boldsymbol{\gamma}}$*,*$\nu_{\boldsymbol{\beta}}$ *(up to a fan-in's square root scaling for* $\boldsymbol{\omega}^l$*).*

In such networks, we assume that: (i) none of the inputs of the dataset $\mathcal{D}$ *is identically zero; (ii)* $\nu_{\boldsymbol{\omega}}$*,* $\nu_{\boldsymbol{\beta}}$*,* $\nu_{\boldsymbol{\gamma}}$ *have well-defined moments, with strictly positive associated root mean squares* $\omega, \gamma, \beta > 0$*; (iii)* $\nu_{\boldsymbol{\omega}}$*,* $\nu_{\boldsymbol{\beta}}$ *are symmetric around zero.*

Theorem 1 (layer-normalized networks collapse (informal))**.** [D.3] *Fix a layer* $l \in \{1, \ldots, L\}$ *and* $\nu_{\boldsymbol{\omega}}$*,* $\nu_{\boldsymbol{\beta}}$*,* $\nu_{\boldsymbol{\gamma}}$*,* $\mathcal{D}$ *in Definition 1. Further suppose* $\mathrm{Norm} = \mathrm{LN}$ *and suppose that the convolution of Eq. (2) uses periodic boundary conditions.*

Then for random nets of Definition 1, it holds when the width is large enough that

$$\mathcal{P}(\mathbf{y}^l) - \mathcal{P}^{(1)}(\mathbf{y}^l) \lesssim \rho^{l-1}, \qquad \mathcal{P}(\mathbf{y}^l) \simeq 1, \tag{6}$$

where $\rho \equiv \gamma^2/(\gamma^2 + \beta^2) < 1$*, and* $\lesssim$ *and* $\simeq$ *denote inequality and equality up to arbitrarily small constants with probability arbitrarily close to 1 when* $\min_{1 \leq k \leq l} C_k$ *is large enough.*

Discussion on LN's failure mode. Theorem 1 implies that, with high probability, $\mathbf{y}^l$ is subject to *channel-wise collapse* in deep layers ($l \gg 1$) with $\mathcal{P}(\mathbf{y}^l) - \mathcal{P}^{(1)}(\mathbf{y}^l) \ll \mathcal{P}(\mathbf{y}^l)$. This means that $\tilde{\mathbf{y}}^l$ (that is only one affine transformation away from $\mathbf{y}^l$) is likely to have its channel-wise distributions collapsed with $\mathcal{P}_c(\tilde{\mathbf{y}}^l) - \mathcal{P}_c^{(1)}(\tilde{\mathbf{y}}^l) \ll \mathcal{P}_c(\tilde{\mathbf{y}}^l)$ for most c. The nonlinearity ϕ acting on $\tilde{\mathbf{y}}^l$ is then likely to be close to linear with respect to $\tilde{\mathbf{y}}^l$'s channel-wise distributions [C.3]. Being close to channel-wise linear in deep layers, layer-normalized networks are unable to effectively use their whole depth.

Since the inequality $\lesssim$ can be replaced by an equality $\simeq$ in the case $\phi = \text{identity}$ of Theorem 1 [D.4], the aggravation at each layer l of the upper bound of Eq. (6) does not stem from the activation function itself but rather by the preceding affine transformation. The phenomenon of channel-wise collapse — also known under the terms of "domain collapse" [39] or "elimination singularity" [18] — is therefore not only induced by a "mean-shifting" activation function such as $\phi = \text{ReLU}$ [61, 20], but also by the injection of non-centeredness through the application of the channel-wise shift parameter $\boldsymbol{\beta}^l$ at each layer l. The fact that the general case of positive-homogeneous ϕ is upper-bounded by the case $\phi = \text{identity}$ in Eq. (6) still means that the choice $\phi = \text{ReLU}$ can only be an aggravating factor.

Crucially, in the context of random nets of large width, LN's operation at each layer l does not compensate this "mean shift". This stems from the fact that LN's mean and variance statistics can be approximated by zero and a constant value independent of $\mathbf{x}$, respectively. This means that LN's operation can be approximated by a layer-wise constant scaling independent of $\mathbf{x}$.[3]

The predominance of LN's failure mode in the hypothesis space, implied by its predominance in random nets, is expected to have at least two negative effects on the actual learning and final performance: (i) being expected to "linger" along the training trajectory and being associated with reduced effective capacity, the failure mode is expected to cause degraded performance on the training loss; (ii) even if avoided to some extent during training, the failure mode is still expected to be present in the vicinity of the training trajectory, implying an ill-conditioning of the loss landscape [61, 63] and a prohibition of large learning rates that could have led otherwise to generalization benefits [64].

After detailing LN's failure mode, we now detail IN's failure mode.

