

DeepMind

# Normalizer-Free Networks

Soham De

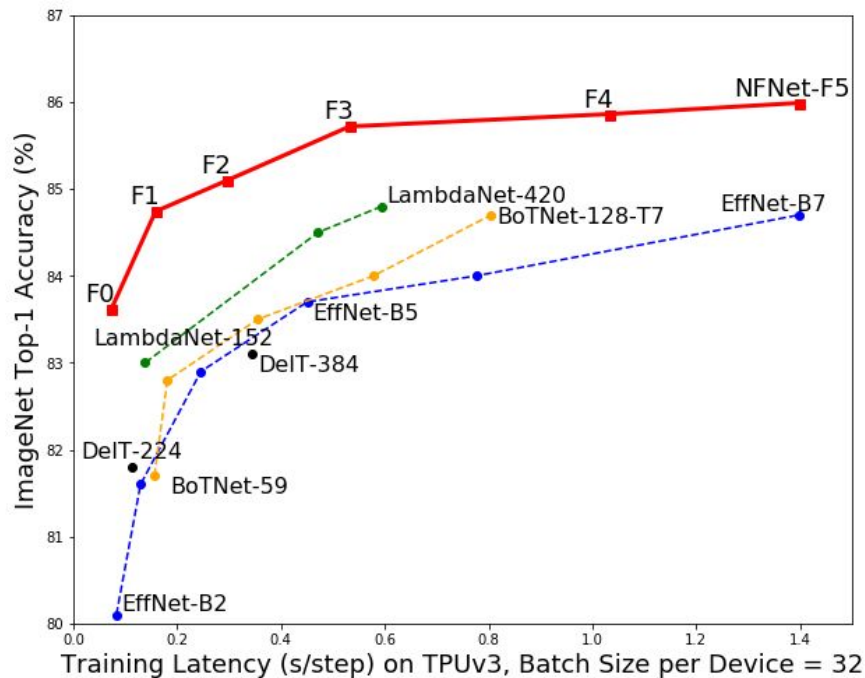
[sohamde@google.com](mailto:sohamde@google.com)

11/03/2021



# Normalizer-Free Networks (NFNets)

- A new family of ResNets
- Normalizer-Free
  - No BatchNorm!
- State-of-the-art on ImageNet for a range of training latencies
  - Up to 86.5% top-1 w/o extra data
  - 8.7x faster to train than EfficientNet-B7 to same validation accuracy
- Code in JAX: <http://dpmd.ai/nfnets>



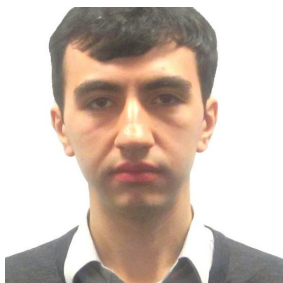
# A sequence of 3 papers



Samuel L. Smith



Andrew Brock



Karen Simonyan

---

## Batch Normalization Biases Residual Blocks Towards the Identity Function in Deep Networks

---

Soham De  
DeepMind, London  
sohamde@google.com

Samuel L. Smith  
DeepMind, London  
slsmith@google.com

## CHARACTERIZING SIGNAL PROPAGATION TO CLOSE THE PERFORMANCE GAP IN UNNORMALIZED RESNETS

Andrew Brock, Soham De & Samuel L. Smith  
Deepmind  
{aibrock, sohamde, slsmith}@google.com

---

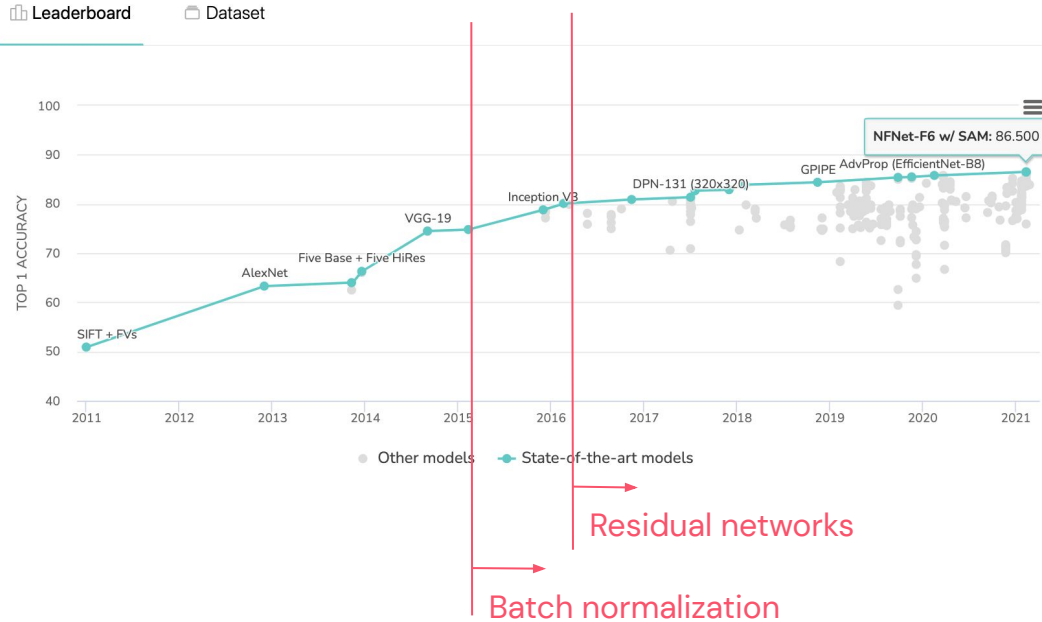
## High-Performance Large-Scale Image Recognition Without Normalization

---

Andrew Brock<sup>1</sup> Soham De<sup>1</sup> Samuel L. Smith<sup>1</sup> Karen Simonyan<sup>1</sup>



# Image Classification on ImageNet



Almost all SOTA networks since 2016 have used both:

1. Skip connections
2. Batch normalization



Why has this combination been so dominant?

## Batch normalization: Accelerating deep network training by reducing internal covariate shift

[S Ioffe](#), [C Szegedy](#) - International conference on machine ..., 2015 - proceedings.mlr.press

Abstract Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as internal covariate shift, and address the problem by normalizing layer inputs. Our method draws its strength from making ...

☆ 77 Cited by 25213 Related articles All 25 versions 🔗

## Deep residual learning for image recognition

[K He](#), [X Zhang](#), [S Ren](#), [J Sun](#) - Proceedings of the IEEE ..., 2016 - openaccess.thecvf.com

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer ...

☆ 77 Cited by 71088 Related articles 🔗



# Batch normalization in ResNets

\*Standard practice to include 2 or 3 convolutions on each residual branch

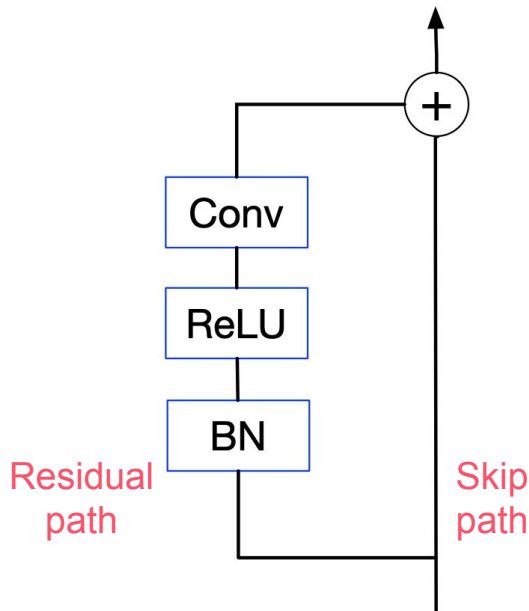
## Residual networks:

Identity skip connection bypasses all linear/non-linear operations (except transition blocks)

## Batch normalization layer:

$$O_{b,x,y,c} = \gamma_c \frac{I_{b,x,y,c} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + \beta_c$$

- $b$ : minibatch,  $c$ : channels,  $x, y$ : spatial coordinates
- $\mu$ : per-channel mean,  $\sigma$ : per-channel variance
- $\gamma$  and  $\beta$ : “scale” and “shift” parameters
- We store running averages of  $\mu$  and  $\sigma$  to use at inference



# Batch normalization in ResNets

BatchNorm has helped us made significant progress on popular benchmarks. However:

- **Surprisingly expensive during training, incurring memory overhead**
- **Introduces a train–test discrepancy**
  - Requiring maintenance of running statistics/hidden hyper–parameters
- **Breaks the independence between training examples in the loss**
  - Hard to replicate and often the cause of bugs, especially during distributed training
  - Incompatible with certain tasks (e.g. sequence modelling or contrastive losses)

\*Alternative normalizers don't generalize as well, and add their own disadvantages such as additional compute costs at inference.



# Batch normalization in ResNets

BatchNorm has helped us made significant progress on popular benchmarks. However:

- **Surprisingly expensive during training, incurring memory overhead**
- **Introduces a train–test discrepancy**
  - Requiring maintenance of running statistics/hidden hyper–parameters
- **Breaks the independence between training examples in the loss**
  - Hard to replicate and often the cause of bugs, especially during distributed training
  - Incompatible with certain tasks (e.g. sequence modelling or contrastive losses)

**Philosophy of this talk:**

1. Identify the origin of BatchNorm's benefits
2. Replicate these benefits in **unnormalized** networks

\*Alternative normalizers don't generalize as well, and add their own disadvantages such as additional compute costs at inference.



# Four benefits of BatchNorm (in ResNets)

1. **BatchNorm biases ResNets towards the skip path, fixing bad init**
2. **BatchNorm enables efficient training with larger minibatches**
3. **BatchNorm can act as an implicit regularizer**
4. **BatchNorm eliminates mean-shift in ReLU networks**





# Four benefits of BatchNorm (in ResNets)

**1. BatchNorm biases ResNets towards the skip path, fixing bad init**

This is the single most important benefit:

Explains why we can train BN-ResNets with 1000s of layers!

**2. BatchNorm enables efficient training with larger minibatches**

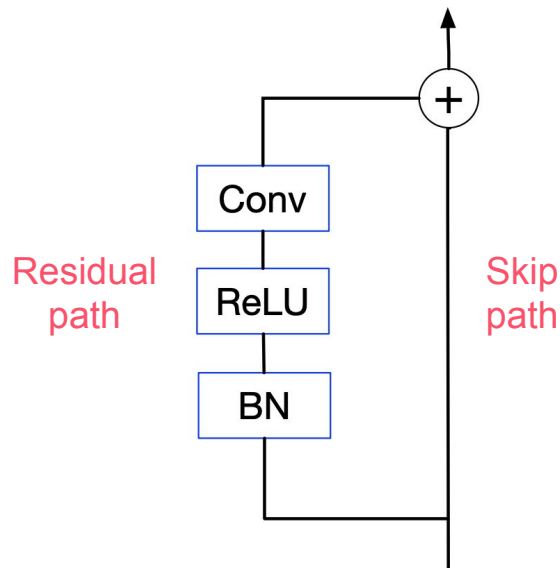
**3. BatchNorm can act as an implicit regularizer**

**4. BatchNorm eliminates mean-shift in ReLU networks**

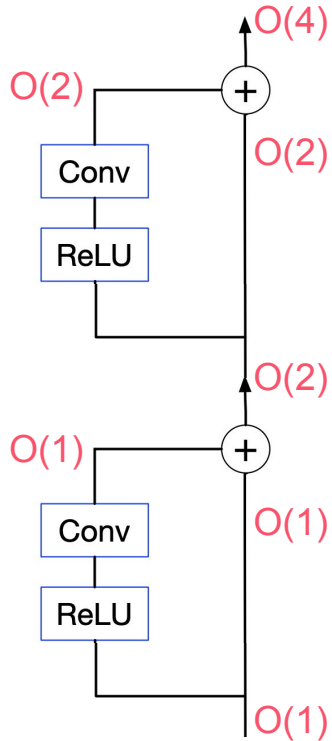


# Benefit 1: BatchNorm biases the signal at initialization towards the skip path

- This key benefit arises when we combine ResNets with activation normalization layers
- Normalization layers placed on the residual branch
- We compare the scale of hidden activations at init on unnormalized and normalized ResNets



