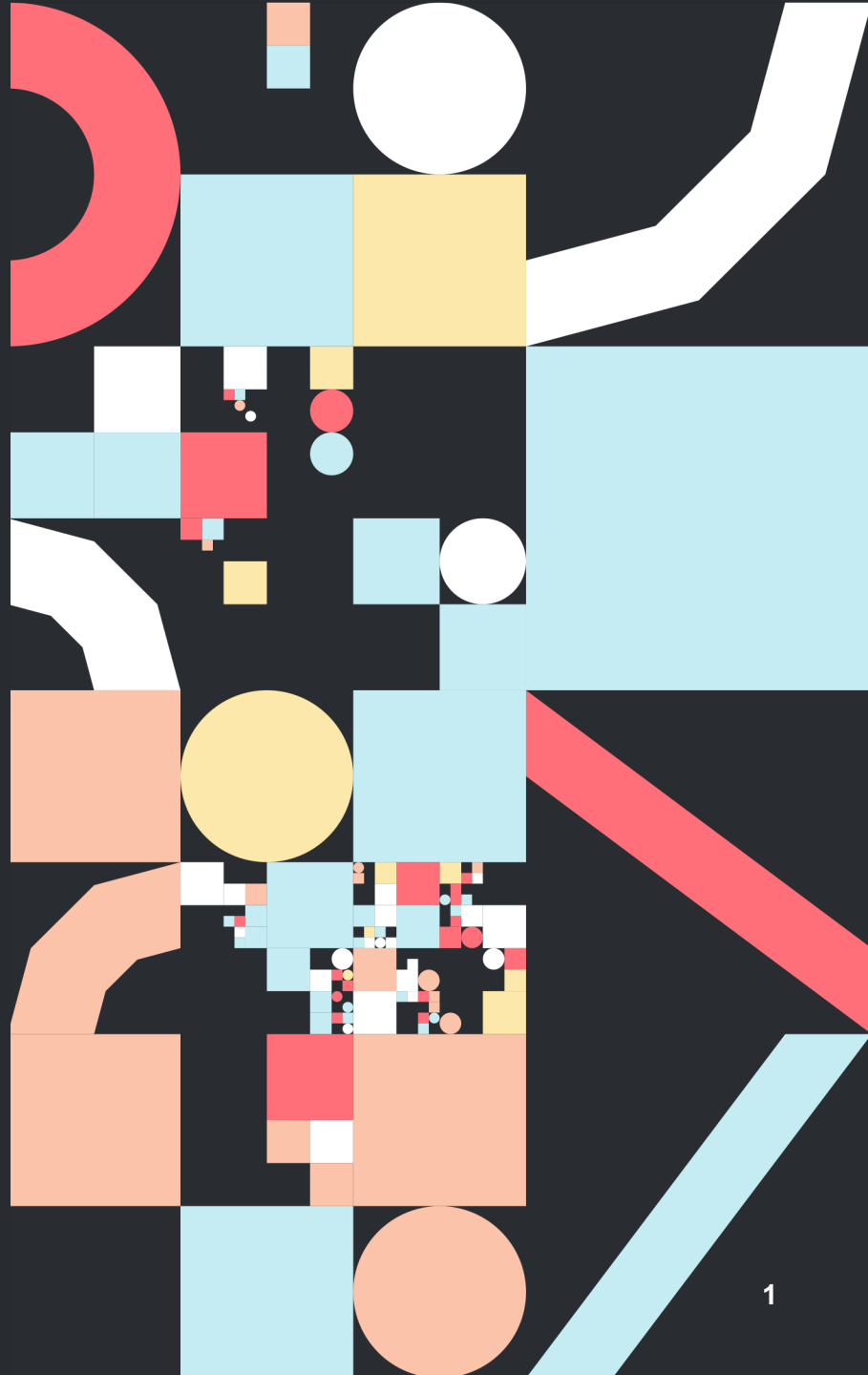
Removing Batch Dependence in CNNs by Proxy-Normalizing Activations

Antoine Labatie — October 2021

GRAPHCORE



OUTLINE

1. Batch Norm
2. Batch-independent alternatives to Batch Norm
3. Proxy Norm
4. Results with Proxy Norm
5. Conclusion

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

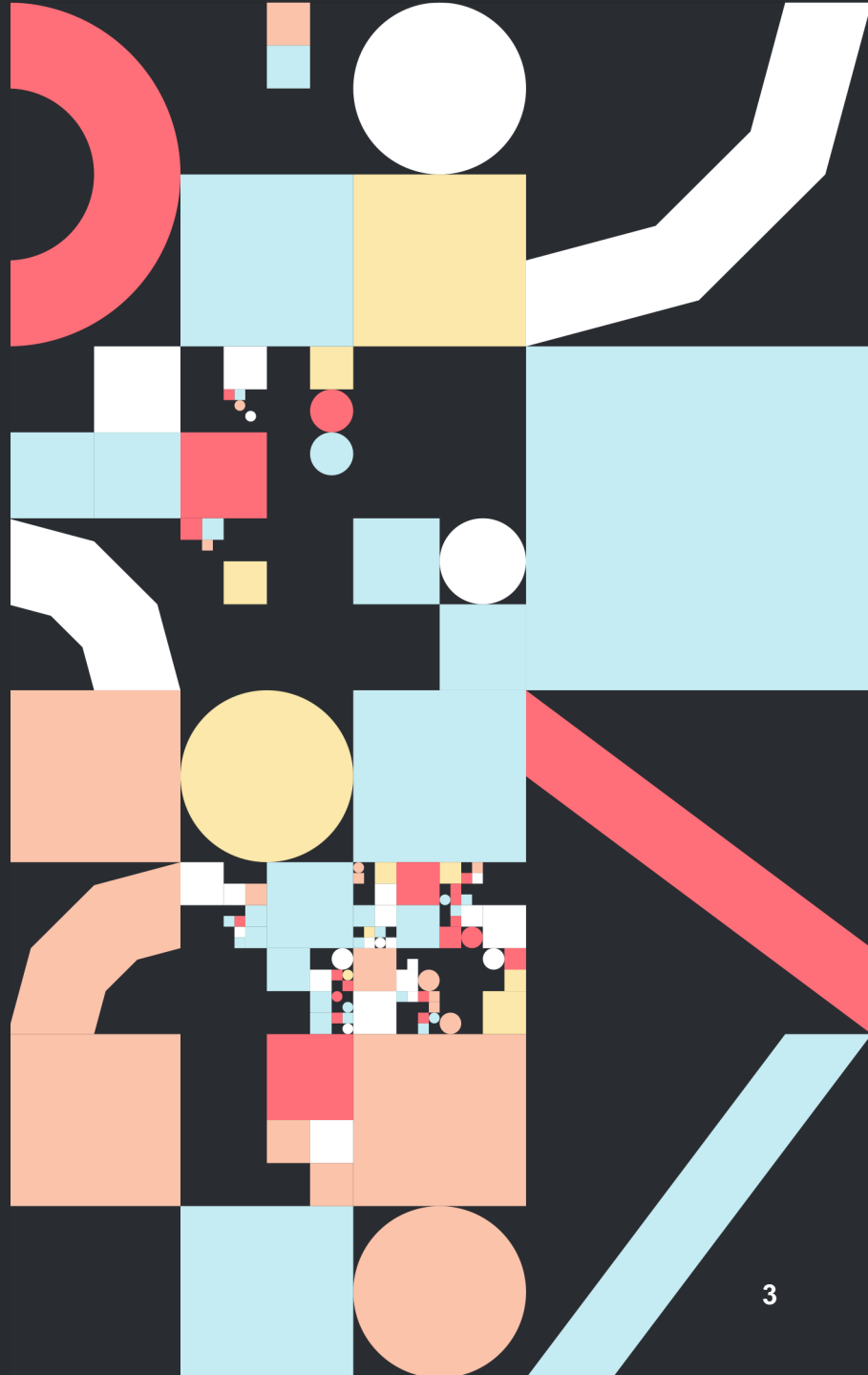
Antoine Labatie **Dominic Masters** **Zach Eaton-Rosen** **Carlo Luschi**
Graphcore Research, UK
{antoinel, dominicm, zacher, carlo}@graphcore.ai

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.

