TRAINING BATCHNORM AND ONLY BATCHNORM:
ON THE EXPRESSIVE POWER OF RANDOM FEATURES IN CNNS

ICLR 2021

MIT, FAIR

Recap

- Batch normalization

The following is the batch normalization algorithm proposed by Ioffe & Szegedy (2015).

1. Let $x^{(1)}, ..., x^{(n)}$ be the pre-activations for a particular unit in a neural network for inputs 1 through n in a mini-batch.
2. Let $\mu = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}$
3. Let $\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \mu)^2$
4. The batch-normalized pre-activation $\hat{x}^{(i)} = \gamma \frac{x^{(i)} - \mu}{\sqrt{\sigma^2}} + \beta$ where γ and β are trainable parameters.
5. The activations are $f(\hat{x}^{(i)})$ where f is the activation function.

Related work

- Why BatchNorm works well?
  → several hypotheses and observations
  - BatchNorm reduces internal covariance shift (ICS)
  - BatchNorm makes the optimization landscape smoother
  - BatchNorm decouples optimization of weight magnitude and direction
  - …
- Application
  - Multi-task learning with a shared network backbone and per-task BatchNorm parameters
  - …
- How about freezing random weight? …

Training only BatchNorm

- Common observation: Training without BatchNorm

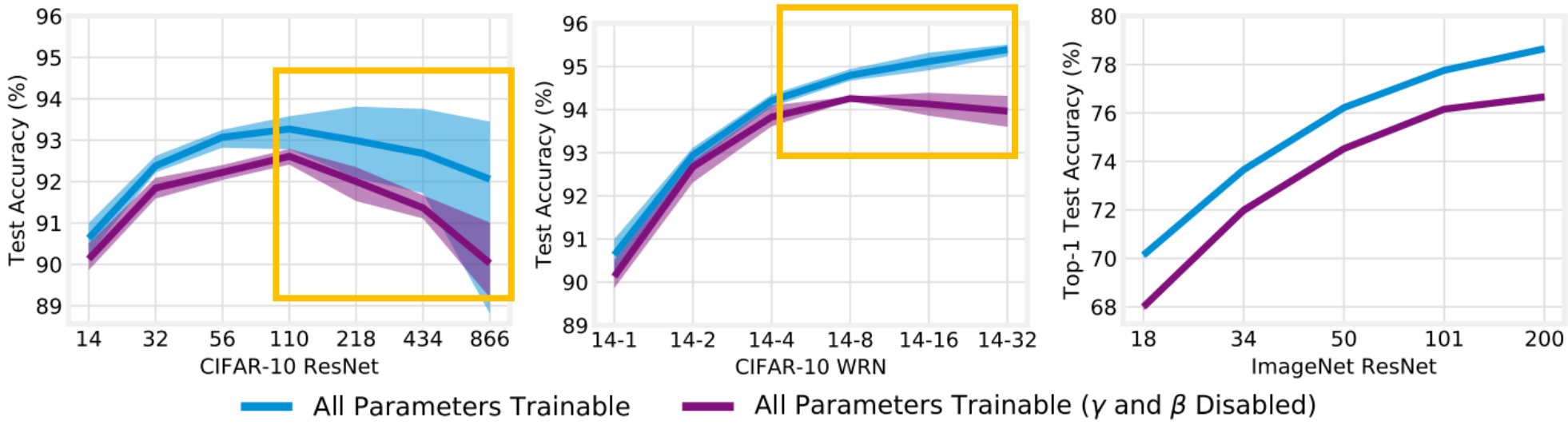


Figure 1: Accuracy when training deep (left) and wide (center) ResNets for CIFAR-10 and deep ResNets for ImageNet (right) as described in Table 1 when all parameters are trainable (blue) and all parameters except γ and β are trainable (purple). Training with γ and β enabled results in accuracy 0.5% to 2% (CIFAR-10) and 2% (ImageNet) higher than with γ and β disabled.

Note: Initial values of γ and β is 1 and 0 (PyTorch)

Training only BatchNorm

- Train ResNets on CIFAR-10 and ImageNet
  - Architecture: Conv-BN-Act
  - Initialization: $\gamma \sim U(0,1)$ and $\beta = 0$ (different from PyTorch)
  - Five runs with CIFAR-10 and three runs with ImageNet
  - Note that γ and β account for 0.64% of all parameters (CIFAR-10), and 0.27% for ImageNet