Theorem 2 (instance-normalized networks lack variability in instance statistics)**.** [E] *Fix a layer $l \in \{1, \dots, L\}$ and lift any assumption on ϕ. Further suppose* $\text{Norm} = \text{IN}$*, with Eq. (3) having nonzero denominator at layer l for all inputs and channels.*

Then it holds that

- $\mathbf{y}^l$ *is normalized in each channel c with*

$$\mathcal{P}_c^{(1)}(\mathbf{y}^l) = 0, \qquad \mathcal{P}_c(\mathbf{y}^l) = 1;$$

- $\mathbf{y}^l$ *lacks variability in instance statistics in each channel c with*

$$\mathcal{P}_c^{(2)}(\mathbf{y}^l) = 0, \qquad \mathcal{P}_c^{(3)}(\mathbf{y}^l) = 1, \qquad \mathcal{P}_c^{(4)}(\mathbf{y}^l) = 0.$$

Discussion on IN's failure mode. We see in Theorem 2 that $\mathbf{y}^l$'s power decomposition with IN is constrained to be such that $\mathcal{P}^{(1)}(\mathbf{y}^l), \mathcal{P}^{(2)}(\mathbf{y}^l), \mathcal{P}^{(4)}(\mathbf{y}^l) = 0$ and $\mathcal{P}^{(3)}(\mathbf{y}^l) = 1$. While the constraints on $\mathcal{P}^{(1)}(\tilde{\mathbf{y}}^l)$, $\mathcal{P}^{(3)}(\tilde{\mathbf{y}}^l)$ are removed by the affine transformation between $\mathbf{y}^l$ and $\tilde{\mathbf{y}}^l$, the constraints on $\mathcal{P}^{(2)}(\tilde{\mathbf{y}}^l)$, $\mathcal{P}^{(4)}(\tilde{\mathbf{y}}^l)$, on the other hand, remain even after the affine transformation. These constraints on $\mathcal{P}^{(2)}(\tilde{\mathbf{y}}^l)$, $\mathcal{P}^{(4)}(\tilde{\mathbf{y}}^l)$ translate in activation space into $2C_l$ fixed constraints that apply to each $\tilde{\mathbf{y}}^l \in \mathbb{R}^{H \times W \times C_l}$ associated with each choice of input $\mathbf{x}$ in the dataset $\mathcal{D}$ [65].

Such constraints on $\tilde{\mathbf{y}}^l$ are symptomatic of an alteration of expressivity. In particular, they entail that some network mappings that can be expressed without normalization cannot be expressed with IN. One such example is the identity mapping [C.1]. Another such example is a network mapping that would provide in channel c through $\tilde{\mathbf{y}}^l_{\alpha,c}$, just before the nonlinearity, a detector of a given concept at position α in the input $\mathbf{x}$. With IN, the lack of variability in instance statistics implies that the mean $\mu_{\mathbf{x},c}(\tilde{\mathbf{y}}^l)$ and standard deviation $\sigma_{\mathbf{x},c}(\tilde{\mathbf{y}}^l)$ of the feature map in channel c are necessarily constant for all $\mathbf{x}$, equal to $\boldsymbol{\beta}_c$ and $\boldsymbol{\gamma}_c$, respectively. This does not allow to express for some inputs $\mathbf{x}$ the presence

[3] In this view, we expect layer-normalized networks to be equally subject to a phenomenon of increasingly imbalanced channels with depth. This phenomenon would be driven mainly by the succession of affine transformations and activation functions, but also by the succession of convolutions [46, 62, 48].