# Unnormalized ResNets (at initialization)

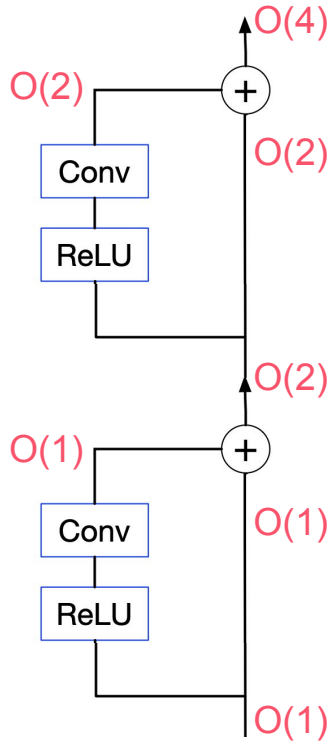


Tracking the variance of the activations

Variance of the output of the  $L$ -th residual block =  $O(2^L)$



# Unnormalized ResNets (at initialization)



## Tracking the variance of the activations

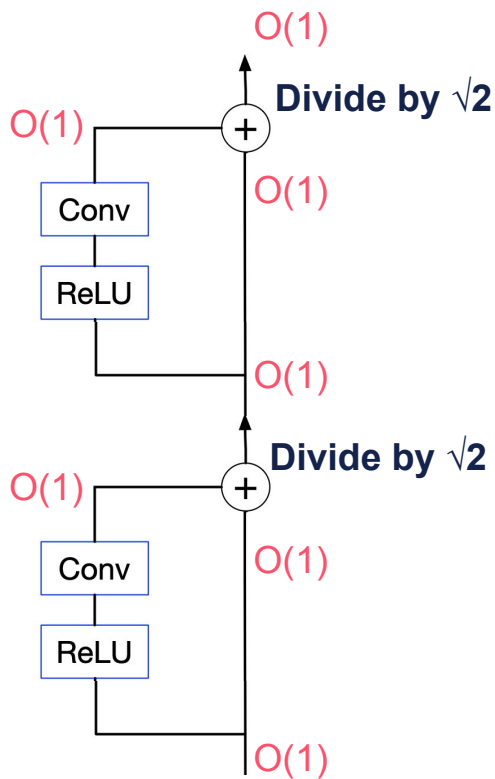
Variance of the output of the  $L$ -th residual block =  $O(2^L)$

Depth	Test accuracy	Learning rate
16	$93.0 \pm 0.1$	$2^{-2}$ ( $2^{-2}$ to $2^{-1}$ )
100	—	—
1000	—	—

(CIFAR-10 on a Wide-ResNet with fixed width and varying depth)



# Unnormalized ResNets (at initialization)



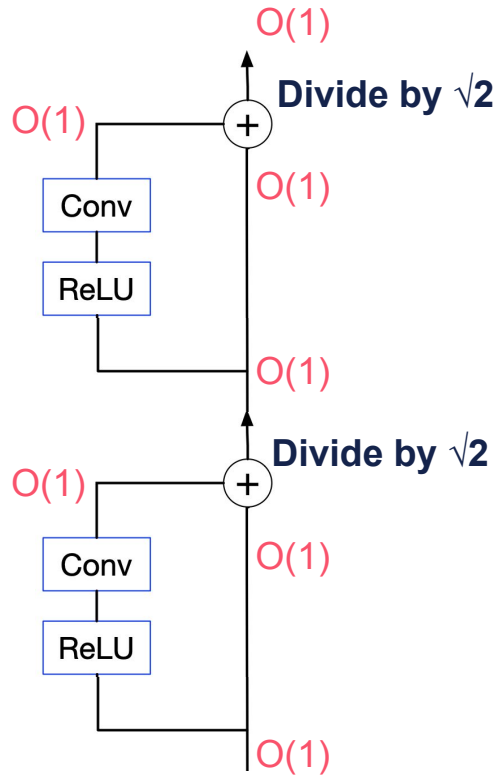
Variance of the output of the L-th residual block =  $O(1)$

Training still fails with depth:

	Divide residual block by $\sqrt{2}$	
Depth	Test accuracy	Learning rate
16	$92.4 \pm 0.1$	$2^{-2}$ ( $2^{-2}$ to $2^{-1}$ )
100	$88.9 \pm 0.4$	$2^{-5}$ ( $2^{-5}$ to $2^{-5}$ )
1000	—	—



# Unnormalized ResNets (at initialization)



Why does this fail?

Residual and skip branches contribute equally to output

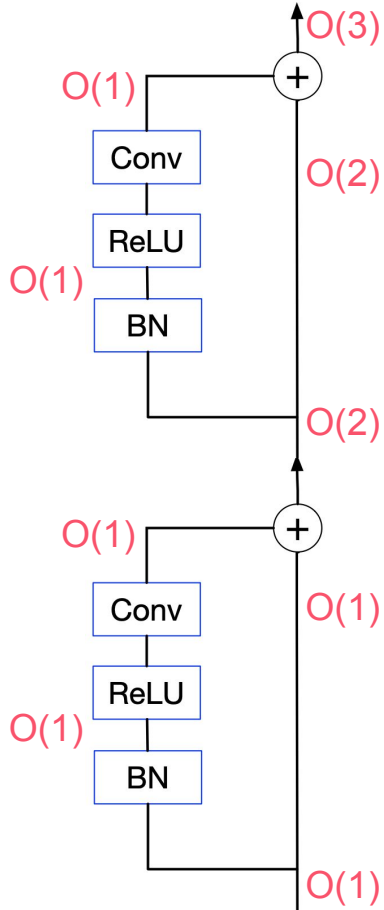
“Effective depth” of this ResNet =  $d/2$

→ Signal poorly conditioned

( $d$  = total number of residual blocks)



# Normalized ResNets (at initialization)



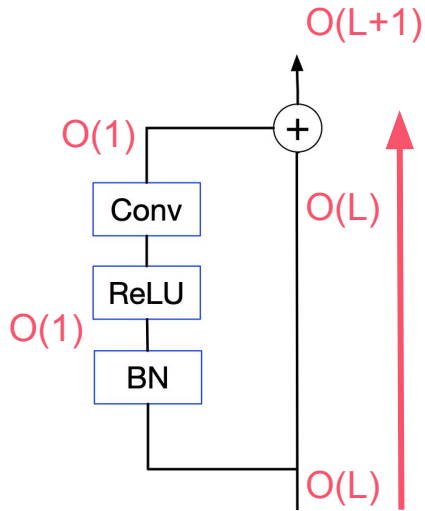
Residual branches always contribute  $O(1)$

Variance of the output of the  $L$ -th residual block =  $O(L)$



# Normalized ResNets (at initialization)

BatchNorm downscales L-th residual branch by  $O(\sqrt{L})$



Residual branch contributes  $1/(L+1)$  fraction of the variance

Output signal dominated by skip path for large  $L$

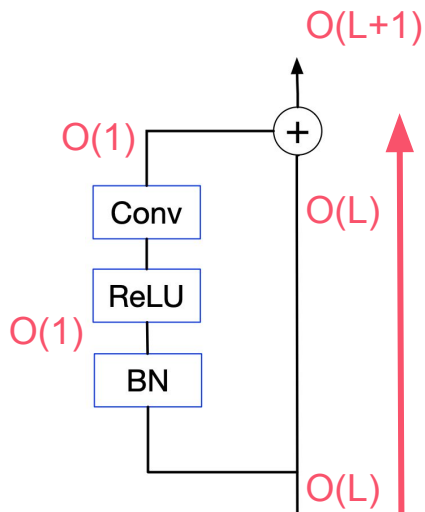
**“Effective depth”  $\ll d$**





# Normalized ResNets (at initialization)

BatchNorm downscales L-th residual branch by  $O(\sqrt{L})$



Residual branch contributes  $1/(L+1)$  fraction of the variance

Output signal dominated by skip path for large  $L$

“Effective depth”  $\ll d$

Optimal learning rate almost independent of depth

Batch Normalization		
Depth	Test accuracy	Learning rate
16	$93.5 \pm 0.1$	$2^{-1}$ ( $2^{-1}$ to $2^{-1}$ )
100	$94.7 \pm 0.1$	$2^{-1}$ ( $2^{-2}$ to $2^{-0}$ )
1000	$94.6 \pm 0.1$	$2^{-2}$ ( $2^{-3}$ to $2^{-0}$ )



# Simplest way to recover this benefit?

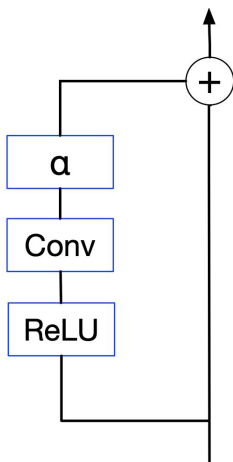
BatchNorm downscales L-th residual branch by  $O(\sqrt{L})$

Depth of typical residual block:  $O(d)$

→ BatchNorm downscales by  $O(\sqrt{d})$  on average



# Simplest way to recover this benefit?



BatchNorm downscales L-th residual branch by  $O(\sqrt{L})$

Depth of typical residual block:  $O(d)$

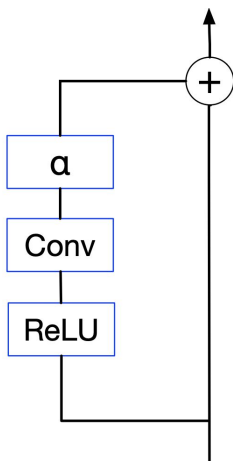
→ BatchNorm downscales by  $O(\sqrt{d})$  on average

**SkipInit:**

**Include a learnable scalar multiplier on each residual branch initialized to  $\alpha \leq 1/\sqrt{d}$**



# Simplest way to recover this benefit?



**SkipInit** can train very deep residual networks w/o BatchNorm

Optimal learning rate almost independent of depth

<b>SkipInit (<math>\alpha = 1/\sqrt{d}</math>)</b>		
<b>Depth</b>	<b>Test accuracy</b>	<b>Learning rate</b>
16	$93.0 \pm 0.1$	$2^{-2}$ ( $2^{-2}$ to $2^{-1}$ )
100	$94.2 \pm 0.1$	$2^{-1}$ ( $2^{-2}$ to $2^{-1}$ )
1000	$94.2 \pm 0.0$	$2^{-1}$ ( $2^{-2}$ to $2^{-1}$ )

<b>SkipInit (<math>\alpha = 0</math>)</b>		
<b>Depth</b>	<b>Test accuracy</b>	<b>Learning rate</b>
16	$93.3 \pm 0.1$	$2^{-2}$ ( $2^{-2}$ to $2^{-2}$ )
100	$94.2 \pm 0.1$	$2^{-2}$ ( $2^{-2}$ to $2^{-2}$ )
1000	$94.3 \pm 0.2$	$2^{-2}$ ( $2^{-3}$ to $2^{-1}$ )



## Benefits 2 and 3: Large batch training and implicit regularization

**These two benefits are best identified by comparing BN-ResNets and Skiplnit-ResNets at a range of batch sizes**

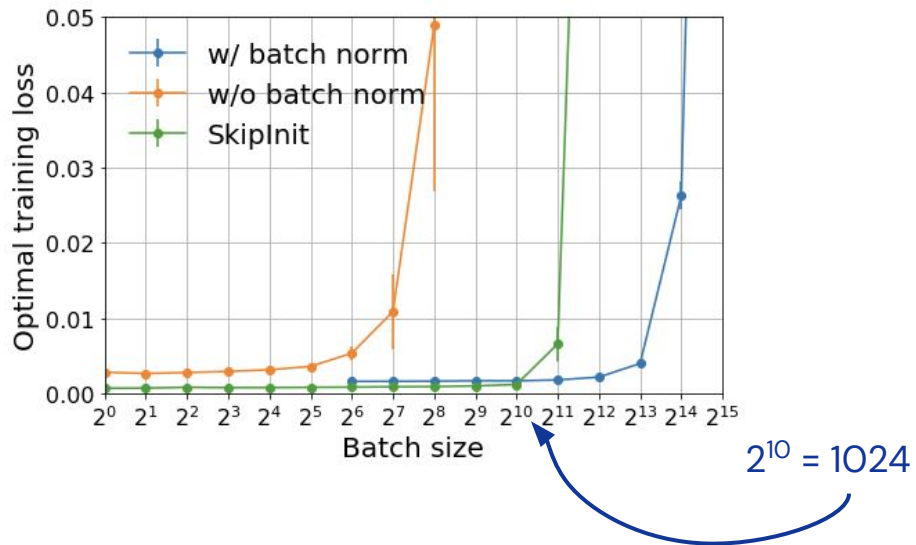
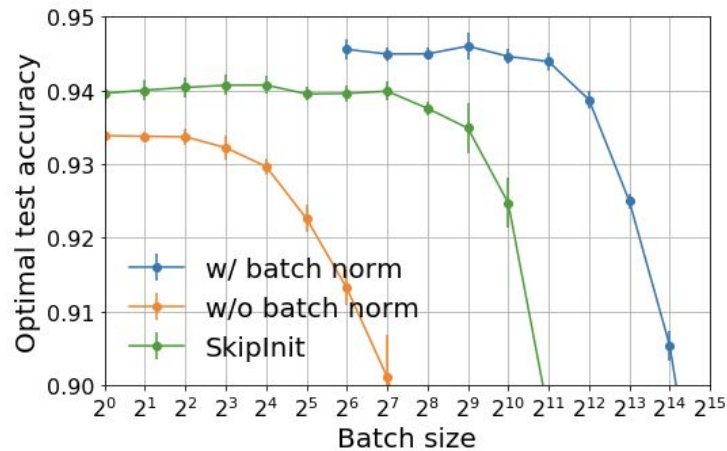
We train a 16-4 Wide-ResNet on CIFAR-10 for 200 epochs