To appear in NeurIPS 2021

1. BATCH NORM



GRAPHCORE

BATCH NORM

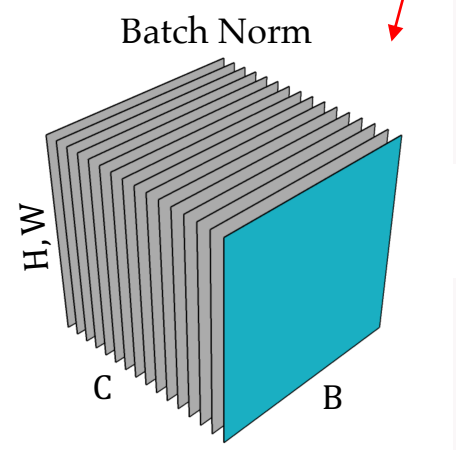
1. Batch Norm normalizes the pre-activations $\boldsymbol{X}$ of shape (B, H, W, C) by:
   - Subtracting the **mean** $\hat{\mu}_c(\boldsymbol{X})$ over (B, H, W)
   - Dividing by the **standard deviation** $\hat{\sigma}_c(\boldsymbol{X})$ over (B, H, W)
   - The **mini-batch** statistics $\hat{\mu}_c(\boldsymbol{X}), \hat{\sigma}_c(\boldsymbol{X})$ are used to approximate the **full-batch** statistics $\mu_c(\boldsymbol{X}), \sigma_c(\boldsymbol{X})$

$$\boldsymbol{Y}_{b,h,w,c} = \frac{\boldsymbol{X}_{b,h,w,c} - \hat{\mu}_c(\boldsymbol{X})}{\hat{\sigma}_c(\boldsymbol{X})} \approx \frac{\boldsymbol{X}_{b,h,w,c} - \mu_c(\boldsymbol{X})}{\sigma_c(\boldsymbol{X})}$$

2. Batch Norm reintroduces the 2 lost degrees of freedom with the scale and shift parameters β_c, γ_c before the nonlinearity ϕ:

$$\boldsymbol{Z}_{b,h,w,c} = \phi(\gamma_c \boldsymbol{Y}_{b,h,w,c} + \beta_c)$$

Figure adapted from Dimitriou et al. 2020



✓ 1ST BENEFIT OF BATCH NORM: Y IS CHANNEL-WISE NORMALIZED

Y is **channel-wise normalized**, i.e.

- $\mu_c(Y) = 0$
- $\sigma_c(Y) = 1$

That means that:

- The network effectively uses its whole **width**
- The network effectively uses its whole **depth**
- In sum, the network effectively uses its whole **capacity**

✓ EFFECTIVE USE OF WIDTH

Since $\boldsymbol{Y}$ is channel-wise normalized:

- All channels c in $\boldsymbol{Y}$ have the same square means:

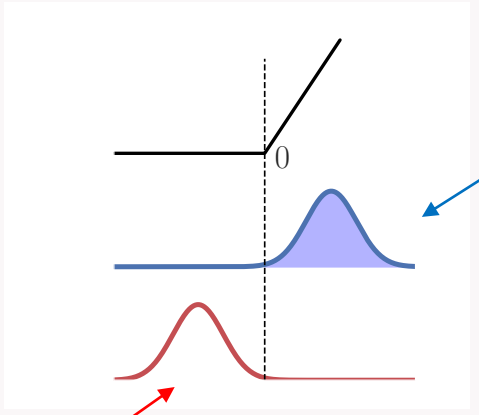
$$\mathbb{E}_{b,h,w}\left[\boldsymbol{Y}_{b,h,w,c}^2\right] = \mu_c(\boldsymbol{Y})^2 + \sigma_c\,(\boldsymbol{Y})^2 = 1$$

- The fact that $\mathbb{E}_{b,h,w}\left[\boldsymbol{Y}_{b,h,w,c}^2\right] = \mathbb{E}_{b,h,w}\left[\boldsymbol{Y}_{b,h,w,c'}^2\right]$ for all c, c' implies that the network effectively uses its **whole width.**

✓ EFFECTIVE USE OF DEPTH

Since $\boldsymbol{Y}$ is channel-wise normalized:

- $\gamma\boldsymbol{Y} + \beta$ is “not too far” from channel-wise normalized
- ϕ “acts” on input distributions “not too far” from channel-wise normalized
- ϕ is effectively channel-wise nonlinear with respect to its input



Input distribution with positive channel-wise mean

Input distribution with negative channel-wise mean

✓ 2ND BENEFIT OF BATCH NORM: PRESERVATION OF EXPRESSIVITY

If mini-batch statistics are good approximations of full-batch statistics:

- $\gamma \boldsymbol{Y} + \beta$ can be made close to $\boldsymbol{X}$ by choosing $\beta_c = \mu_c(\boldsymbol{X})$ and $\gamma_c = \sigma_c(\boldsymbol{X})$:

$$\gamma_c \boldsymbol{Y}_{b,h,w,c} + \beta_c = \sigma_c(\boldsymbol{X}) \boldsymbol{Y}_{b,h,w,c} + \mu_c(\boldsymbol{X}) = \sigma_c(\boldsymbol{X}) \frac{\boldsymbol{X}_{b,h,w,c} - \hat{\mu}_c(\boldsymbol{X})}{\hat{\sigma}_c(\boldsymbol{X})} + \mu_c(\boldsymbol{X}) \approx \boldsymbol{X}_{b,h,w,c}$$

Thus, networks with Batch Norm have approximately the **same expressivity** as unnormalized networks

- This might seem like a trivial property
- But it's hard to retain this property while at the same time guaranteeing that $\boldsymbol{Y}$ is channel-wise normalized

✗ PROBLEM OF BATCH NORM: BATCH DEPENDENCE

The batch dependence is introduced when normalizing with mini-batch statistics instead of full-batch statistics:

$$\boldsymbol{Y}_{b,h,w,c} = \frac{\boldsymbol{X}_{b,h,w,c} - \hat{\mu}_c(\boldsymbol{X})}{\hat{\sigma}_c(\boldsymbol{X})}$$

The random choice of samples in each mini-batch leads to a stochasticity of $\boldsymbol{Y}$:

- This stochasticity translates into a **regularization** of BN
- The regularization strength can only be decreased by increasing the ”normalization” mini-batch size B
- On ImageNet, optimal performance requires $B \geq 32$
- On larger datasets, optimal performance would require even larger B

When the ”compute” mini-batch is smaller than the optimal ”normalization” mini-batch, an “**expensive” synchronisation of Batch Norm’s statistics** across several workers is required.