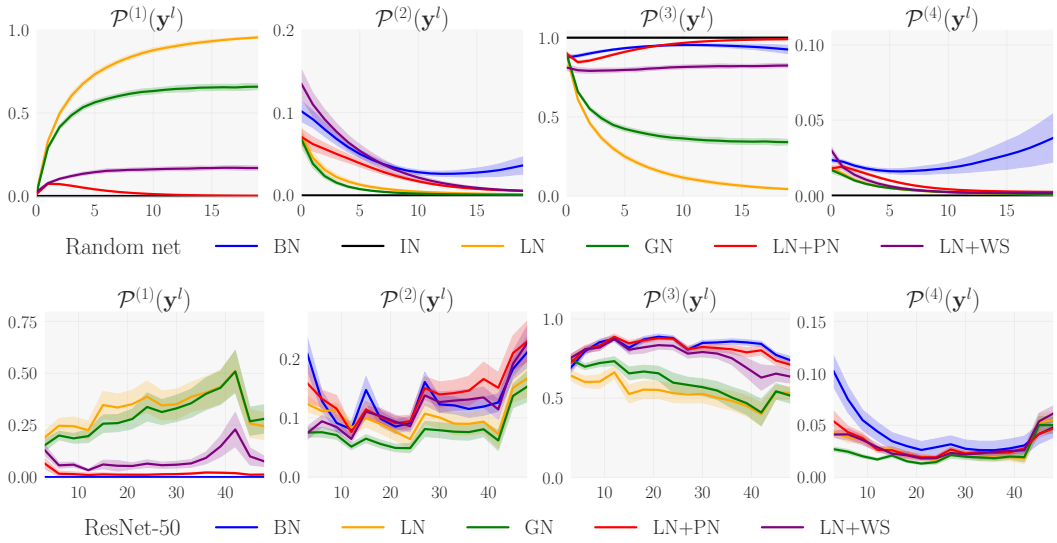

Figure 2: **"Power plots"**. The power decomposition of $\mathbf{y}^l$ is shown as a function of the depth l for BN and different batch-independent norms: IN, LN, GN, LN+PN, LN+WS. Top row: Random net of Definition 1 with $\phi = \text{ReLU}$, widths $C_l = 1024$, kernel sizes $K_l = 3$, convolutions using periodic boundary conditions and components of $\boldsymbol{\beta}^l, \boldsymbol{\gamma}^l$ sampled i.i.d. from $\mathcal{N}(0, 0.2^2)$ and $\mathcal{N}(1, 0.2^2)$, respectively. Bottom row: ResNet-50 (v2) [71] during 100 epochs of training on ImageNet (IN is not shown in this row due to numerical stability issues). Further experimental details are reported in Appendix A.2.

of the concept at some position α: $\mu_{\mathbf{x},c}(\tilde{\mathbf{y}}^l) > 0$, $\sigma_{\mathbf{x},c}(\tilde{\mathbf{y}}^l) > 0$; and for other inputs $\mathbf{x}$ the absence of the concept: $\mu_{\mathbf{x},c}(\tilde{\mathbf{y}}^l) = 0$, $\sigma_{\mathbf{x},c}(\tilde{\mathbf{y}}^l) = 0$.

This latter example is not just anecdotal. Indeed, it is accepted that networks trained on high-level conceptual tasks have their initial layers related to low-level features and their deep layers related to high-level concepts [66, 67]. This view explains the success of IN on the specific task of style transfer with fixed "style" input, IN being then incorporated inside a generator network that only acts on the low-level features of the "content" input [10, 68, 69, 70]. On high-level conceptual tasks, on the other hand, this view hints at a harmful tension between IN's constraints and the requirement of instance variability to express high-level concepts in deep layers. In short, not only is the expressivity altered with IN, but the alteration of the expressivity results in the exclusion of useful network mappings.

Failure modes with GN. Group Normalization is a middle ground between the two extremes of LN ($G = 1$ group) and IN ($G = C_l$ groups at layer l). Networks with GN are consequently affected by both problems of Theorem 1 and Theorem 2, but to a lesser extent than networks with LN and IN. On the one hand, since GN becomes equivalent to a constant scaling when group sizes become large, networks with GN are likely to be subject to channel-wise collapse. On the other hand, since GN can be seen as removing a fraction — with an inverse dependence on the group size — of $\mathcal{P}^{(2)}(\mathbf{y}^l)$, $\mathcal{P}^{(4)}(\mathbf{y}^l)$ in between each layer, networks with GN are likely to lack variability in instance statistics.

The balance stricken by GN between the two failure modes of LN and IN could still be beneficial, which would explain GN's superior performance in practice. It makes sense intuitively that being subject to two failure modes weakly is preferable over being subject to one failure mode strongly.

Experimental validation. The "power plots" of Figure 2 show the power decomposition of $\mathbf{y}^l$ as a function of the depth l in both a random net of Definition 1 and ResNet-50 (v2) trained on ImageNet.[4] Looking at the cases of BN and LN, IN, GN in these power plots, we confirm that: (i) networks with LN, and to a lesser extent GN, are subject to channel-wise collapse as depth increases unlike networks with BN and IN (see Appendix A.2.1 for a precise verification of Theorem 1); (ii) networks with IN, and to a lesser extent GN, lack variability in instance statistics.

[4] To ensure that activation steps are directly preceded by normalization steps (cf Section 5), we always use v2 instantiations of ResNets and instantiations of ResNeXts having the same block reorderings as ResNets v2 [A.1].

5 Proxy Normalization

With the goal of remedying the failure modes of Section 4, we now introduce *our novel technique "Proxy Normalization" (PN)* that is at the core of our novel batch-independent normalization.