At each batch size, we tune the learning rate over a logarithmic grid, in order to identify:

1. Learning rate which minimizes training loss
2. Learning rate which maximizes test accuracy



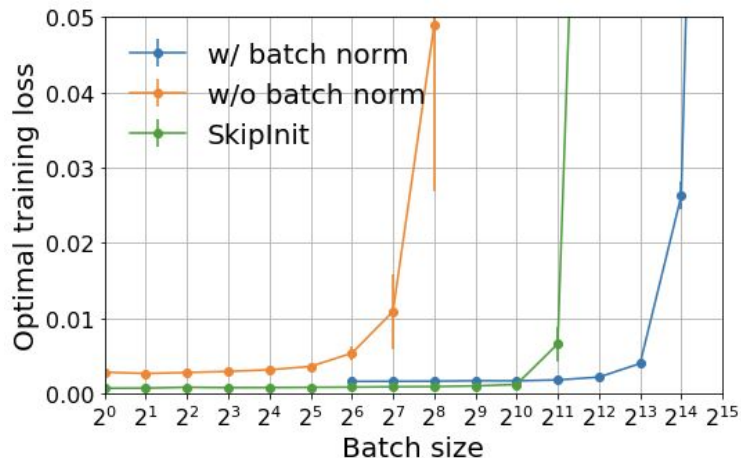
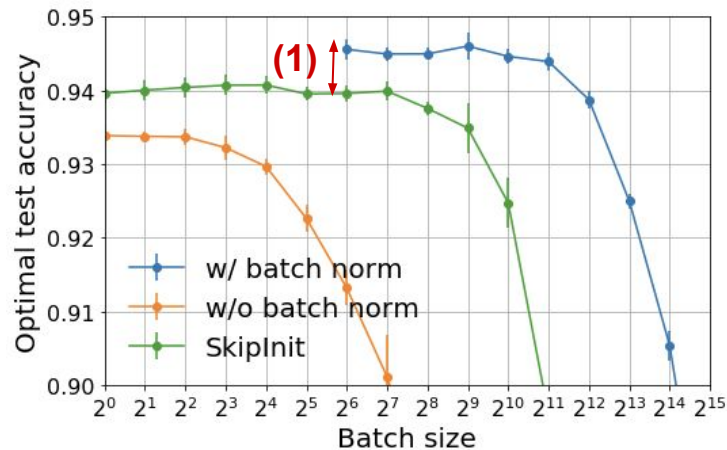
## Benefits 2 and 3:



- SkipInit reaches smaller training losses than BatchNorm but...



## Benefits 2 and 3:



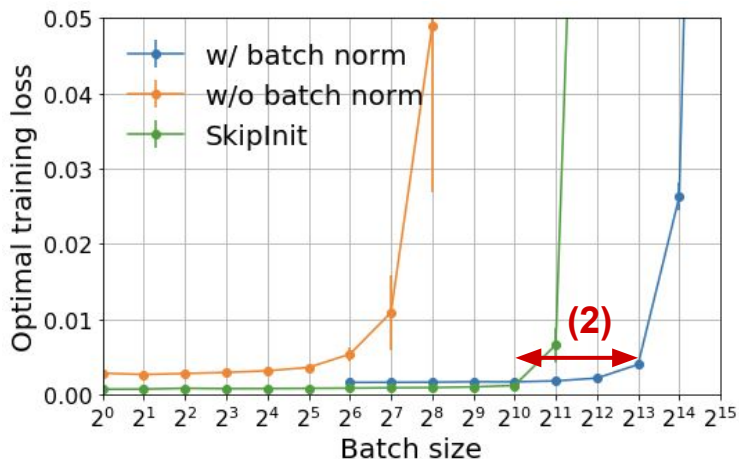
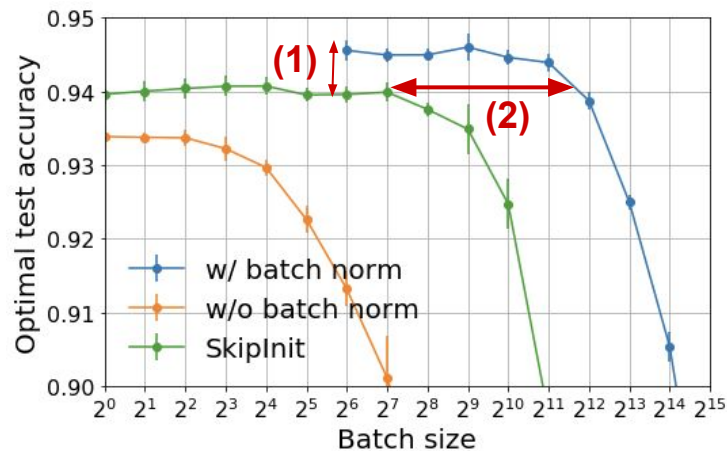
- SkipInit reaches smaller training losses than BatchNorm but...
- Two additional benefits:
  1. **Implicit regularization\***

\*Believed to arise from the noise in the batch statistics. E.g. see [Luo et al., Towards Understanding Regularization in Batch Normalization, ICLR 2019](#)

Requires us to tune "ghost batch size"



## Benefits 2 and 3:



- SkipInit reaches smaller training losses than BatchNorm but...
- Two additional benefits:
  1. **Implicit regularization\***
  2. **Efficient large batch training**

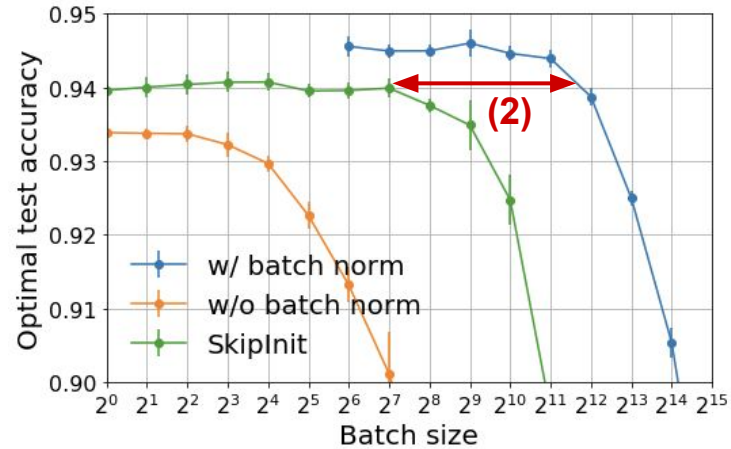
\*Believed to arise from the noise in the batch statistics. E.g. see [Luo et al., Towards Understanding Regularization in Batch Normalization, ICLR 2019](#)

Requires us to tune "ghost batch size"

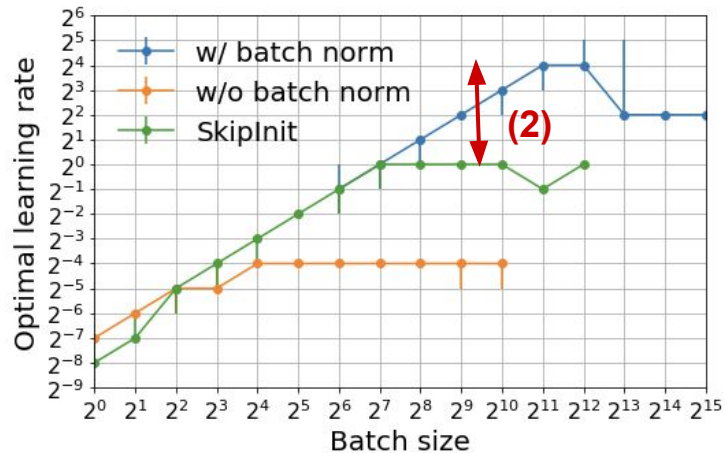
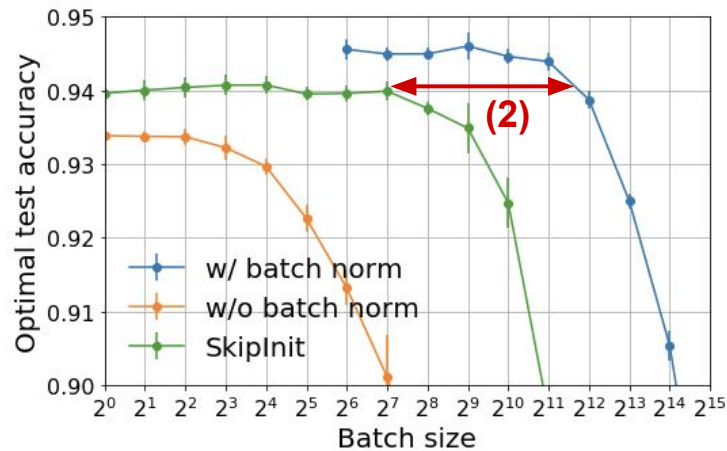




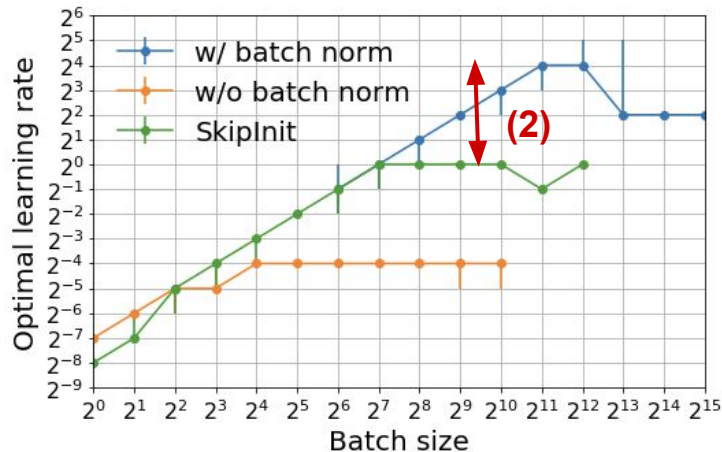
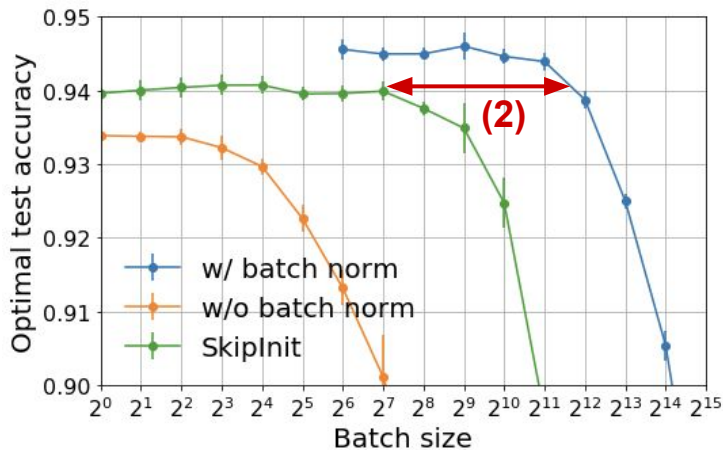
# Why can BatchNorm scale to larger batch sizes?