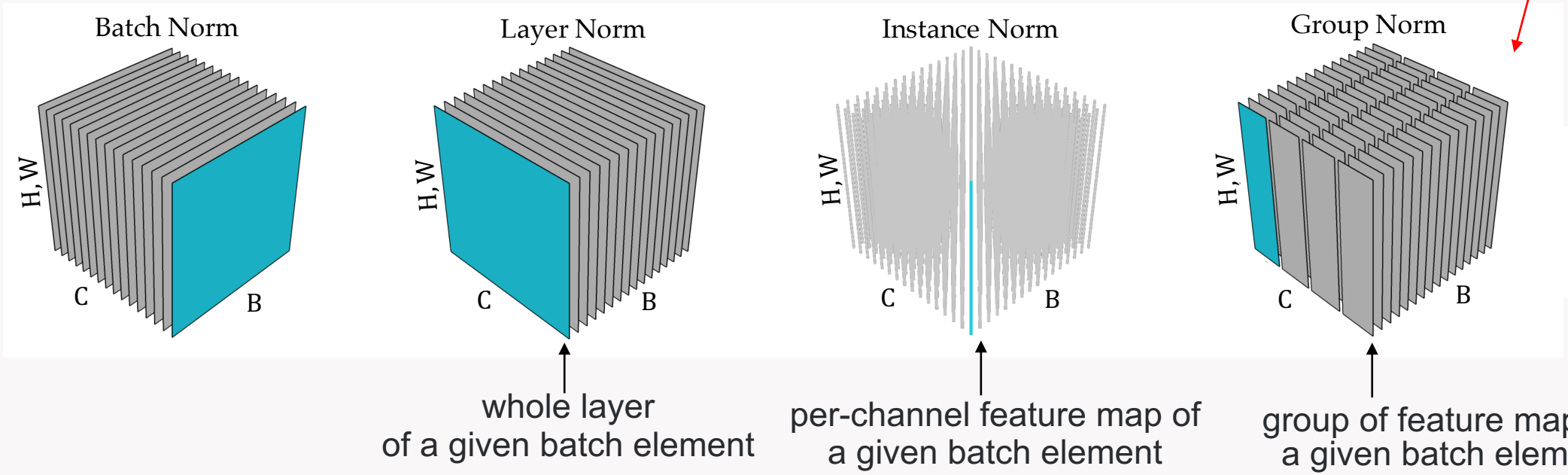
2. BATCH-INDEPENDENT ALTERNATIVES

GRAPHCORE

BATCH-INDEPENDENT ALTERNATIVES

Various batch-independent alternatives to Batch Norm have been proposed

Figure adapted from Dimitriou et al. 2020



✗ PROBLEMS WITH BATCH-INDEPENDENT ALTERNATIVES (I)

It is hard to retain both beneficial properties of Batch Norm simultaneously:

- With Layer Norm: $\boldsymbol{Y}$ is far from channel-wise normalized
- With Instance Norm: expressivity is strongly altered
- Group Norm reaches a trade-off but not a real solution

	Beneficial properties of Batch Norm		Batch independence
	$\mathbf{Y}$ is channel-wise normalized	Expressivity is preserved	
Batch Norm	✓	✓	✗
Layer Norm	✗	✓	✓
Instance Norm	✓	✗	✓
Group Norm	~	~	✓

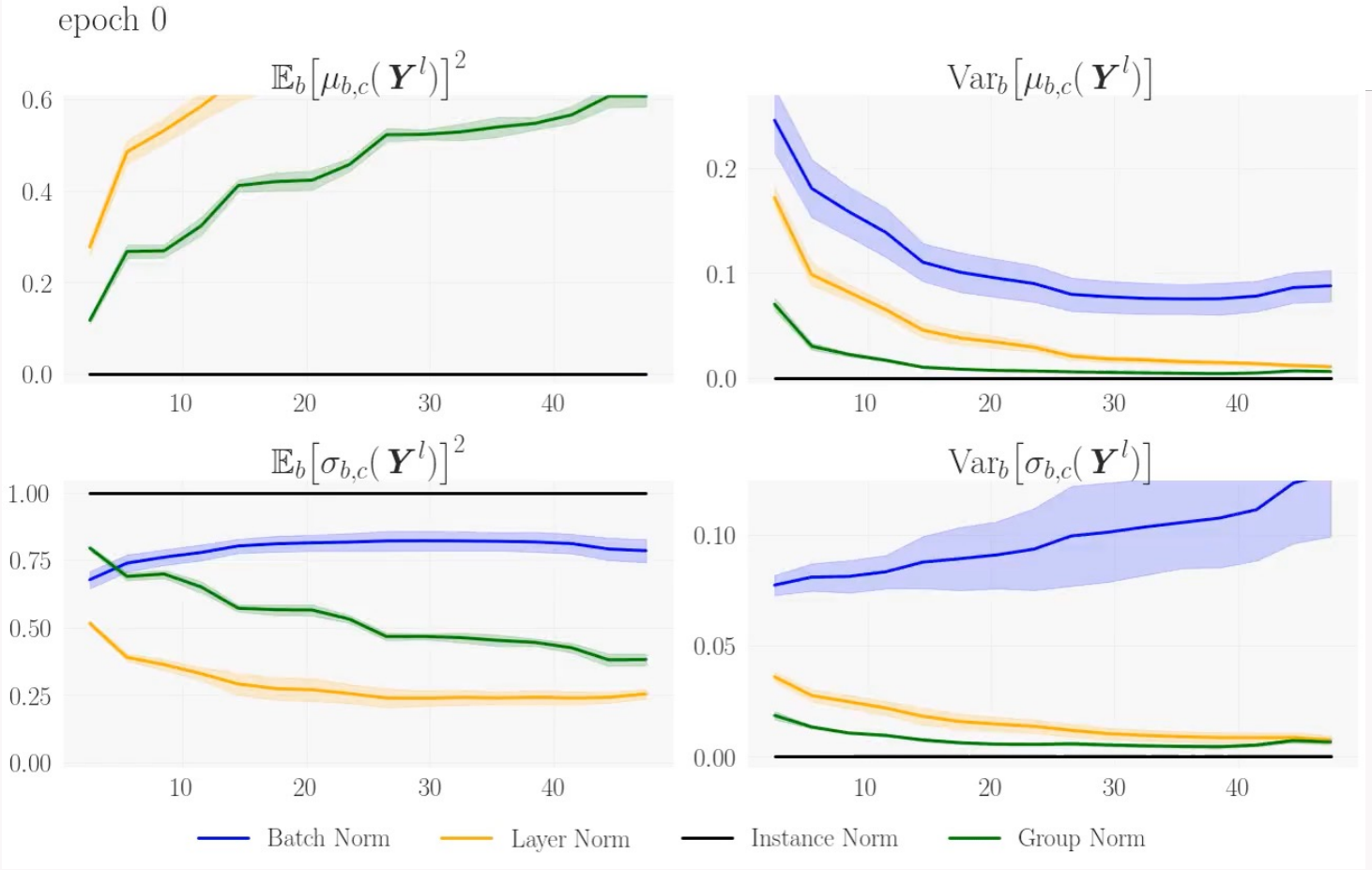
✗ PROBLEMS WITH BATCH-INDEPENDENT ALTERNATIVES (II)