Family	ResNet for CIFAR-10							Wide ResNet (WRN) for CIFAR-10						ResNet for ImageNet				
Depth	14	32	56	110	218	434	866	14	14	14	14	14	14	18	34	50	101	200
Width Scale	1	1	1	1	1	1	1	1	2	4	8	16	32	1	1	1	1	1
Total	175K	467K	856K	1.73M	3.48M	6.98M	14.0M	175K	696K	2.78M	11.1M	44.3M	177M	11.7M	21.8M	25.6M	44.6M	64.7M
BatchNorm	1.12K	2.46K	4.26K	8.29K	16.4K	32.5K	64.7K	1.12K	2.24K	4.48K	8.96K	17.9K	35.8K	9.6K	17.0K	53.1K	105K	176K
Output	650	650	650	650	650	650	650	650	1.29K	2.57K	5.13K	10.3K	20.5K	513K	513K	2.05M	2.05M	2.05M
Shortcut	2.56K	2.56K	2.56K	2.56K	2.56K	2.56K	2.56K	2.56K	10.2K	41.0K	164K	655K	2.62M	172K	172K	2.77M	2.77M	2.77M
BatchNorm	0.64%	0.53%	0.50%	0.48%	0.47%	0.47%	0.46%	0.64%	0.32%	0.16%	0.08%	0.04%	0.02%	0.08%	0.08%	0.21%	0.24%	0.27%
Output	0.37%	0.14%	0.08%	0.04%	0.02%	0.01%	0.01%	0.37%	0.19%	0.09%	0.05%	0.02%	0.01%	4.39%	2.35%	8.02%	4.60%	3.17%
Shortcut	1.46%	0.55%	0.30%	0.15%	0.07%	0.04%	0.02%	1.46%	1.47%	1.47%	1.48%	1.48%	1.48%	1.47%	0.79%	10.83%	6.22%	4.28%

Table 1: ResNet specifications and parameters in each part of the network. ResNets are called *ResNet-D*, where D is the depth. Wide ResNets are called *WRN-D-W*, where W is the width scale. ResNet-18 and 34 have a different block structure than deeper ImageNet ResNets (He et al., 2015a).

Training only BatchNorm

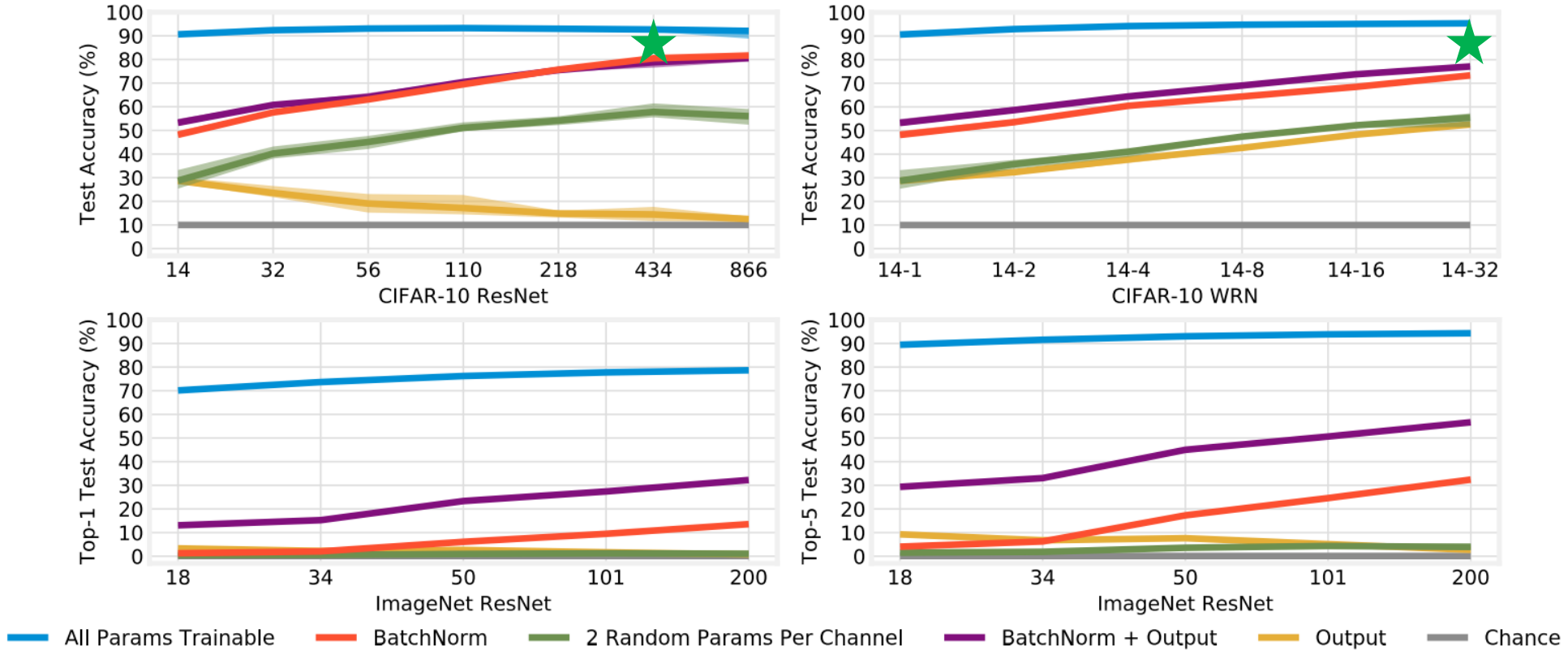


Figure 2: Accuracy of ResNets for CIFAR-10 (top left, deep; top right, wide) and ImageNet (bottom left, top-1 accuracy; bottom right, top-5 accuracy) with different sets of parameters trainable.