# Why can BatchNorm scale to larger batch sizes?



# Why can BatchNorm scale to larger batch sizes?



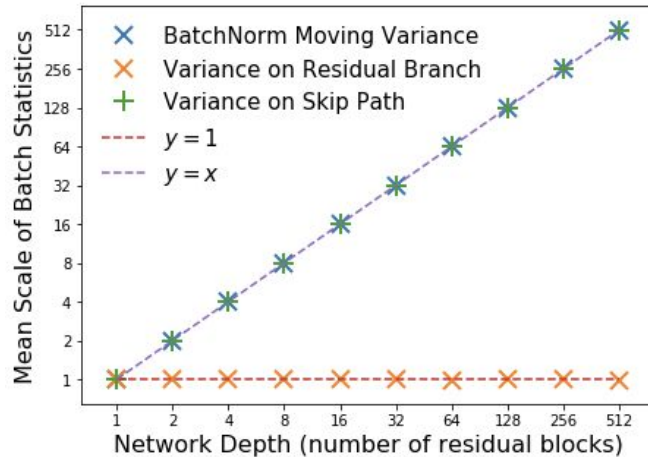
- **BatchNorm increases the maximum stable learning rate**
  - suggests the loss landscape is better conditioned\*
- **Note: small batch training is noise dominated, not curvature dominated**

\*See Santurkar et al., [How Does Batch Normalization Help Optimization?](#), NeurIPS 2018, Bjorck et al., [Understanding Batch Normalization](#), NeurIPS 2018, and Smith et al., [On the Generalization Benefit of Noise in Stochastic Gradient Descent](#), ICML 2020.



## Benefit 4: Eliminating mean-shift (in ReLU networks)

Variance of signal on skip path at init grows linearly as predicted



Deep linear normalized ResNet

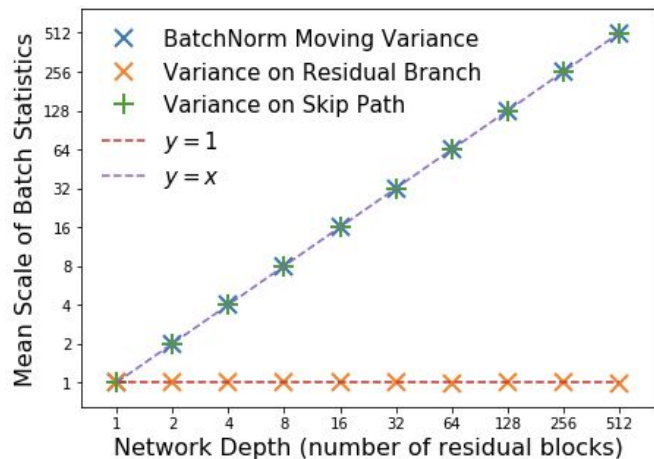


## Benefit 4: Eliminating mean-shift (in ReLU networks)

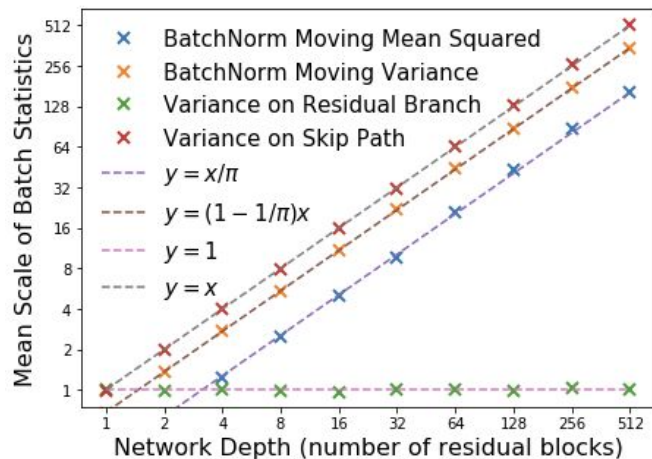
Variance of signal on skip path at init grows linearly as predicted

But in ReLU networks, the squared BatchNorm means also grow linearly

Why?



Deep linear normalized ResNet



Deep ReLU normalized ResNet

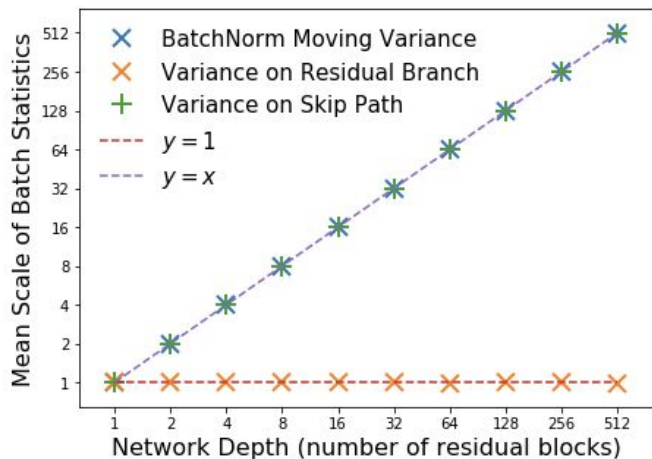


## Benefit 4: Eliminating mean-shift (in ReLU networks)

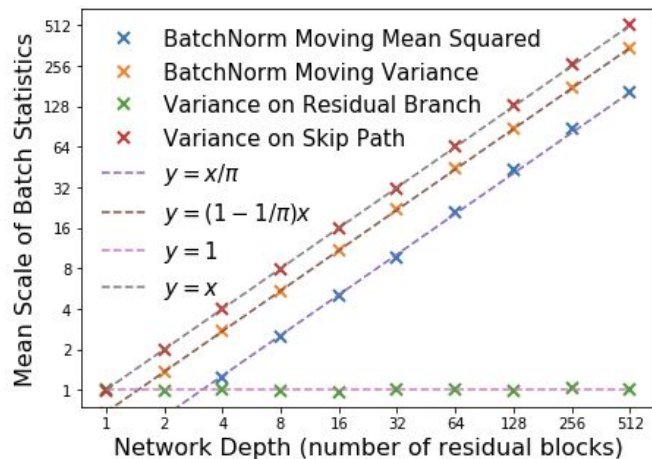
Variance of signal on skip path at init grows linearly as predicted

But in ReLU networks, the squared BatchNorm means also grow linearly

Why? **Because the outputs of ReLU activations have positive mean**      $\text{ReLU}(x) = \max(x, 0)$



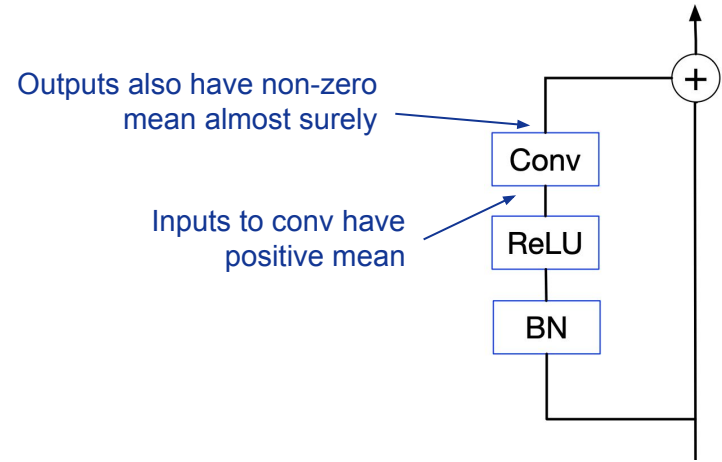
Deep linear normalized ResNet



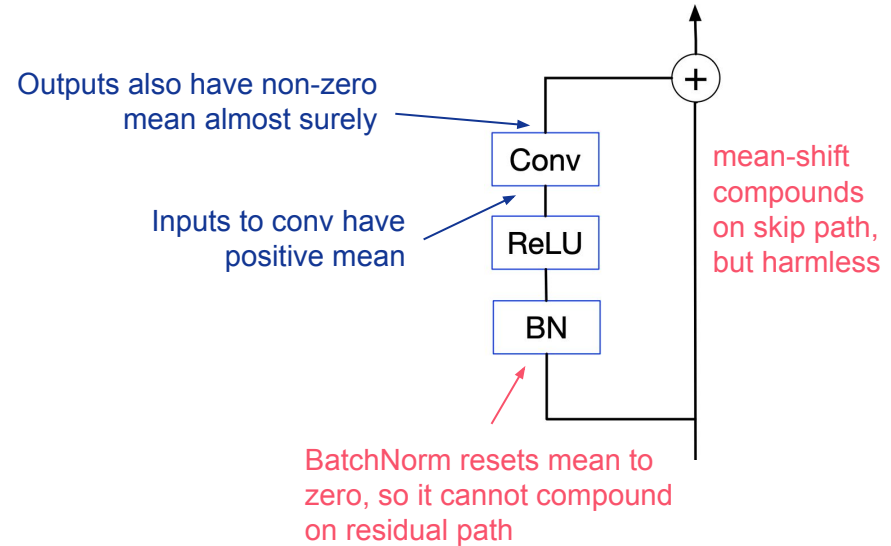
Deep ReLU normalized ResNet



# Why is there a mean-shift?



# Why is there a mean-shift?



\*ReLU-BN-Conv ordering trains equally stably while avoiding the mean-shift on the skip path





# Why is there a mean-shift?

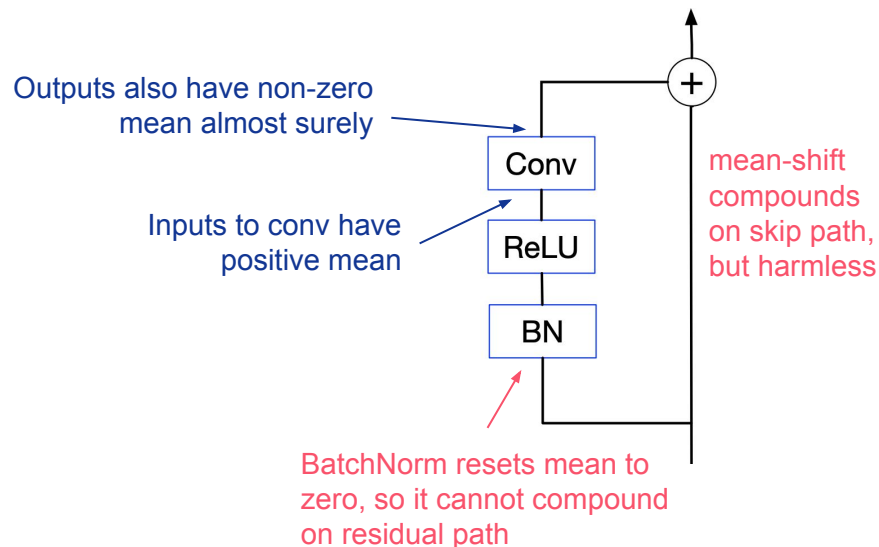
If the activation function  $g(x)$  is not antisymmetric, the output of the conv layer will have mean proportional to  $\mu_W$

$$z = Wg(x)$$

$$z_i = \sum_j^N W_{i,j}g(x_j)$$

$$\mathbb{E}(z_i) = N\mu_g\mu_{W_{i,.}}$$

where  $\mu_{W_{i,.}}$  is the mean of the  $i^{th}$  row of  $W$



Even when sampled from  $N(0, 1)$ , any specific initialization of  $W$  will almost surely have non-zero  $\mu_W$

(even in infinite width limit)



# Four benefits of BatchNorm (in ResNets)

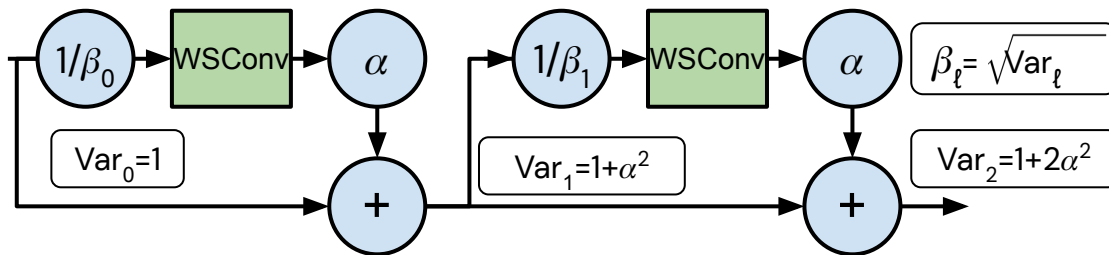
1. BatchNorm biases ResNets towards the skip path, fixing bad init
2. BatchNorm enables efficient training with larger minibatches
3. BatchNorm can act as an implicit regularizer
4. BatchNorm eliminates mean-shift in ReLU networks

Can we build normalizer-free networks that recover each of these benefits?