In-house methodology

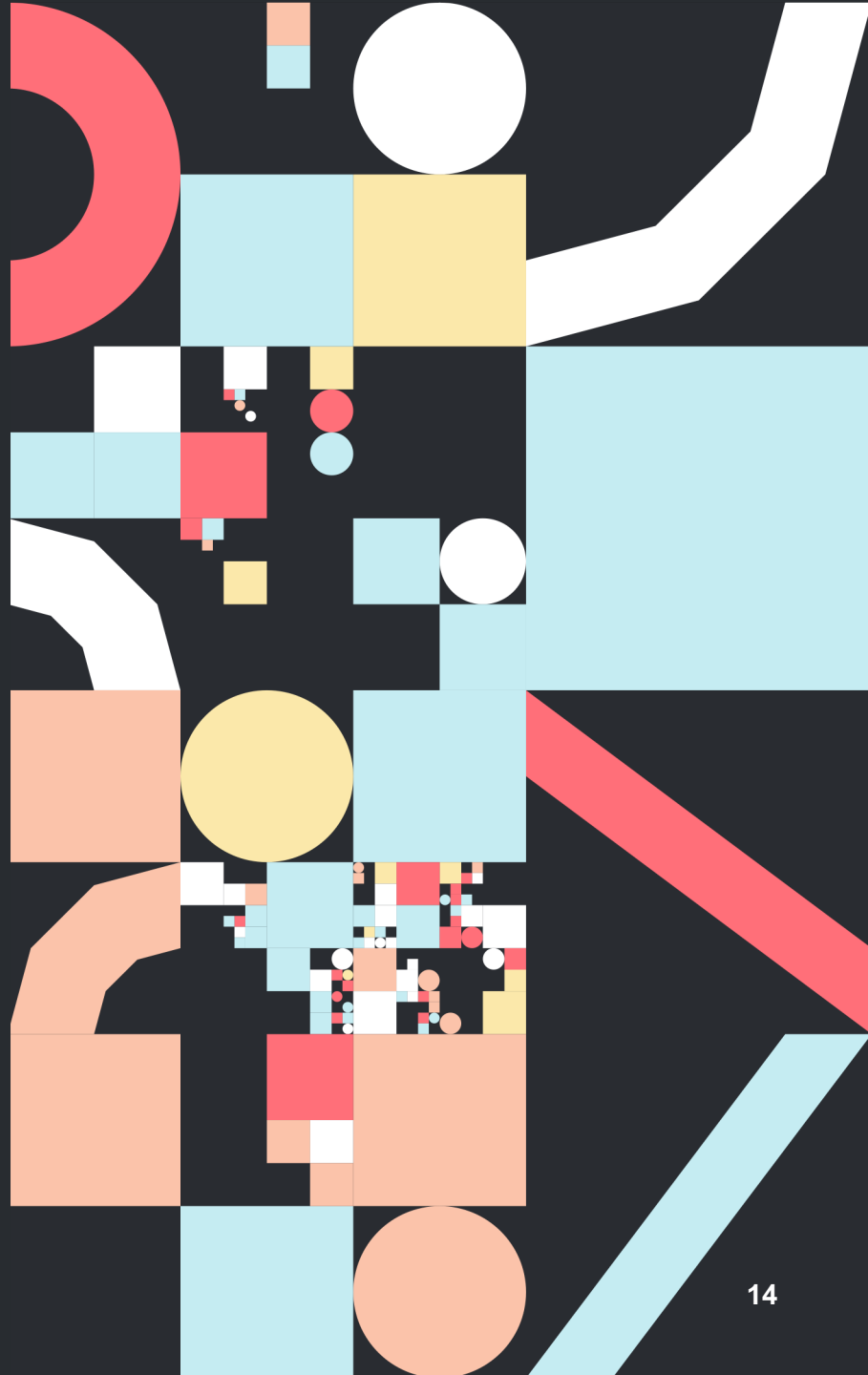
- Look at expectation and variance over batch elements of $\mu_{b,c}(\boldsymbol{Y^l})$ and $\sigma_{b,c}(\boldsymbol{Y^l})$
- The sum of these four terms equals 1

Results

- Layer Norm: 1st term is dominant in deep layers
- Instance Norm: 2nd and 4th terms are constrained to be 0
- Group Norm: middle ground between those 2 issues



3. PROXY NORM



GRAPHCORE

RATIONALE FOR PROXY NORM (I)

Can we fix the issues of batch-independent norms?

Fixing Instance Norm's issue

- Can we restore the variances $\mathrm{Var}_b[\mu_{b,c}(\boldsymbol{Y})]$ and $\mathrm{Var}_b[\sigma_{b,c}(\boldsymbol{Y})]$?
- That seems difficult…

Fixing Layer Norm's issue

- Can we restore $\mu_c(\boldsymbol{Y}) = \mathbb{E}_b\left[\mu_{b,c}(\boldsymbol{Y})\right] \approx 0$ and $\sigma_c(\boldsymbol{Y}) \approx \sigma_{c'}(\boldsymbol{Y})$ for all c, c'?
- It turns out this is feasible!

RATIONALE FOR PROXY NORM (II)

What causes $\mu_c(\boldsymbol{Y}) \not\approx 0$ and $\sigma_c(\boldsymbol{Y}) \not\approx \sigma_{c'}(\boldsymbol{Y})$?

- Convolutions and Layer Norm have little role in this
- The culprits are the affine transform $\boldsymbol{Y} \mapsto \gamma\boldsymbol{Y} + \beta$ and the activation function ϕ