PN is incorporated into the neural network by replacing the activation step of Eq. (4) by the following "proxy-normalized" activation step (cf Figure 3 and the practical implementation of Appendix B):

$$\tilde{\mathbf{z}}^l = \text{PN-Act}(\mathbf{y}^l), \qquad \forall \alpha, c: \quad \text{PN-Act}(\mathbf{y}^l)_{\alpha,c} = \frac{\phi\big(\boldsymbol{\gamma}^l_c \mathbf{y}^l_{\alpha,c} + \boldsymbol{\beta}^l_c\big) - \mathbb{E}_{Y^l_c}\big[\phi\big(\boldsymbol{\gamma}^l_c Y^l_c + \boldsymbol{\beta}^l_c\big)\big]}{\sqrt{\text{Var}_{Y^l_c}\big[\phi\big(\boldsymbol{\gamma}^l_c Y^l_c + \boldsymbol{\beta}^l_c\big)\big] + \epsilon}}, \quad (7)$$

where $\epsilon \geq 0$ is a numerical stability constant and Y^l_c is a Gaussian "proxy" variable of mean $\tilde{\boldsymbol{\beta}}^l_c$ and variance $(1 + \tilde{\boldsymbol{\gamma}}^l_c)^2$ depending on additional parameters $\tilde{\boldsymbol{\beta}}^l, \tilde{\boldsymbol{\gamma}}^l \in \mathbb{R}^{C_l}$ of PN. Unless otherwise stated, we let $\tilde{\boldsymbol{\beta}}^l$, $\tilde{\boldsymbol{\gamma}}^l$ be nonzero but still subject to weight decay and thus close to zero. We show in Section A.3.3 that it is also effective to let $\tilde{\boldsymbol{\beta}}^l$, $\tilde{\boldsymbol{\gamma}}^l$ be strictly zero and $Y^l_c \sim \mathcal{N}(0, 1)$.

If one assumes, as hinted by Section 4, that only the affine transformation and the activation function ϕ (i.e. the activation step) play a role in the aggravation at each layer l of the channel-wise collapse and channel imbalance, then PN provides the following guarantee of channel-wise normalization.

Theorem 3 (guarantee of channel-wise normalization in proxy-normalized networks (informal))**.** [F] *Fix a layer $l \in \{1, \ldots, L\}$ and lift any assumptions on ϕ and $\mathbf{x}$'s distribution. Further suppose that the neural network implements Eq. (2), (3), (7) at every layer up to depth L, with $\epsilon = 0$ and Eq. (3), (7) having nonzero denominators for all layers, inputs and channels.*

Then both $\mathbf{y}^l$ and $\tilde{\mathbf{z}}^l$ at layer l are channel-wise normalized if the following conditions hold:

(i) $\tilde{\mathbf{z}}^{l-1}$ at layer $l-1$ is channel-wise normalized;

(ii) The convolution and normalization steps at layer l do not cause any aggravation of channel-wise collapse and channel imbalance;

(iii) $\mathbf{y}^l$ at layer l is channel-wise Gaussian and PN's additional parameters $\tilde{\boldsymbol{\beta}}^l$, $\tilde{\boldsymbol{\gamma}}^l$ are zero.

Our batch-independent approach: LN+PN or GN+PN. At this point, we crucially note that PN: (i) is batch-independent; (ii) does not cause any alteration of expressivity. *This leads us to adopt a batch-independent normalization approach that uses either LN or GN (with few groups) in the normalization step and that replaces the activation step by the proxy-normalized activation step (+PN).* With such a choice of normalization step, we guarantee three of the benefits detailed in Table 1: "scale invariance", "control of activation scale" and "preservation of expressivity". With the proxy-normalized activation step, we finally guarantee the fourth benefit of "avoidance of collapse" without compromising any of the benefits provided by the normalization step.

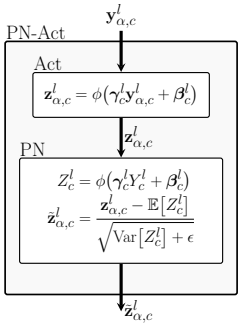

Figure 3: **Plugging PN**. PN is plugged into the activation step Act to yield the proxy-normalized activation step PN-Act.

Experimental validation. We confirm in Figure 2 *that BN's behavior is emulated in a fully batch-independent manner with our approach LN+PN or GN+PN*. Indeed, the power plots of networks with LN+PN resemble the power plots of networks with BN. As wanted, PN remedies LN's failure mode without incurring IN's failure mode.

Approach strength 1: Normalizing beyond initialization. Our batch-independent approach maintains channel-wise normalization throughout training (cf Figure 2). In contrast, many alternative approaches, either explicitly or implicitly, focus on initialization [21, 24, 17, 18, 49, 20, 72]. Centered Weight Normalization [17], PreLayerNorm [49] or WS [18, 20, 72] notably rely on the implicit assumption that different channels have the same channel-wise means after the activation function ϕ. While valid in networks at initialization with $\boldsymbol{\beta}^l = 0$ and $\boldsymbol{\gamma}^l = 1$ (cf Section A.2.2), this assumption becomes less valid as the affine transformation starts deviating from the identity. We see in Figure 2 that networks with LN+WS are indeed less effective in maintaining channel-wise normalization, both in networks with random $\boldsymbol{\beta}^l$, $\boldsymbol{\gamma}^l$ (top row) and in networks considered throughout training (bottom row). Such a coarser channel-wise normalization might in turn lead to a less effective use of model capacity and a degradation of the conditioning of the loss landscape [61, 63].