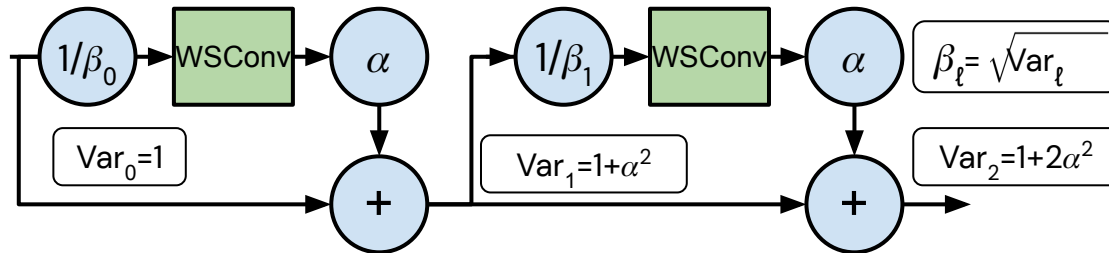
# The Normalizer-Free (NF-) Strategy

- Compatible with most existing ResNets architectures
- Fixes Initialization by downscaling residual branch and eliminating mean shift:
  - Downscale residual branch by predicting the expected variance at init
  - Eliminate mean-shift using “Scaled Weight Standardization”



# The Normalizer-Free (NF-) Strategy

- Compatible with most existing ResNets architectures
- Fixes Initialization by downscaling residual branch and eliminating mean shift:
  - Downscale residual branch by predicting the expected variance at init
  - Eliminate mean-shift using “Scaled Weight Standardization”



- We also get competitive results on ResNets using Skiplnit + Scaled WS, but the NF-strategy above is more stable at large learning rates



# Scaled Weight Standardization

**Weight Standardization (WS) prevents mean-shift by reparameterizing the model to ensure the mean of each row of the weight matrix is exactly zero**

- We use a slight modification (“Scaled Weight Standardization”) which includes a nonlinearity-specific gain ( $\gamma$ ) and applies fan-in scaling ( $1/\sqrt{N}$ ) to preserve signal variance

$$\hat{W}_{i,j} = \gamma \cdot \frac{W_{i,j} - \mu_{W_{i,\cdot}}}{\sigma_{W_{i,\cdot}} \sqrt{N}}$$

where  $\mu_{W_{i,\cdot}}$  and  $\sigma_{W_{i,\cdot}}$  are the mean and standard deviation of the  $i^{th}$  row of  $W$ :

$$\mu_{W_{i,\cdot}} = \frac{1}{N} \sum_j^N W_{i,j}, \quad \text{and} \quad \sigma_{W_{i,\cdot}}^2 = \frac{1}{N} \sum_j^N W_{i,j}^2 - \mu_{W_{i,\cdot}}^2.$$



# Scaled Weight Standardization

\*Note: none of the limitations of activation normalizers but performs poorly with depthwise convolutions

**Weight Standardization (WS) prevents mean-shift by reparameterizing the model to ensure the mean of each row of the weight matrix is exactly zero**

- We use a slight modification (“Scaled Weight Standardization”) which includes a nonlinearity-specific gain ( $\gamma$ ) and applies fan-in scaling ( $1/\sqrt{N}$ ) to preserve signal variance

$$\hat{W}_{i,j} = \gamma \cdot \frac{W_{i,j} - \mu_{W_{i,\cdot}}}{\sigma_{W_{i,\cdot}} \sqrt{N}}$$

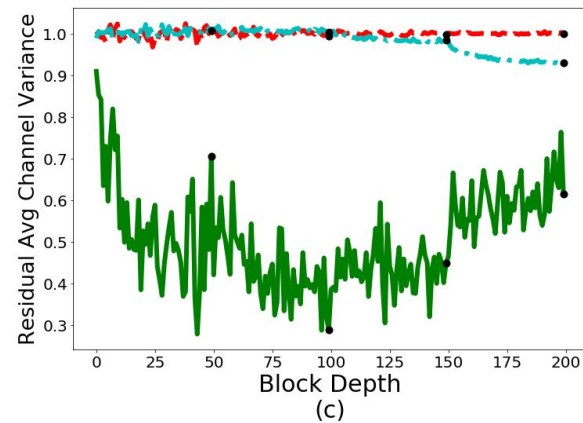
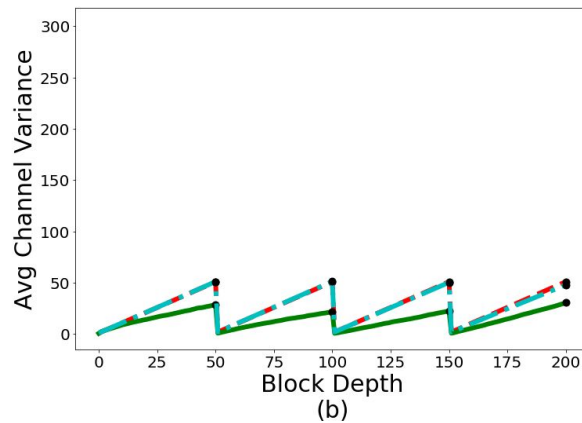
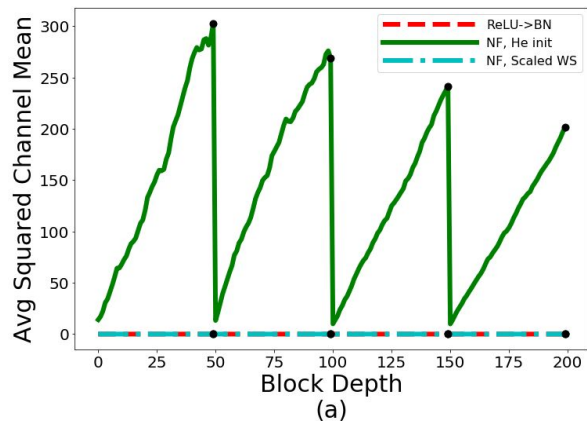
where  $\mu_{W_{i,\cdot}}$  and  $\sigma_{W_{i,\cdot}}$  are the mean and standard deviation of the  $i^{th}$  row of  $W$ :

$$\mu_{W_{i,\cdot}} = \frac{1}{N} \sum_j^N W_{i,j}, \quad \text{and} \quad \sigma_{W_{i,\cdot}}^2 = \frac{1}{N} \sum_j^N W_{i,j}^2 - \mu_{W_{i,\cdot}}^2.$$



# Signal propagation in NF-ResNets

ResNet-600 with 4 transition blocks marked by black dots

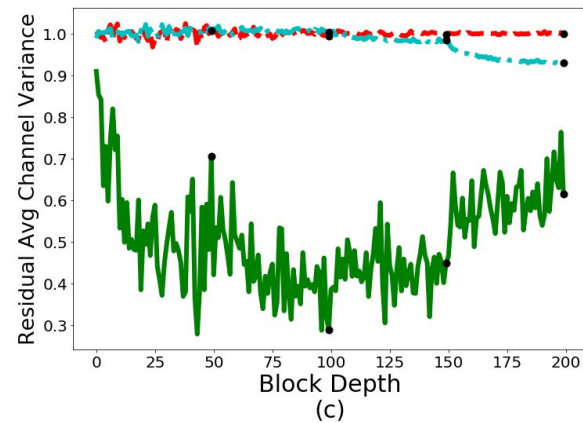
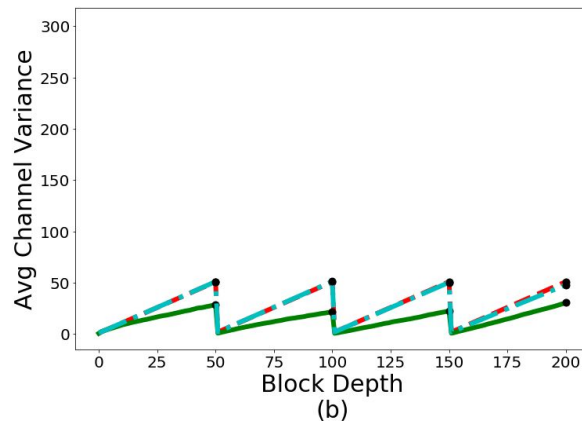
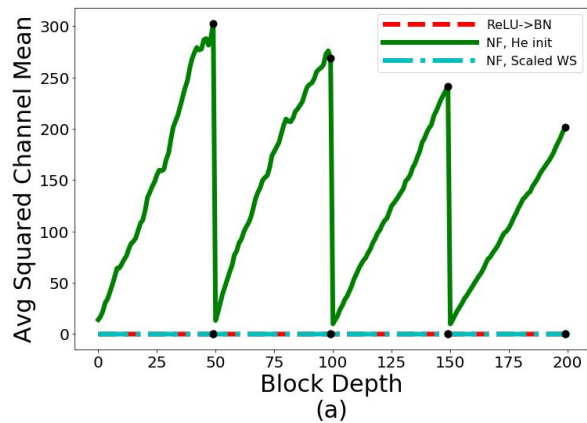


Combined with Scaled Weight Standardization, the hidden activation statistics of NF-ResNets almost exactly match the statistics of BN-ResNets at init



# Signal propagation in NF-ResNets

ResNet-600 with 4 transition blocks marked by black dots



**Combined with Scaled Weight Standardization, the hidden activation statistics of NF-ResNets almost exactly match the statistics of BN-ResNets at init**

Note, it is often sufficient to consider the forward pass in ResNets at initialization because the network is effectively shallow





## NF-ResNets at batch size 1024

- Without regularization, NF-ResNets achieve smaller training losses than BN-ResNets, but worse generalization (they do not share BatchNorm's implicit regularization effect)
- With Dropout and Stochastic Depth, NF-ResNets match BN-ResNets on ImageNet

	NF-ResNets (ours)		BN-ResNets	
	Unreg.	Reg.	Unreg.	Reg.
RN50	75.8 ± .1	<b>76.8 ± .1</b>	<b>76.8 ± .1</b>	76.4 ± .1
RN101	77.1 ± .1	<b>78.4 ± .1</b>	78.0 ± .1	78.1 ± .1
RN152	77.6 ± .1	<b>79.1 ± .1</b>	78.6 ± .2	78.8 ± .1
RN200	77.9 ± .2	<b>79.6 ± .1</b>	79.0 ± .2	79.2 ± .1
RN288	78.1 ± .1*	<b>79.5 ± .1</b>	78.8 ± .1	<b>79.5 ± .1</b>

Top-1 Accuracy on ImageNet,  
no extra data



## NF-ResNets at batch size 1024

- Without regularization, NF-ResNets achieve smaller training losses than BN-ResNets, but worse generalization (they do not share BatchNorm's implicit regularization effect)
- With Dropout and Stochastic Depth, NF-ResNets match BN-ResNets on ImageNet
- **NF-ResNets achieve substantially higher performance after pre-training**

	NF-ResNets (ours)		BN-ResNets	
	Unreg.	Reg.	Unreg.	Reg.
RN50	75.8 ± .1	<b>76.8 ± .1</b>	<b>76.8 ± .1</b>	76.4 ± .1
RN101	77.1 ± .1	<b>78.4 ± .1</b>	78.0 ± .1	78.1 ± .1
RN152	77.6 ± .1	<b>79.1 ± .1</b>	78.6 ± .2	78.8 ± .1
RN200	77.9 ± .2	<b>79.6 ± .1</b>	79.0 ± .2	79.2 ± .1
RN288	78.1 ± .1*	<b>79.5 ± .1</b>	78.8 ± .1	<b>79.5 ± .1</b>


Top-1 Accuracy on ImageNet,  
no extra data

	224px	320px	384px
BN-ResNet-50	78.1	79.6	79.9
NF-ResNet-50	<b>79.5</b>	<b>80.9</b>	<b>81.1</b>
BN-ResNet-101	80.8	82.2	82.5
NF-ResNet-101	<b>81.4</b>	<b>82.7</b>	<b>83.2</b>
BN-ResNet-152	81.8	83.1	83.4
NF-ResNet-152	<b>82.7</b>	<b>83.6</b>	<b>84.0</b>
BN-ResNet-200	81.8	83.1	83.5
NF-ResNet-200	<b>82.9</b>	<b>84.1</b>	<b>84.3</b>

Top-1 Accuracy  
on ImageNet,  
after pre-training  
on JFT



# Four benefits of BatchNorm (in ResNets)

1. **BatchNorm biases ResNets towards the skip path, fixing bad init**  
(recover with the NF-Strategy)
2. **BatchNorm enables efficient training with larger minibatches** 
3. **BatchNorm can act as an implicit regularizer**  
(Can be detrimental when pretraining on massive datasets. Recover with explicit regularization if required)
4. **BatchNorm eliminates mean-shift in ReLU networks**  
(recover with scaled weight standardization)



# Alternative methods for large batch training

Many methods enable faster convergence on ill-conditioned losses:

- Momentum
- Adaptive gradient methods like Adam
- Second order methods like KFAC
- Normalized optimizers like LARS or Fromage
- Gradient clipping

(Noise dominated,  
small learning rate)

We expect these methods to perform no better than SGD if the batch size is small,  
but to outperform SGD when the batch size is large

(Curvature dominated,  
large learning rate)

Zhang et al., [Which Algorithmic Choices Matter at Which Batch Sizes? Insights From a Noisy Quadratic Model](#), NeurIPS 2019.