Idea of Proxy Norm: cancel the effect of the affine transform and ϕ

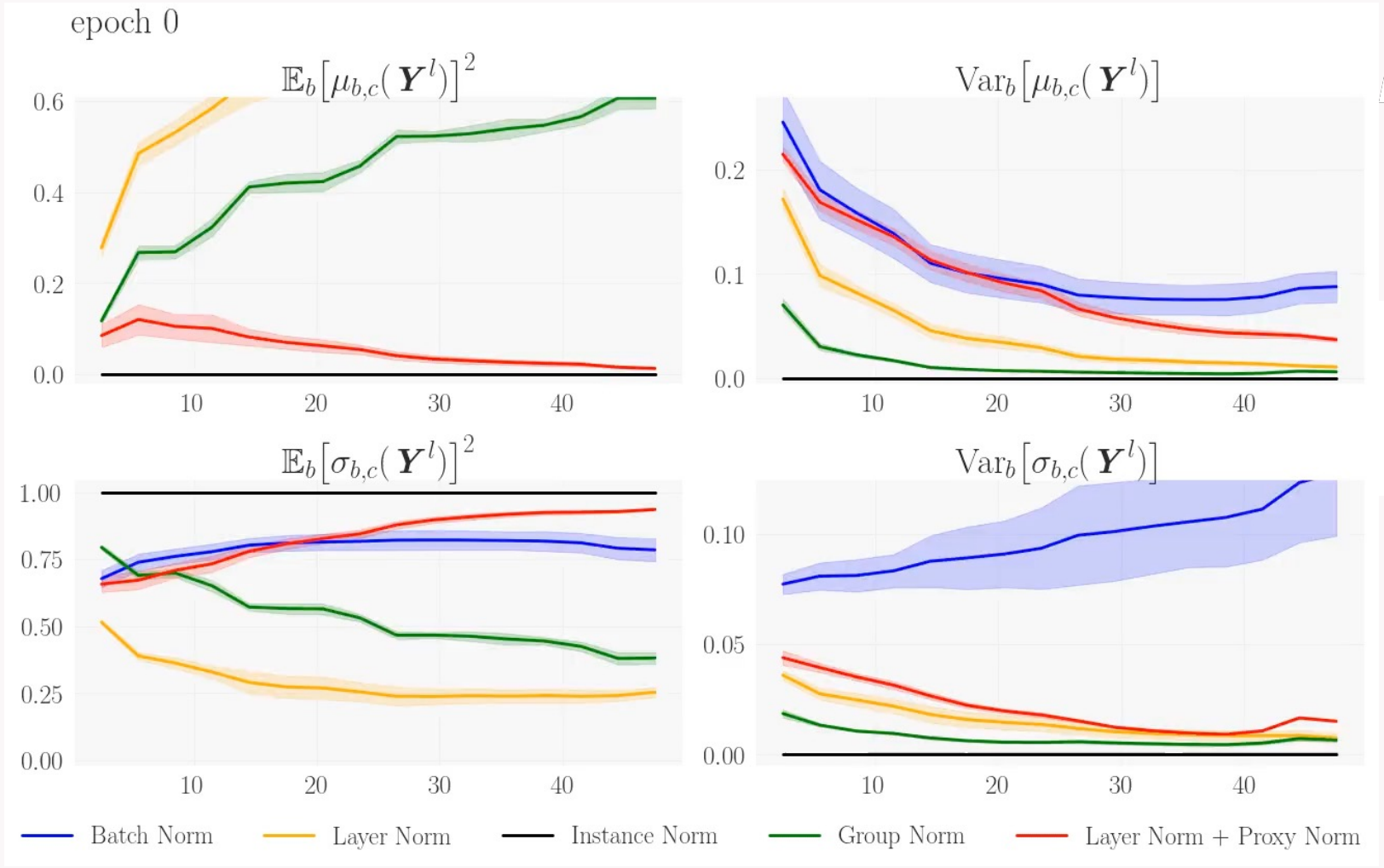
- Assimilate $\boldsymbol{Y}$ to a proxy variable $\widetilde{\boldsymbol{Y}} \sim \mathcal{N}\left(\tilde{\beta}, (1+\tilde{\gamma})^2\right) \approx \mathcal{N}(0,1)$
- Replace the activation by a proxy-normalized activation:

$$\widetilde{\boldsymbol{Z}} = \mathrm{PN{-}Act}(\boldsymbol{Y}) = \frac{\phi(\gamma\boldsymbol{Y}+\beta) - \mathbb{E}_{\widetilde{\boldsymbol{Y}}}[\phi(\gamma\widetilde{\boldsymbol{Y}}+\beta)]}{\sqrt{\mathrm{Var}_{\widetilde{\boldsymbol{Y}}}[\phi(\gamma\widetilde{\boldsymbol{Y}}+\beta)]}}$$

Iterative guarantee of Proxy Norm. If Y is close channel-wise normalized, then:

- The assimilation of $\boldsymbol{Y}$ by $\widetilde{\boldsymbol{Y}}$ is sensible
- $\widetilde{\boldsymbol{Z}}$ after ϕ is close to channel-wise normalized
- $\boldsymbol{Y}$ at the next layer is close to channel-wise normalized

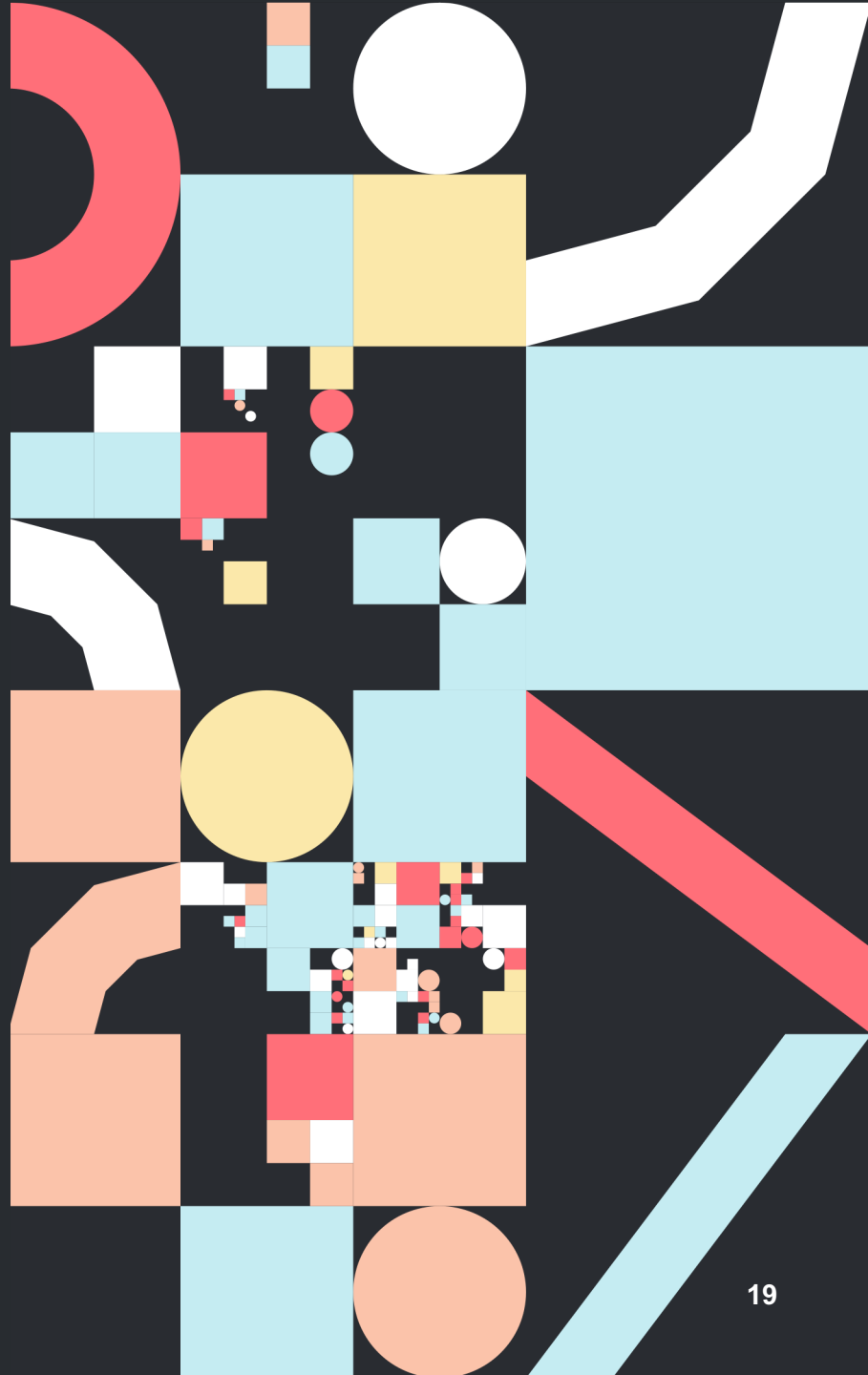
✓ BENEFICIAL PROPERTIES WITH PROXY NORM



SUMMARY

	Batch Norm's beneficial properties		Batch independence
	Y is channel-wise normalized	Expressivity is preserved	
Batch Norm	✓	✓	✗
Layer Norm	✗	✓	✓
Instance Norm	✓	✗	✓
Group Norm	~	~	✓
Layer Norm / Group Norm (w/ few groups) + Proxy Norm	✓	✓	✓