Approach Strength 2: Wide applicability. Our batch-independent approach matches BN with consistency across choices of model architectures, model sizes and activation functions (cf Section 6). Its only restriction, namely that activation steps should be immediately preceded by normalization steps for $\mathbf{y}^l$ and its associated proxy Y^l to be at the same scale, has easy workarounds. Applicability restrictions might be more serious with alternative approaches: (i) alternative approaches involving the tracking of activation statistics [21, 22, 23] might be nontrivial to apply to residual networks [73, 74]; (ii) approaches involving a normalization in weight space [16, 17, 18, 19, 20] might be ill-suited to architectures with less "dense" kernels such as EfficientNets [20]; (iii) approaches involving a change of activation function [24, 5, 15] might precisely restrict the choice of activation function.

Approach strength 3: Ease of implementation. Our approach is straightforward to implement when starting from a batch-normalized network. It simply requires replacing (i) all BN steps with LN or GN steps, and (ii) all activation steps with proxy-normalized activation steps. The proxy-normalized activation steps themselves are easily implemented (cf our implementation of Appendix B).

6 Results

We finally assess the practical performance of our batch-independent normalization approach. While we focus on ImageNet [75] in the main text of this paper, we report additional results on CIFAR [76] in Appendix A. In Appendix A, we also provide all the details on our experimental setup.

Choices of regularization and batch size. As mentioned in Section 2, we perform all our experiments with different degrees of regularization to disentangle normalization's effects from regularization's effects. We detail all our choices of regularization in Appendices A.1 and A.3.4.

In terms of batch size, we set: (i) the "global batch size" in between weight updates to the same value independent of the choice of norm; (ii) the "normalization batch size" to a near-optimal value with BN. These choices enable us to be conservative when concluding on a potential advantage of our approach over BN at small batch size. Indeed, while the performance of batch-independent approaches would remain the same or slightly improve at small batch size [77, 64], the performance of BN would eventually become degraded due to a regularization that eventually becomes excessive [2, 3, 4, 5, 6, 7].

Effect of adding PN. Without further ado, we start by analyzing the effect of adding PN on top of various norms in ResNet-50 (RN50). As visible in Table 2, PN is mostly beneficial when added on top of LN or GN with a small number of groups G. The consequence is that the optimal G shifts to lower values in GN+PN compared to GN. This confirms the view that PN's benefit lies in addressing LN's failure mode.

It is also visible in Table 2 that PN does not provide noticeable benefits to BN. This confirms again the view that PN's benefit lies in addressing the problem, not present with BN, of channel-wise collapse. Importantly, since PN does not entail effects other than normalization that could artificially boost the performance, *GN+PN can be compared in a fair way to BN when assessing the effectiveness of normalization*.

Table 2: **Effect of adding PN**. ResNet-50 is trained on ImageNet with BN and LN, GN, GN+WS with G groups, either without or with PN added on top (plain vs. PN). Results are formatted as X / Y with X, Y the validation accuracies (%) without and with extra regularization, respectively.

		RN50	
	G	plain	+PN
BN		**76.3** / 75.8	76.2 / 76.0
LN	1	74.5 / 74.6	75.9 / 76.5
GN	8	75.4 / 75.4	76.3 / **76.7**
GN	32	75.4 / 75.3	75.8 / 76.1
GN+WS	8	76.6 / 76.7	76.8 / **77.1**

This is unlike WS which has been shown to improve BN's performance [18]. In our results of Table 2, the high performance of GN+WS without extra regularization and the fact that PN still provides benefits to GN+WS suggests that: (i) on top of its normalization benefits, WS induces a form of regularization; (ii) GN+WS is still not fully optimal in terms of normalization.

GN+PN vs. BN. Next, we turn to comparing the performance of our approach GN+PN to that of BN across a broad range of models trained on ImageNet. As visible in Figure 4 and Tables 3, 4, GN+PN outperforms BN in ResNet-50 (RN50) and ResNet-101 (RN101) [71], matches BN in ResNeXt-50 (RNX50) and ResNeXt-101 (RNX101) [78], and matches BN in EfficientNet-B0 (EN-B0) and EfficientNet-B2 (EN-B2), both in the original variant with depthwise convolutions and expansion

ratio of 6 [52] and in an approximately parameter-preserving variant with group convolutions of group size 16 and expansion ratio of 4 [79] (cf Section A.1). *In short, our batch-independent normalization approach GN+PN matches BN not only in behavior but also in performance.*

With regard to matching BN's performance with alternative norms, various positive results have been reported in ResNets and ResNeXts [8, 7, 12, 13, 15, 20, 36] but only a limited number in EfficientNets [15]. In EfficientNets, we are notably not aware of any other work outlining that BN's performance can be matched using a batch-independent approach. As a confirmation, we assess the performance of various batch-independent approaches: GN [3], GN+WS [18], Evo-S0 [15], FRN+TLU [80, 5]. Unlike GN+PN, none of these approaches is found in Table 4 to match BN with consistency.