Smith et al., [On the Generalization Benefit of Noise in Stochastic Gradient Descent](#), ICML 2020.



# “Adaptive Gradient Clipping”

**Intuition: Parameter updates should be small relative to the magnitude of the weight**

Consider the gradient descent update for layer  $\ell$ :  $\Delta W^\ell = -hG^\ell$

A single gradient descent step will change the original weights by:  $\frac{\|\Delta W^\ell\|}{\|W^\ell\|} = h \frac{\|G^\ell\|_F}{\|W^\ell\|_F}$

Use above ratio to clip gradients (in practice we use unit-wise ratios):

$$G_i^\ell \rightarrow \begin{cases} \lambda \frac{\|W_i^\ell\|_F^*}{\|G_i^\ell\|_F} G_i^\ell & \text{if } \frac{\|G_i^\ell\|_F}{\|W_i^\ell\|_F^*} > \lambda, \\ G_i^\ell & \text{otherwise.} \end{cases}$$

$G_i^\ell$ :  $i$  denotes the fan-in extent (channels and spatial dimensions)

Frobenius norm of  $W$  capped below at  $1e-3$



# Adaptive Gradient Clipping: Ablations

- AGC can train Normalizer-Free ResNets stably at high batch sizes with large learning rates, and with very strong augmentations
- Optimal clipping values depend on batch size
  - This is expected: training instability is related to large batch sizes/learning rates

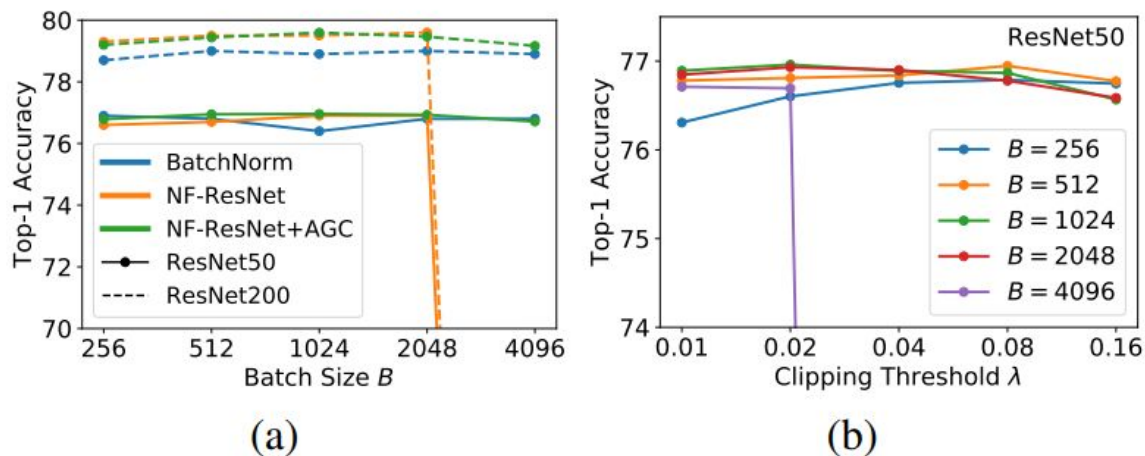


Figure 2. (a) AGC efficiently scales NF-ResNets to larger batch sizes. (b) The performance across different clipping thresholds  $\lambda$ .



# Scaling Normalizer-Free Networks to SOTA

- We now have all the pieces required to remove BatchNorm from our networks
- However the existing SOTA architecture (EfficientNets) is co-adapted to a specific form of BatchNorm, and contains Depthwise-Convs, which don't work well with Scaled WS
- We therefore set out to design a new SOTA model family
- Unlike EfficientNets, which were designed to minimize theoretical FLOPS, we chose to design networks to minimize actual training latency on target accelerators



# The NFNet Architecture

Start with simple baseline (SE-ResNeXt-D) and hand-tune using simple best practices and considering the specific properties of existing accelerators

- Different depth scaling pattern, and gently scaled input resolution
- Second added 3x3 conv, group widths of 128
- Different width pattern, no width scaling

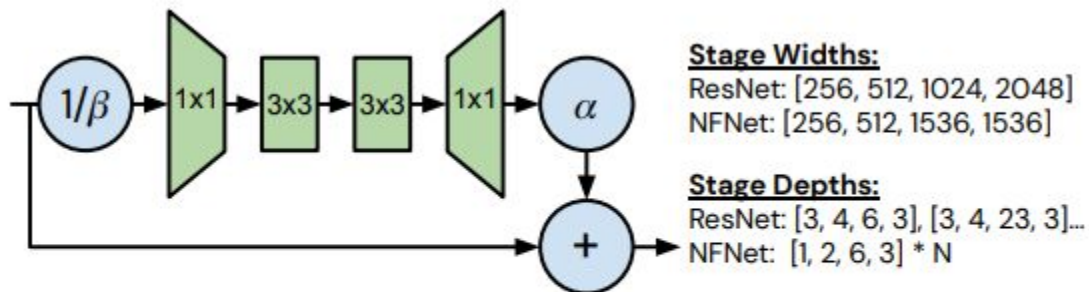


Figure 3. Summary of NFNet bottleneck block design and architectural differences. See Figure 5 in Appendix C for more details.





# The NFNets Architecture

## NFNets benefit from very strong data augmentations

- Very expressive (without the implicit regularization of BatchNorm)

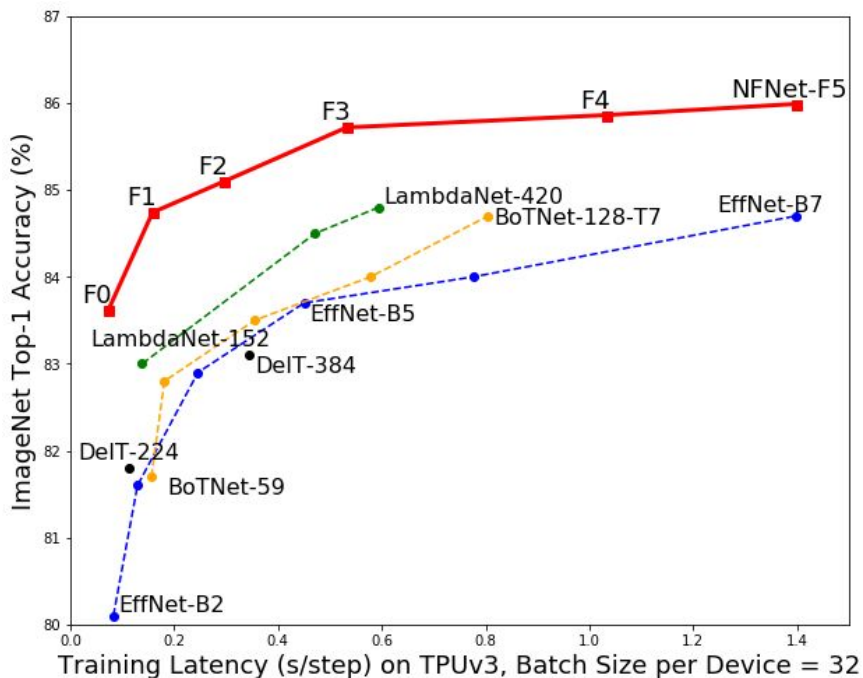
*Table 2.* The effect of architectural modifications and data augmentation on ImageNet Top-1 accuracy (averaged over 3 seeds).

	F0	F1	F2	F3
Baseline	80.4	81.7	82.0	82.3
+ Modified Width	80.9	81.8	82.0	82.3
+ Second Conv	81.3	82.2	82.4	82.7
+ MixUp	82.2	82.9	83.1	83.5
+ RandAugment	83.2	84.6	84.8	85.0
+ CutMix	<b>83.6</b>	<b>84.7</b>	<b>85.1</b>	<b>85.7</b>



# ImageNet Results

Directly optimizing for speed on accelerators yields big speedups, and a new state-of-the-art on ImageNet (8.7x faster than EfficientNet-B7 to same accuracy).



\*With Sharpness Aware Minimization (SAM), this increases to 86.5% top-1.



# ImageNet Results

Table 3. ImageNet Accuracy comparison for NFNet and a representative set of models, including SENet (Hu et al., 2018), LambdaNet, (Bello, 2021), BoTNet (Srinivas et al., 2021), and DeIT (Touvron et al., 2020). Except for results using SAM, our results are averaged over three random seeds. Latencies are given as the time in milliseconds required to perform a single full training step on TPU or GPU (V100).

Model	#FLOPs	#Params	Top-1	Top-5	TPUv3 Train	GPU Train
ResNet-50	4.10B	26.0M	78.6	94.3	41.6ms	35.3ms
EffNet-B0	0.39B	5.3M	77.1	93.3	51.1ms	44.8ms
SENet-50	4.09B	28.0M	79.4	94.6	64.3ms	59.4ms
<b>NFNet-F0</b>	<b>12.38B</b>	<b>71.5M</b>	<b>83.6</b>	<b>96.8</b>	<b>73.3ms</b>	<b>56.7ms</b>
EffNet-B3	1.80B	12.0M	81.6	95.7	129.5ms	116.6ms
LambdaNet-152	–	51.5M	83.0	96.3	138.3ms	135.2ms
SENet-152	19.04B	66.6M	83.1	96.4	149.9ms	151.2ms
BoTNet-110	10.90B	54.7M	82.8	96.3	181.3ms	–
<b>NFNet-F1</b>	<b>35.54B</b>	<b>132.6M</b>	<b>84.7</b>	<b>97.1</b>	<b>158.5ms</b>	<b>133.9ms</b>
EffNet-B4	4.20B	19.0M	82.9	96.4	245.9ms	221.6ms
BoTNet-128-T5	19.30B	75.1M	83.5	96.5	355.2ms	–
<b>NFNet-F2</b>	<b>62.59B</b>	<b>193.8M</b>	<b>85.1</b>	<b>97.3</b>	<b>295.8ms</b>	<b>226.3ms</b>
SENet-350	52.90B	115.2M	83.8	96.6	593.6ms	–
EffNet-B5	9.90B	30.0M	83.7	96.7	450.5ms	458.9ms
LambdaNet-350	–	105.8M	84.5	97.0	471.4ms	–
BoTNet-77-T6	23.30B	53.9M	84.0	96.7	578.1ms	–
<b>NFNet-F3</b>	<b>114.76B</b>	<b>254.9M</b>	<b>85.7</b>	<b>97.5</b>	<b>532.2ms</b>	<b>524.5ms</b>
LambdaNet-420	–	124.8M	84.8	97.0	593.9ms	–
EffNet-B6	19.00B	43.0M	84.0	96.8	775.7ms	868.2ms
BoTNet-128-T7	45.80B	75.1M	84.7	97.0	804.5ms	–
<b>NFNet-F4</b>	<b>215.24B</b>	<b>316.1M</b>	<b>85.9</b>	<b>97.6</b>	<b>1033.3ms</b>	<b>1190.6ms</b>
EffNet-B7	37.00B	66.0M	84.7	97.0	1397.0ms	1753.3ms
DeIT 1000 epochs	–	87.0M	85.2	–	–	–
EffNet-B8+MaxUp	62.50B	87.4M	85.8	–	–	–
<b>NFNet-F5</b>	<b>289.76B</b>	<b>377.2M</b>	<b>86.0</b>	<b>97.6</b>	<b>1398.5ms</b>	<b>2177.1ms</b>
NFNet-F5+SAM	289.76B	377.2M	86.3	97.9	1958.0ms	–
<b>NFNet-F6+SAM</b>	<b>377.28B</b>	<b>438.4M</b>	<b>86.5</b>	<b>97.9</b>	<b>2774.1ms</b>	–




# Transferring from JFT

- In this setting, NFNetS obtain second-best performance while using ~12x less compute than the current state of the art

Model	#FLOPS	#Params	ImageNet Top-1	TPUv3-core-days
NFNet-F4+ (ours)	367B	527M	89.2	1.86k
NFNet-F4 (ours)	215B	316M	89.2	3.7k
EffNet-L2 + Meta Pseudo Labels	-	480M	<b>90.2</b>	22.5k
EffNet-L2 + NoisyStudent + SAM	-	480M	88.6	12.3k
ViT-H/14	-	632M	88.55 ± 0.04	2.5k
ViT-L/16	-	307M	87.76 ± 0.03	0.68k
BiT-L ResNet152x4	-	928M	87.54 ± 0.02	9.9k
ResNeXt-101 32x48d (IG-940M)	-	829M	86.4	-