4. RESULTS WITH PROXY NORM



GRAPHCORE

EXPERIMENTAL SETUP

Architectures:

- ResNet / ResNeXt / EfficientNet

Tasks:

- ImageNet / CIFAR-10 / CIFAR-100

Conservative choices of batch size:

- Global batch size chosen independently of norms
- “Normalization” batch size with BN chosen close to the optimum

Regularization:

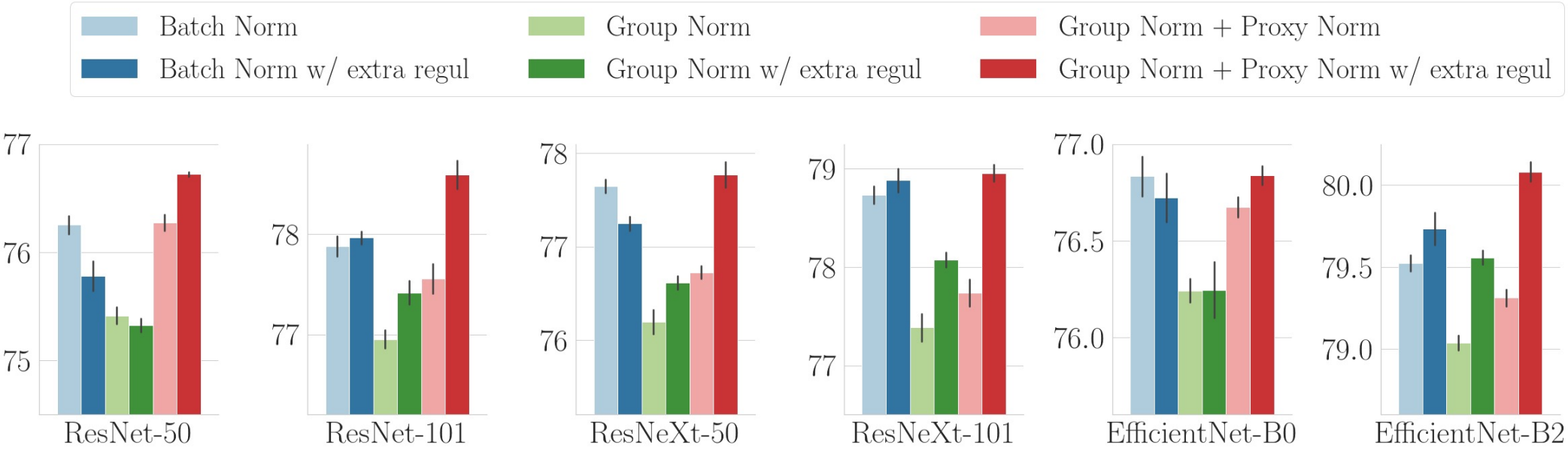
- BN introduces an inherent regularization
- We subtract away the effect of this by running each configuration without and with additional degrees of regularization

RESULTS ON IMAGENET

When subtracting away the effect of regularization, Group Norm + Proxy Norm's performance consistently matches or exceeds Batch Norm's performance.

Recipe for good performance on ImageNet:

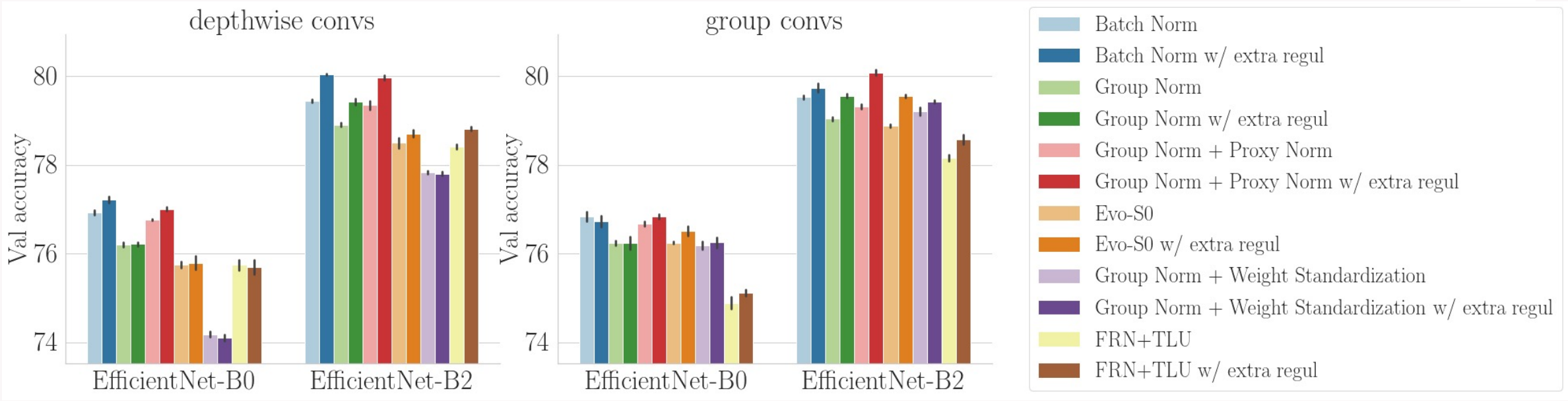
good performance ⇔ efficient normalization + efficient regularization



RESULTS ON IMAGENET — EFFICIENTNETS

On EfficientNets, none of the alternative batch-independent approaches from the literature matches Batch Norm:

- This was the **starting point** of the design of Proxy Norm



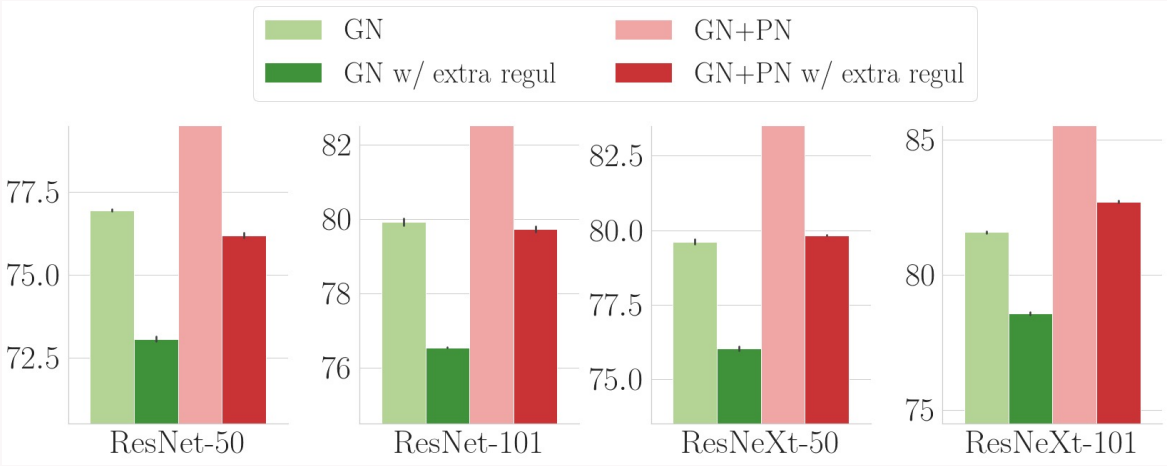
RESULTS ON IMAGENET — TRAINING ACCURACY

Group Norm + Proxy Norm always outperformed other batch-independent approaches in terms of final training accuracy.