Normalization and regularization. Our results suggest that while an efficient normalization is not sufficient in itself to achieve good performance on ImageNet, it is still a necessary condition, together with regularization. In our results, it is always with extra regularization that GN+PN yields the most benefits. Importantly, the fact that GN+PN consistently leads to large improvements in training accuracy [A.3.2] suggests that *additional benefits would be obtained on larger datasets without the requirement of relying on regularization* [81, 72].

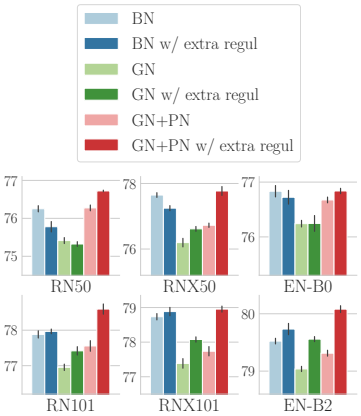

Figure 4: **BN vs. GN, GN+PN**. Validation accuracies (%) of ResNets, ResNeXts and EfficientNets trained on ImageNet with BN and GN, GN+PN, without and with extra regularization. EfficientNets are considered in the variant with group convolutions [79].

Table 3: **BN vs. GN, GN+PN**. ResNets and ResNeXts are trained on ImageNet with BN and GN, GN+PN. Results are formatted as in Table 2.

	RN50	RN101	RNX50	RNX101
BN	**76.3** / 75.8	77.9 / **78.0**	**77.6** / 77.2	78.7 / **78.9**
GN	75.4 / 75.3	77.0 / 77.4	76.2 / 76.6	77.4 / 78.1
GN+PN	76.3 / **76.7**	77.6 / **78.6**	76.7 / **77.8**	77.7 / **79.0**

Table 4: **BN vs. batch-independent approaches**. EfficientNets are trained on ImageNet with BN and different batch-independent approaches. Results are formatted as in Table 2.

	depthwise convs [52]		group convs [79]	
	EN-B0	EN-B2	EN-B0	EN-B2
BN	76.9 / **77.2**	79.4 / **80.0**	**76.8** / 76.7	79.5 / **79.7**
GN	76.2 / 76.2	78.9 / 79.4	76.2 / 76.2	79.0 / 79.6
GN+PN	76.8 / **77.0**	79.3 / **80.0**	76.7 / **76.8**	79.3 / **80.1**
Evo-S0	75.8 / 75.8	78.5 / 78.7	76.2 / 76.5	78.9 / 79.6
GN+WS	74.2 / 74.1	77.8 / 77.8	76.2 / 76.3	79.2 / 79.4
FRN+TLU	75.7 / 75.7	78.4 / 78.8	74.9 / 75.1	78.2 / 78.6

7 Summary and broader impact

We introduced a novel framework to finely characterize the various neural network properties affected by the choice of normalization. Using this framework, we showed that while BN's beneficial properties are not retained when solely using the prototypical batch-independent normalization techniques, they are retained when combining some of these techniques with the technique hereby introduced of Proxy Normalization. We demonstrated on an extensive set of experiments that our batch-independent normalization approach consistently matches BN in both behavior and performance.

The main implications of this work could stem from the unlocked possibility to retain the normalization benefits of BN while removing batch dependence. Firstly, our approach could be used to retain BN's normalization benefits while alleviating the burden of large activation memory stemming from BN's requirement of large enough batch sizes. This is expected to be important in memory-intensive applications such as object detection or image segmentation, but also when using A.I. accelerators that leverage local memory to provide extra acceleration and energy savings in exchange for tighter memory constraints. Secondly, our approach could be used to retain BN's normalization benefits while avoiding BN's regularization when the latter is detrimental. As discussed in Section 6, this is expected to be important in the context of large datasets, that will likely be prevalent in the future.

Acknowledgments and Disclosure of Funding

We are thankful to Simon Knowles, Luke Hudlass-Galley, Luke Prince, Alexandros Koliousis, Anastasia Dietrich and the wider research team at Graphcore for the useful discussions and feedbacks. We are also thankful to the anonymous reviewers for their insightful comments that helped improve the paper.

References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning, ICML 2015*, pages 448–456, 2015.

[2] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 1945–1953, 2017.

[3] Yuxin Wu and Kaiming He. Group normalization. In *Computer Vision - ECCV 2018 - 15th European Conference, Proceedings, Part XIII*, pages 3–19, 2018.

[4] Chris Ying, Sameer Kumar, Dehao Chen, Tao Wang, and Youlong Cheng. Image classification at supercomputer scale. *CoRR*, abs/1811.06992, 2018.

[5] Saurabh Singh and Shankar Krishnan. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pages 11234–11243, 2020.