# Conclusions

- **We remove BatchNorm by studying and replicating its effects on signal propagation**
  - Downscales residual branch
  - Enables large batch training
  - Implicit regularization
  - Prevents mean-shift
  - NF-strategy
  - Adaptive gradient clipping
  - Explicit regularization
  - Scaled Weight Standardization
- **The resulting Normalizer-Free ResNets attain similar performance to BN-ResNets on ImageNet, and substantially outperform BN-ResNets after large-scale pre-training**
- **We introduce the NFNet model family, which set a new ImageNet SOTA of 86.5%, while being significantly faster to train than existing networks**

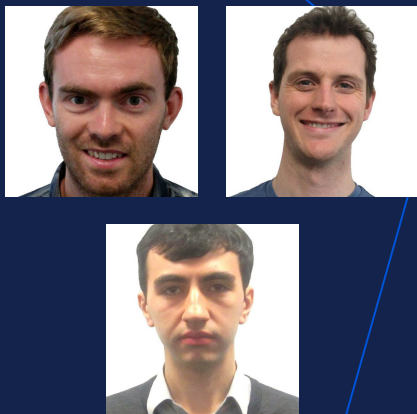


# Try our models!

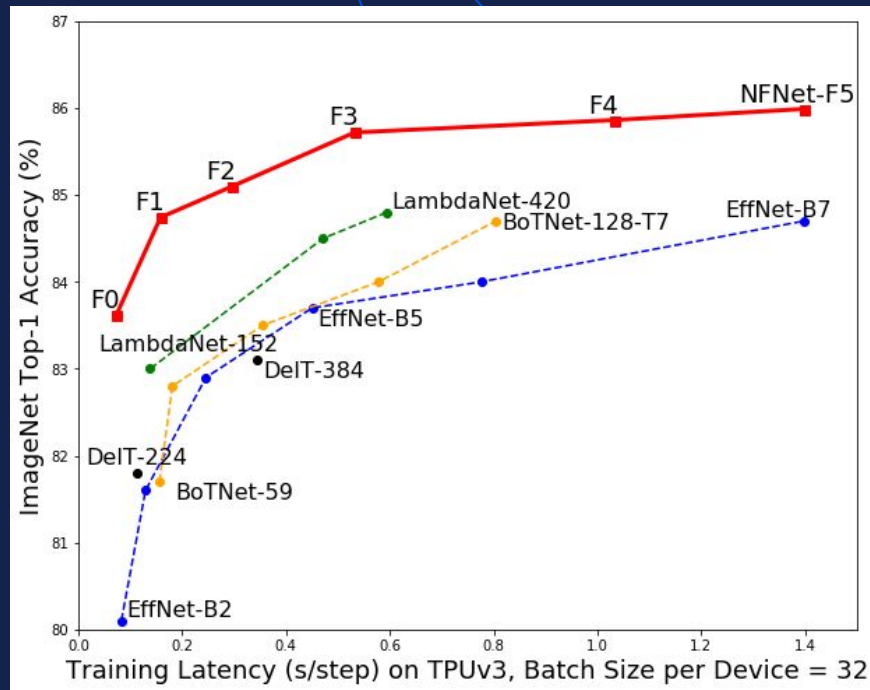
- If you're currently using ResNets for a vision backbone, our NFNet models are a drop-in replacement and are substantially more expressive
- Code and pre-trained JAX/Haiku models available at <http://dpmd.ai/nfnets>



# The end and thank you



If you have any questions please reach out [sohamde@google.com](mailto:sohamde@google.com)



Code: <http://dpmd.ai/nfnets>



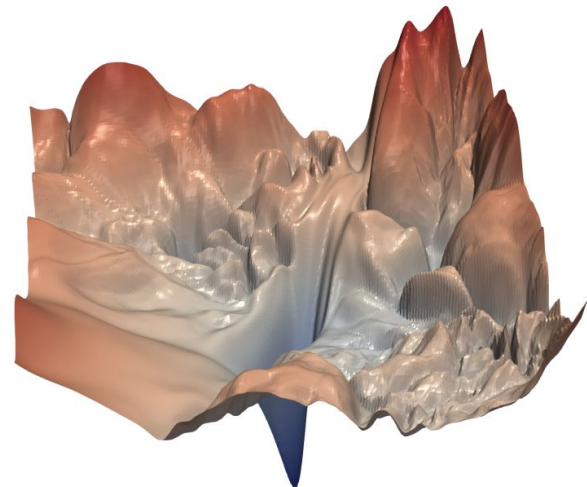
# Batch normalization + Residual networks

Led to a dramatic improvement in performance

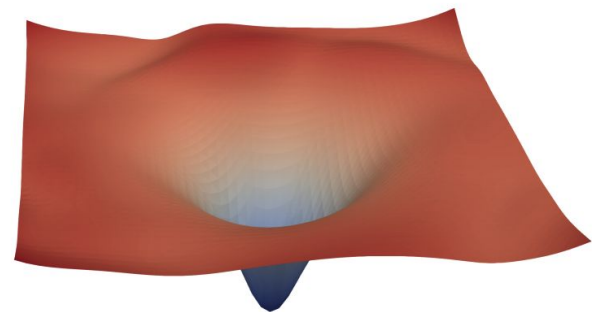
- Train significantly deeper networks (1000+ layers)
- Faster convergence
- Better generalization

Origins of these benefits are not well understood

Plots from: Li et al., [Visualizing the Loss Landscape of Neural Nets](#), NeurIPS 2018



(a) without skip connections



(b) with skip connections



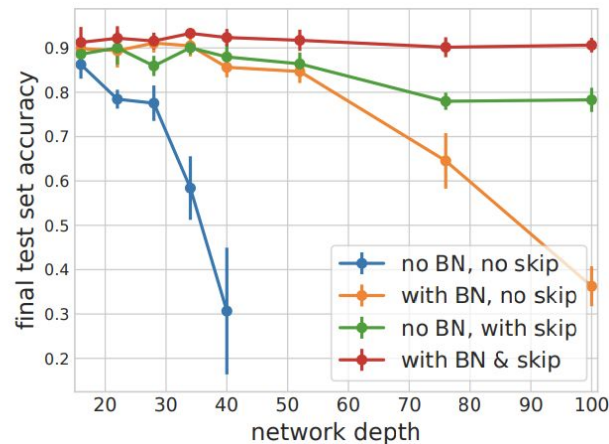
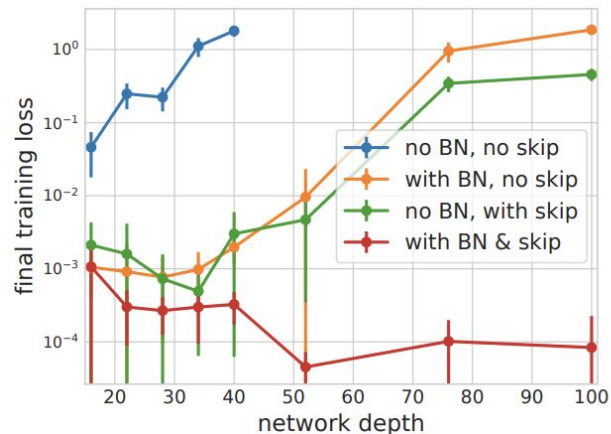
# Batch normalization + Residual networks

Led to a dramatic improvement in performance

- Train significantly deeper networks (1000+ layers)
- Faster convergence
- Better generalization

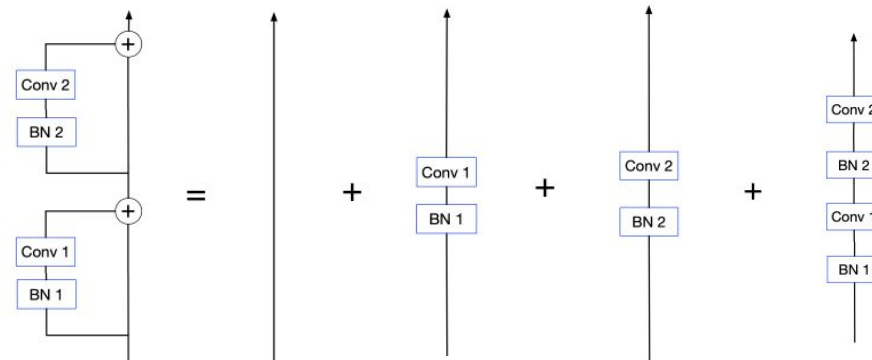
Origins of these benefits are not well understood

Plots from: Sankararaman et al., [The Impact of Neural Network Overparameterization on Gradient Confusion and Stochastic Gradient Descent](#), ICML 2020

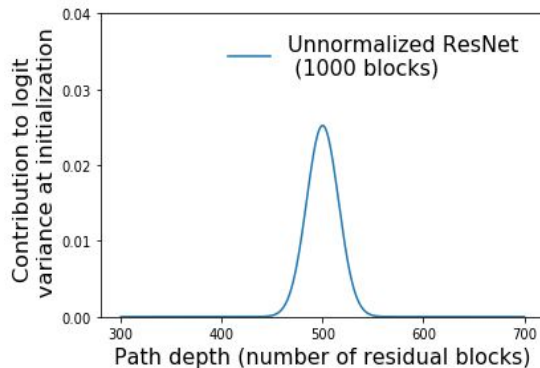


# Intuition from deep linear networks

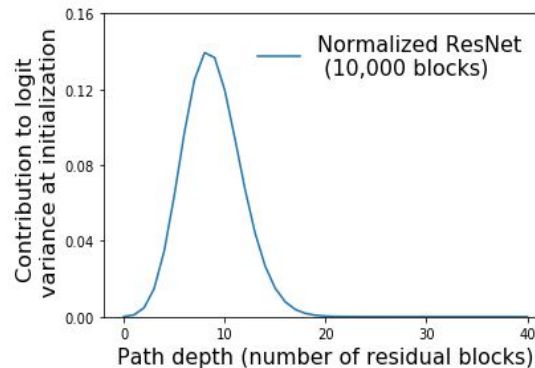
Linear ResNet = ensemble of paths



BatchNorm biases linear ResNets towards shallow paths at initialization



500



~10



# Implications?

All trainable deep networks use one of the following:

- BatchNorm + ResNet
- Orthogonal Initialization + antisymmetric activations (e.g. Tanh)
- Skiplnit (or alternatives like Fixup) + ResNet

Common property: **Initialize layers/residual blocks close to the identity function**



# Implications?

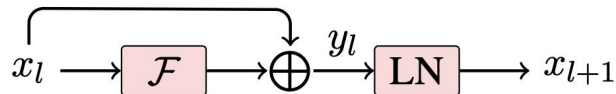
All trainable deep networks use one of the following:

- BatchNorm + ResNet
- Orthogonal Initialization + antisymmetric activations (e.g. Tanh)
- SkipInit (or alternatives like Fixup) + ResNet

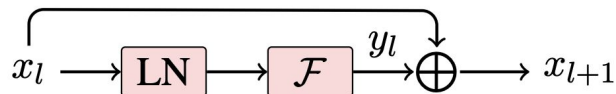
Common property: **Initialize layers/residual blocks close to the identity function**

Idea extends to other normalization schemes:

- E.g. LayerNorm in Transformers should be placed on residual path for easier training



(a) post-norm residual unit



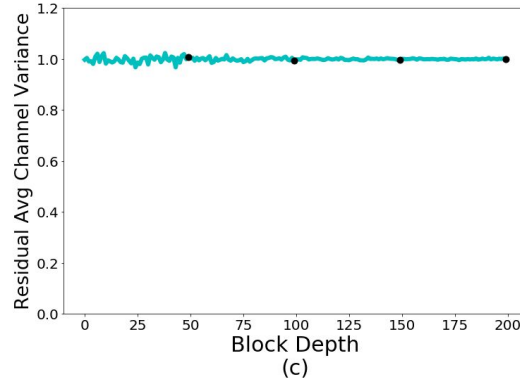
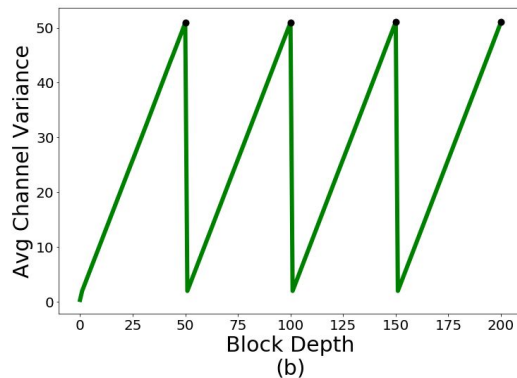
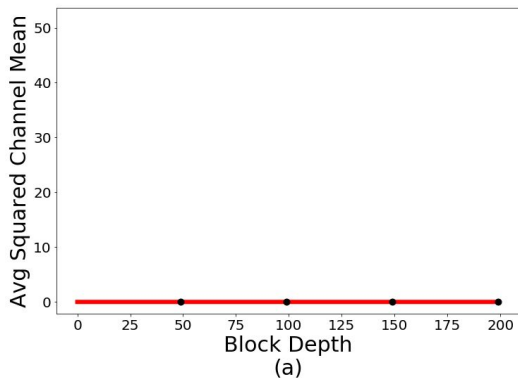
(b) pre-norm residual unit