The benefits of Group Norm + Proxy Norm are likely to increase for larger datasets:

- Validation accuracy will likely become more tied to training accuracy
- BN's regularization will likely become more detrimental
- The recipe for good performance will likely become:

good performance ⇔ efficient normalization

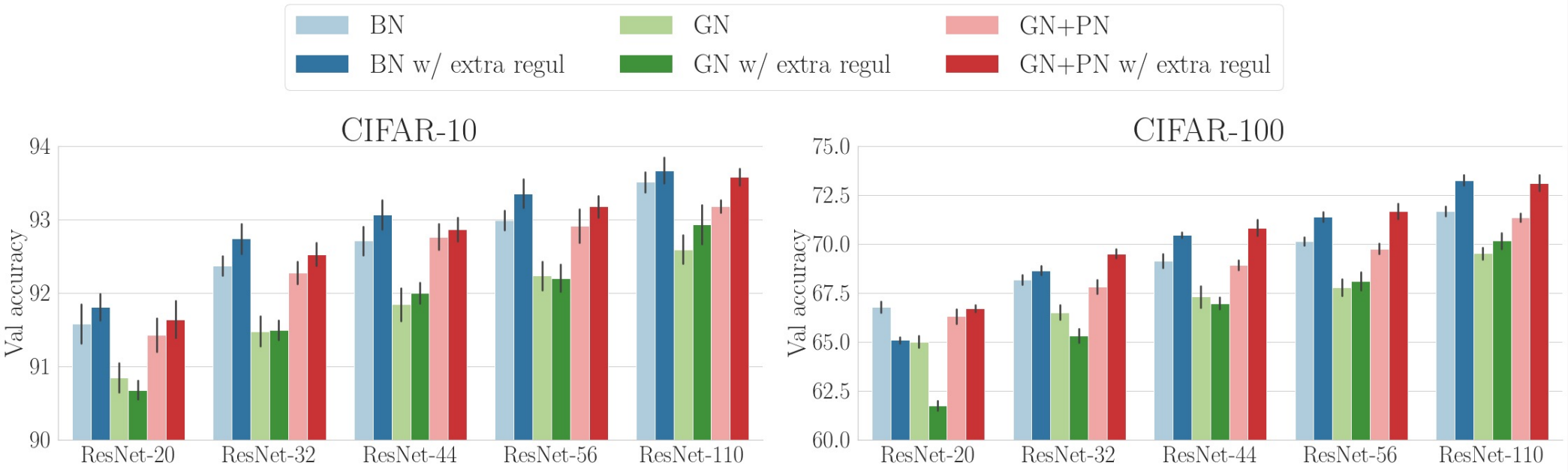


RESULTS ON CIFAR

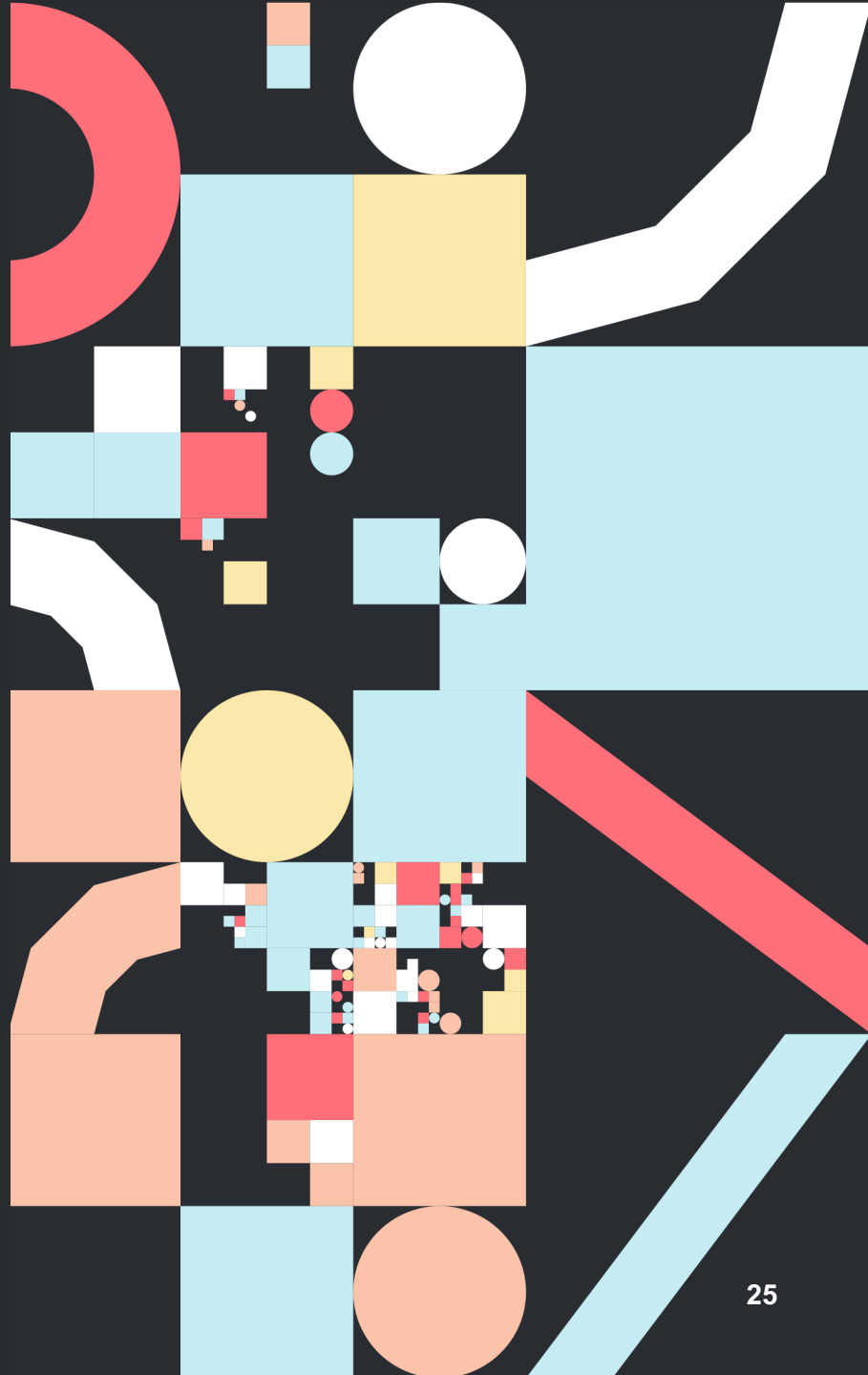
Again, when subtracting away the effect of regularization, Group Norm + Proxy Norm's performance consistently matches or exceeds Batch Norm's performance.