[6] Cecilia Summers and Michael J. Dinneen. Four things everyone should know to improve batch normalization. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[7] Junjie Yan, Ruosi Wan, Xiangyu Zhang, Wei Zhang, Yichen Wei, and Jian Sun. Towards stabilizing batch statistics in backward propagation of batch normalization. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[8] Vitaliy Chiley, Ilya Sharapov, Atli Kosson, Urs Köster, Ryan Reece, Sofia Samaniego de la Fuente, Vishal Subbiah, and Michael James. Online normalization for training neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 8431–8441, 2019.

[9] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[10] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[11] Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H. Sinz, and Richard S. Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, 2017.

[12] Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[13] Ping Luo, Zhanglin Peng, Wenqi Shao, Ruimao Zhang, Jiamin Ren, and Lingyun Wu. Differentiable dynamic normalization for learning deep representation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 4203–4211, 2019.

[14] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 12360–12371, 2019.

[15] Hanxiao Liu, Andy Brock, Karen Simonyan, and Quoc Le. Evolving normalization-activation layers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[16] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, page 901, 2016.

[17] Lei Huang, Xianglong Liu, Yang Liu, Bo Lang, and Dacheng Tao. Centered weight normalization in accelerating training of deep neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017*, pages 2822–2830, 2017.

[18] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan L. Yuille. Weight standardization. *CoRR*, abs/1903.10520, 2019.

[19] Brendan Ruff, Taylor Beck, and Joscha Bach. Mean shift rejection: Training deep neural networks without minibatch statistics or normalization. *CoRR*, abs/1911.13173, 2019.

[20] Andrew Brock, Soham De, and Samuel L. Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[21] Devansh Arpit, Yingbo Zhou, Bhargava Urala Kota, and Venu Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1168–1176, 2016.

[22] César Laurent, Nicolas Ballas, and Pascal Vincent. Recurrent normalization propagation. In *5th International Conference on Learning Representations, ICLR 2017, Workshop Track Proceedings*, 2017.

[23] Alexander Shekhovtsov and Boris Flach. Normalization of neural networks using analytic variance propagation. *CoRR*, abs/1803.10560, 2018.

[24] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 971–980, 2017.

[25] Twan van Laarhoven. L2 regularization versus batch and weight normalization. *CoRR*, abs/1706.05350, 2017.

[26] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[27] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 2164–2174, 2018.

[28] Zhiyuan Li and Sanjeev Arora. An exponential learning rate schedule for deep learning. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[29] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger B. Grosse. Three mechanisms of weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[30] Ruosi Wan, Zhanxing Zhu, Xiangyu Zhang, and Jian Sun. Spherical motion dynamics of deep neural networks with batch normalization and weight decay. *CoRR*, abs/2006.08419, 2020.

[31] Minhyung Cho and Jaehyung Lee. Riemannian approach to batch normalization. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 5225–5235, 2017.

[32] Jonas Moritz Kohler, Hadi Daneshmand, Aurélien Lucchi, Thomas Hofmann, Ming Zhou, and Klaus Neymeyr. 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, AISTATS 2019*, volume 89 of *Proceedings of Machine Learning Research*, pages 806–815, 2019.

[33] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 569–579, 2018.

[34] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[35] Soham De and Samuel L. Smith. Batch normalization biases residual blocks towards the identity function in deep networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[36] Jie Shao, Kai Hu, Changhu Wang, Xiangyang Xue, and Bhiksha Raj. Is normalization indispensable for training deep neural network? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[37] Ekdeep Singh Lubana, Robert P. Dick, and Hidenori Tanaka. Beyond batchnorm: Towards a general understanding of normalization in deep learning. *CoRR*, abs/2106.05956, 2021.

[38] Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pages 550–558, 2016.

[39] George Philipp, Dawn Song, and Jaime G. Carbonell. Gradients explode - deep networks are shallow - resnet explained. In *6th International Conference on Learning Representations, ICLR 2018, Workshop Track Proceedings*, 2018.

[40] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. A mean field theory of batch normalization. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[41] Antoine Labatie. Characterizing well-behaved vs. pathological deep neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 3611–3621, 2019.

[42] Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W. Taylor. Batch normalization is a cause of adversarial vulnerability. *CoRR*, abs/1905.02161, 2019.

[43] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4914–4923, 2018.

[44] Alexander Shekhovtsov and Boris Flach. Stochastic normalizations as bayesian learning. In *Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, Revised Selected Papers, Part II*, volume 11362 of *Lecture Notes in Computer Science*, pages 463–479, 2018.

[45] Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[46] Johan Bjorck, Carla P. Gomes, Bart Selman, and Kilian Q. Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 7705–7716, 2018.

[47] Arthur Jacot, Franck Gabriel, and Clément Hongler. Freeze and chaos for dnns: an ntk view of batch normalization, checkerboard and boundary effects. *CoRR*, abs/1907.05715, 2019.