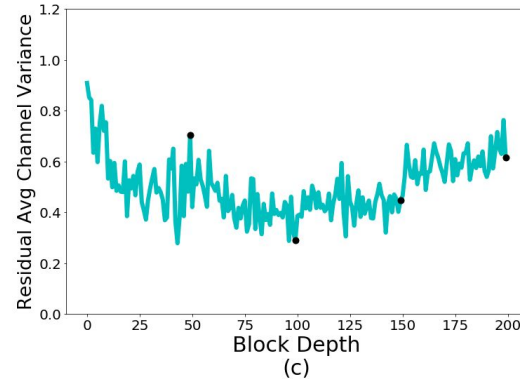
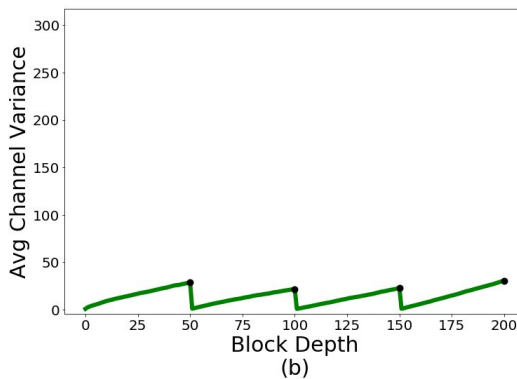
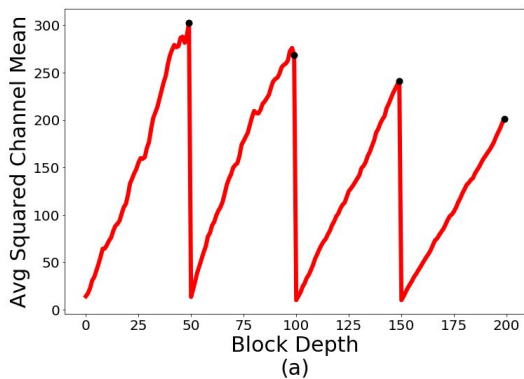
# This mean shift accumulates with depth!

ResNet-600 with 4 transition blocks marked by black dots

**With  
BatchNorm**



**Without  
BatchNorm  
&  
with Variance  
Downscaling**

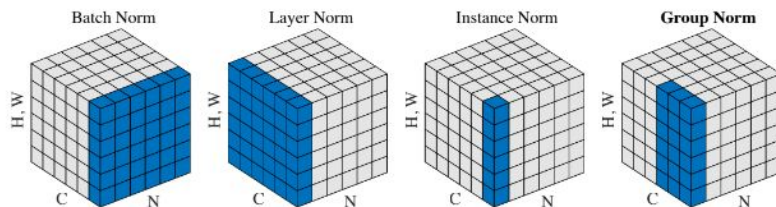


# Related Work

- **GroupNorm + Weight Standardization**

- Noted by Brain Zurich as being a good drop-in replacement for BN
- We find it improves self-supervised performance on SimCLR

## Activation-based Normalizers



## Weight Standardization

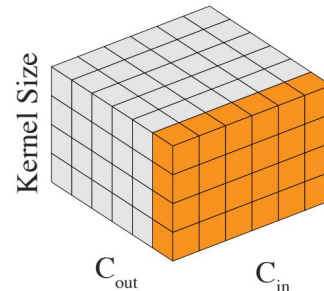


Figure 2. **Normalization methods.** Each subplot shows a feature map tensor, with  $N$  as the batch axis,  $C$  as the channel axis, and  $(H, W)$  as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

### Group Normalization

Yuxin Wu

Kaiming He

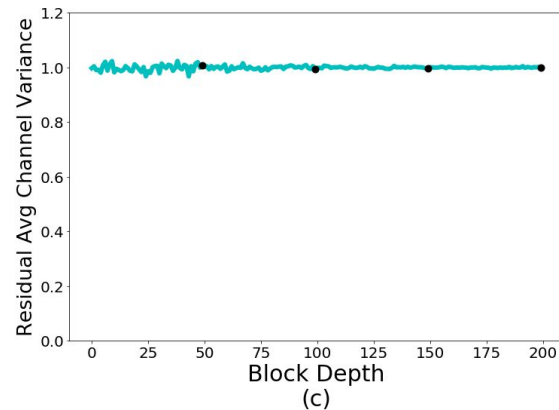
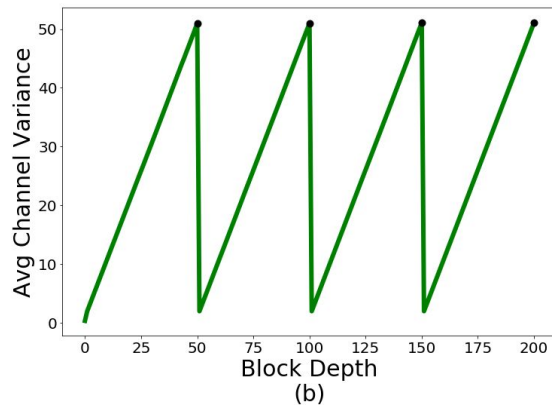
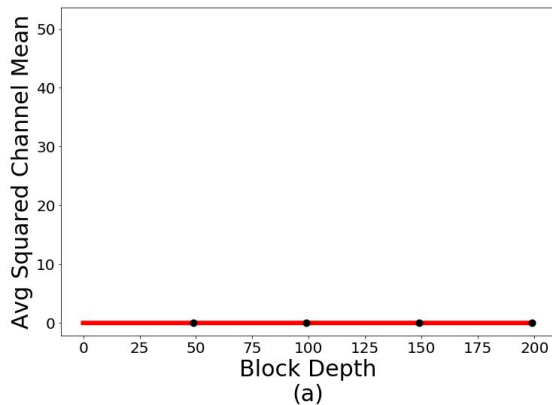
### Micro-Batch Training with Batch-Channel Normalization and Weight Standardization

Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille, *Fellow, IEEE*



# Signal Propagation Plots

- **Directly measuring statistics of hidden activations helps characterize how signals evolve**
  - Allows comparison of different architectures



**SPP for a 600 layer ResNet v2 with ReLU-BN-Conv ordering**



## The Mean Shift Issue

To develop an intuition for this phenomenon, consider the transformation  $z = Wg(x)$ , where  $W$  is arbitrary and fixed, and  $g(\cdot)$  is an activation function that acts component-wise on iid inputs  $x$  such that  $g(x)$  is also iid. Thus,  $g(\cdot)$  can be any popular activation function like ReLU, tanh, SiLU, etc. Let  $\mathbb{E}(g(x_i)) = \mu_g$  and  $\text{Var}(g(x_i)) = \sigma_g^2$  for all dimensions  $i$ . It is straightforward to show that the expected value and the variance of any unit  $i$  of the output  $z_i = \sum_j^N W_{i,j}g(x_j)$  is given by:

$$\mathbb{E}(z_i) = N\mu_g\mu_{W_{i,\cdot}}, \quad \text{and} \quad \text{Var}(z_i) = N\sigma_g^2(\sigma_{W_{i,\cdot}}^2 + \mu_{W_{i,\cdot}}^2), \quad (1)$$

where  $\mu_{W_{i,\cdot}}$  and  $\sigma_{W_{i,\cdot}}$  are the mean and standard deviation of the  $i^{\text{th}}$  row of  $W$ :

$$\mu_{W_{i,\cdot}} = \frac{1}{N} \sum_j^N W_{i,j}, \quad \text{and} \quad \sigma_{W_{i,\cdot}}^2 = \frac{1}{N} \sum_j^N W_{i,j}^2 - \mu_{W_{i,\cdot}}^2. \quad (2)$$





## The Mean Shift Issue

To develop an intuition for this phenomenon, consider the transformation  $z = Wg(x)$ , where  $W$  is arbitrary and fixed, and  $g(\cdot)$  is an activation function that acts component-wise on iid inputs  $x$  such that  $g(x)$  is also iid. Thus,  $g(\cdot)$  can be any popular activation function like ReLU, tanh, SiLU, etc. Let  $\mathbb{E}(g(x_i)) = \mu_g$  and  $\text{Var}(g(x_i)) = \sigma_g^2$  for all dimensions  $i$ . It is straightforward to show that the expected value and the variance of any unit  $i$  of the output  $z_i = \sum_j^N W_{i,j}g(x_j)$  is given by:

$$\mathbb{E}(z_i) = N\mu_g\mu_{W_{i,\cdot}}, \quad \text{and} \quad \text{Var}(z_i) = N\sigma_g^2(\sigma_{W_{i,\cdot}}^2 + \mu_{W_{i,\cdot}}^2), \quad (1)$$

where  $\mu_{W_{i,\cdot}}$  and  $\sigma_{W_{i,\cdot}}$  are the mean and standard deviation of the  $i^{\text{th}}$  row of  $W$ :

$$\mu_{W_{i,\cdot}} = \frac{1}{N} \sum_j^N W_{i,j}, \quad \text{and} \quad \sigma_{W_{i,\cdot}}^2 = \frac{1}{N} \sum_j^N W_{i,j}^2 - \mu_{W_{i,\cdot}}^2. \quad (2)$$

$$\mathbb{E}(|\mu_{W_{i,\cdot}}|) > 0$$



$$\mathbb{E}(z_i^2) = 2\mathbb{E}(\mu_g^2)$$

(He init, any network width)



## The Mean Shift Issue

$$\mathbb{E}(|\mu_{W_i}|) > 0$$

To develop an intuition for this phenomenon, consider the transformation  $z = Wg(x)$ , where  $W$  is arbitrary and fixed, and  $g(\cdot)$  is an activation function that acts component-wise on iid inputs  $x$  such that  $g(x)$  is also iid. Thus,  $g(\cdot)$  can be any popular activation function like ReLU, tanh, SiLU, etc. Let  $\mathbb{E}(g(x_i)) = \mu_g$  and  $\text{Var}(g(x_i)) = \sigma_g^2$  for all dimensions  $i$ . It is straightforward to show that the expected value and the variance of any unit  $i$  of the output  $z_i = \sum_j^N W_{i,j}g(x_j)$  is given by:

$$\mathbb{E}(z_i) = N\mu_g\mu_{W_{i,\cdot}}, \quad \text{and} \quad \text{Var}(z_i) = N\sigma_g^2(\sigma_{W_{i,\cdot}}^2 + \mu_{W_{i,\cdot}}^2), \quad (1)$$

where  $\mu_{W_{i,\cdot}}$  and  $\sigma_{W_{i,\cdot}}$  are the mean and standard deviation of the  $i^{\text{th}}$  row of  $W$ :

$$\mu_{W_{i,\cdot}} = \frac{1}{N} \sum_j^N W_{i,j}, \quad \text{and} \quad \sigma_{W_{i,\cdot}}^2 = \frac{1}{N} \sum_j^N W_{i,j}^2 - \mu_{W_{i,\cdot}}^2. \quad (2)$$

$$\begin{aligned} \mathbb{E}(\mu_{W_i}^2) &= (1/N^2) \sum_j \sum_k \mathbb{E}(W_{ij}W_{ik}) \\ &= (1/N^2) \sum_j \sum_k (2/N)\delta_{jk} \\ &= (2/N^2) \end{aligned}$$



$$\begin{aligned} \mathbb{E}(z_i^2) &= N^2\mathbb{E}(\mu_g^2\mu_{W_i}^2) \\ &= 2\mathbb{E}(\mu_g^2) \end{aligned}$$



# Gradient Clipping

- Often used in language modeling to stabilize training
- Allows training with larger learning rates

Typically performed by constraining the norm of the gradient:

$$G \rightarrow \begin{cases} \lambda \frac{G}{\|G\|} & \text{if } \|G\| > \lambda, \\ G & \text{otherwise.} \end{cases}$$

**But performance is very sensitive to clipping threshold  $\lambda$**

(requires retuning when varying model depth, batch size and learning rate)