Recipe for good performance on CIFAR:

good performance ⇔ efficient normalization + efficient regularization



5. CONCLUSION



GRAPHCORE

CONCLUSION

- Normalization is a critical component of deep neural networks to reach optimal performance for a given model size.

- With Batch Norm — the go-to normalization in convolutional networks:
  - ✓ $\boldsymbol{Y}$ remains channel-wise normalized **and** expressivity is preserved
  - ✗ The batch dependence is incompatible with memory-efficient training

- With Layer Norm / Group Norm / Instance Norm:
  - ✗ Either $\boldsymbol{Y}$ is “far” from channel-wise normalized **or** expressivity is not preserved
  - ✓ The batch independence is compatible with memory-efficient training

- With the combination of Layer Norm / Group Norm (w/ few groups) and our technique Proxy Norm:
  - ✓ $\boldsymbol{Y}$ remains channel-wise normalized **and** expressivity is preserved
  - ✓ The batch independence is compatible with memory-efficient training

PROXY NORM'S PRACTICALITY

How is Proxy Norm implemented?

```
def proxy_norm_act(y,
                   activation_fn=tf.nn.relu,
                   proxy_epsilon=0.03,
                   num_samples=256):
    """
    TensorFlow 1 implementation of the proxy normalized activation step.
    """
    def create_channelwise_variable(name, init):
        num_channels = int(y.get_shape()[-1])
        return tf.get_variable(name,
                               dtype=y.dtype,
                               shape=[1, 1, 1, num_channels],
                               initializer=tf.constant_initializer(init))
    # scale and shift parameters after the norm
    beta = create_channelwise_variable('beta', 0.0)
    gamma = create_channelwise_variable('gamma', 1.0)

    # real activations
    Z = activation_fn(gamma * Y + beta)

    # proxy activations
    tilde_Y = uniformly_sampled_gaussian(num_samples)
    tilde_Z = activation_fn(gamma * tilde_Y + beta)

    # normalize real activations according to proxy statistics
    proxy_mean, proxy_var = tf.nn.moments(tilde_Z, axes=[0], keepdims=True)
    tilde_Z = (Z - proxy_mean) * tf.rsqrt(proxy_var + proxy_epsilon)
    return tilde_Z
```

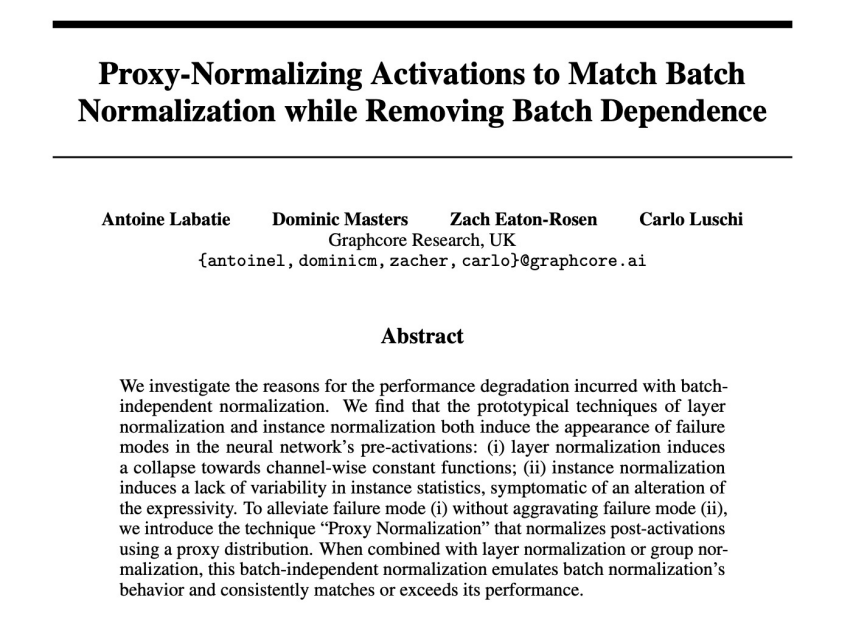
Is Proxy Norm always applicable?

- Nearly…
- Still favorable to have activation functions directly preceded by a normalization (ResNets v2 / ResNeXts v2 / EfficientNets)

FURTHER INFORMATION

Paper

Proxy-Normalizing Activations to Match Batch Normalization while Removing Batch Dependence, A. Labatie, D. Masters, Z. Eaton-Rosen, C. Luschi, to appear in NeurIPS 2021



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

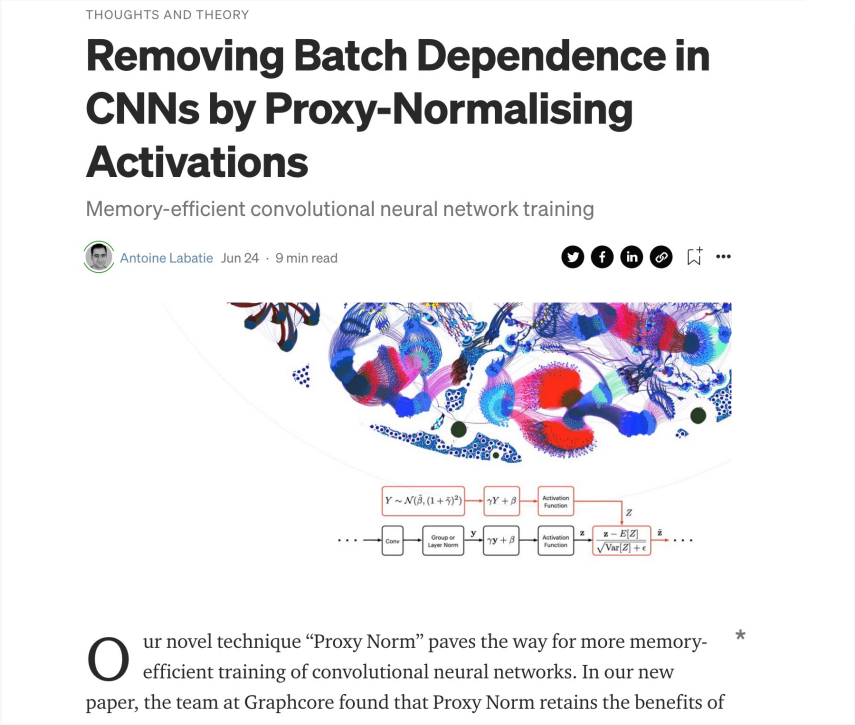
Antoine Labatie **Dominic Masters** **Zach Eaton-Rosen** **Carlo Luschi**
Graphcore Research, UK
{antoinel, dominicm, zacher, carlo}@graphcore.ai

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.

Blog post

Removing Batch Dependence in CNNs by Proxy-Normalizing Activations, A. Labatie, Towards Data Science, 2021



THOUGHTS AND THEORY

Removing Batch Dependence in CNNs by Proxy-Normalising Activations

Memory-efficient convolutional neural network training

Antoine Labatie Jun 24 · 9 min read

Our novel technique "Proxy Norm" paves the way for more memory-efficient training of convolutional neural networks. In our new paper, the team at Graphcore found that Proxy Norm retains the benefits of