[48] Hadi Daneshmand, Jonas Moritz Kohler, Francis R. Bach, Thomas Hofmann, and Aurélien Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[49] Vinay Rao and Jascha Sohl-Dickstein. Is batch norm unique? an empirical investigation and prescription to emulate the best properties of common normalizers without batch dependence. *CoRR*, abs/2010.10687, 2020.

[50] Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 2596–2604, 2019.

[51] Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 359–368, 2019.

[52] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114, 2019.

[53] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. Adversarial examples improve image recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pages 816–825, 2020.

[54] Cihang Xie and Alan L. Yuille. Intriguing properties of adversarial training at scale. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[55] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*, 2018.

[56] Harm de Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C. Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 6594–6604, 2017.

[57] Lucas Deecke, Iain Murray, and Hakan Bilen. Mode normalization. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[58] Ximei Wang, Ying Jin, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. Transferable normalization: Towards improving transferability of deep neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 1951–1961, 2019.

[59] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, pages 7354–7362, 2019.

[60] Yuxin Wu and Justin Johnson. Rethinking "batch" in batchnorm. *CoRR*, abs/2105.07576, 2021.

[61] David Page. How to train your resnet 7: Batch norm, 2019.

[62] Jeffrey Pennington and Pratik Worah. Nonlinear random matrix theory for deep learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 2637–2646, 2017.

[63] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 2232–2241, 2019.

[64] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018.

[65] Lei Huang, Yi Zhou, Li Liu, Fan Zhu, and Ling Shao. Group whitening: Balancing learning efficiency and representational capacity. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*, pages 9512–9521, 2021.

[66] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1349–1357, 2016.

[67] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pages 2414–2423, 2016.

[68] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 4105–4113, 2017.

[69] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, 2017.

[70] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision, ICCV 2017*, pages 1510–1519, 2017.

[71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision - ECCV 2016 - 14th European Conference*, 2016.

[72] Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pages 1059–1071, 2021.

[73] Wenling Shang, Justin Chiu, and Kihyuk Sohn. Exploring normalization in deep residual networks with concatenated rectified linear units. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1509–1516, 2017.

[74] Igor Gitman and Boris Ginsburg. Comparison of batch normalization and weight normalization algorithms for the large-scale image classification. *CoRR*, abs/1709.08145, 2017.

[75] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, pages 248–255, 2009.

[76] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[77] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017.

[78] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[79] Dominic Masters, Antoine Labatie, Zach Eaton-Rosen, and Carlo Luschi. Making efficientnet more efficient: Exploring batch-independent normalization, group convolutions and reduced resolution training. *CoRR*, abs/2106.03640, 2021.

[80] Sitao Xiang and Hao Li. On the effect of batch normalization and weight normalization in generative adversarial networks. *CoRR*, abs/1704.03971, 2017.

[81] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *Computer Vision - ECCV 2020 - 16th European Conference*, 2020.

[82] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pages 2818–2826, 2016.

[83] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[84] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *Computer Vision - ECCV 2016 - 14th European Conference, Proceedings, Part IV*, volume 9908, pages 646–661, 2016.

[85] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*, 2018.

[86] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019*, pages 6022–6031, 2019.

[87] Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, pages 113–123, 2019.

[88] Jasmine Collins, Johannes Ballé, and Jonathon Shlens. Accelerating training of deep neural networks with a standardization loss. *CoRR*, abs/1903.00925, 2019.

[89] Yann Dauphin and Ekin Dogus Cubuk. Deconstructing the regularization of batchnorm. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[90] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 2488–2498, 2018.

[91] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We believe the paper's title and abstract are consistent with the theoretical results of Sections 4 and 5 and the experimental results of Sections 5 and 6.
   (b) Did you describe the limitations of your work? [Yes] The scope and assumptions of our theoretical results are provided in Sections 3 and 4. The limitation stemming from PN's requirement that each activation step should be directly preceded by a normalization step is stated in Section 4.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work directly impacts the advancement of the field of computer vision. Its negative impacts could come only indirectly from the negative impacts of this field in general. We thought that this observation was not needed in the main text, as it is already implied.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The full assumptions of our theoretical results are provided in Sections 3 and 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] Complete proofs of all our theoretical results are provided in Appendix C, D, E and F.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The experimental results can be reproduced based on Appendix A and B.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All the experimental details are reported in Appendix A
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The error bars associated with each of our experiments are reported in Appendix A.3.1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] Our experiments with batch-independent norms were run on Graphcore's MK1 IPUs and MK2 IPUs. Our experiments with Batch Normalization were run on TPUs, GPUs and Graphcore's MK2 IPUs. We refer the reader to [79] for throughput numbers on Graphcore's MK2 IPUs.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] ImageNet's images are owned by copyright holders. We are unsure about the existence, and potentially the type, of licenses of CIFAR-10 and CIFAR-100.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]