★: accuracy is 7%p higher for ResNet-434 than for WRN-14-32
although both have similar numbers of BatchNorm parameters (32.5K vs 35.8K)
→ Is performance always better when increasing depth rather than increasing width?

Training only BatchNorm

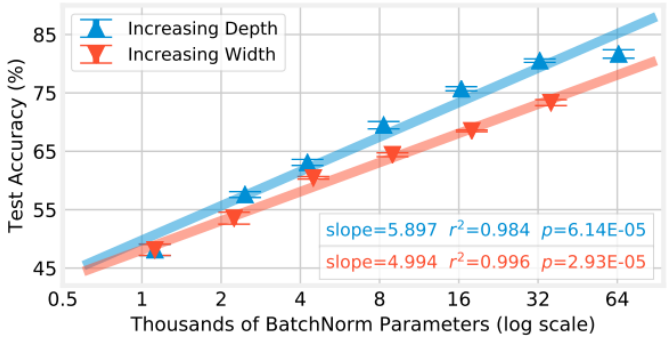


Figure 3: The relationship between BatchNorm parameter count and accuracy when scaling depth and width of CIFAR-10 ResNets.

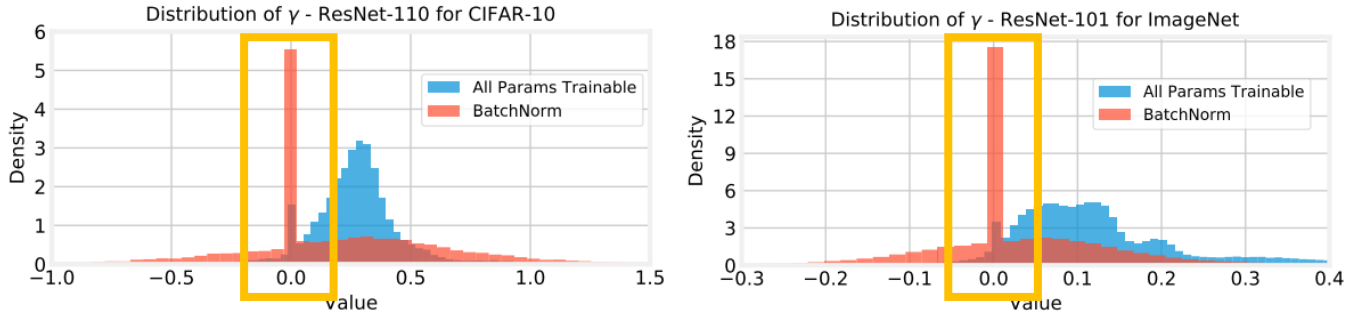


Figure 4: The distribution of γ for ResNet-110 and ResNet-101 aggregated from five (CIFAR-10) or three replicates (ImageNet). Distributions of γ and β for all networks are in Appendix H.

27% (ResNet-110) and 33% (ResNet-101) of all γ values have a magnitude < 0.01

→ BatchNorm seemingly learns to disable needless random features or to increase sparsity

Training only BatchNorm

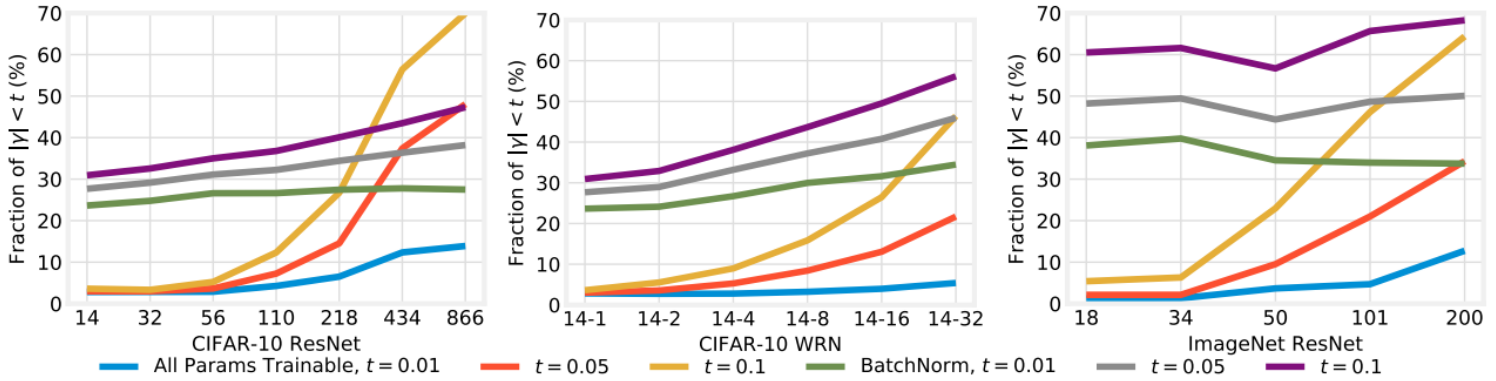


Figure 5: Fraction of γ parameters for which $|\gamma|$ is smaller than various thresholds.

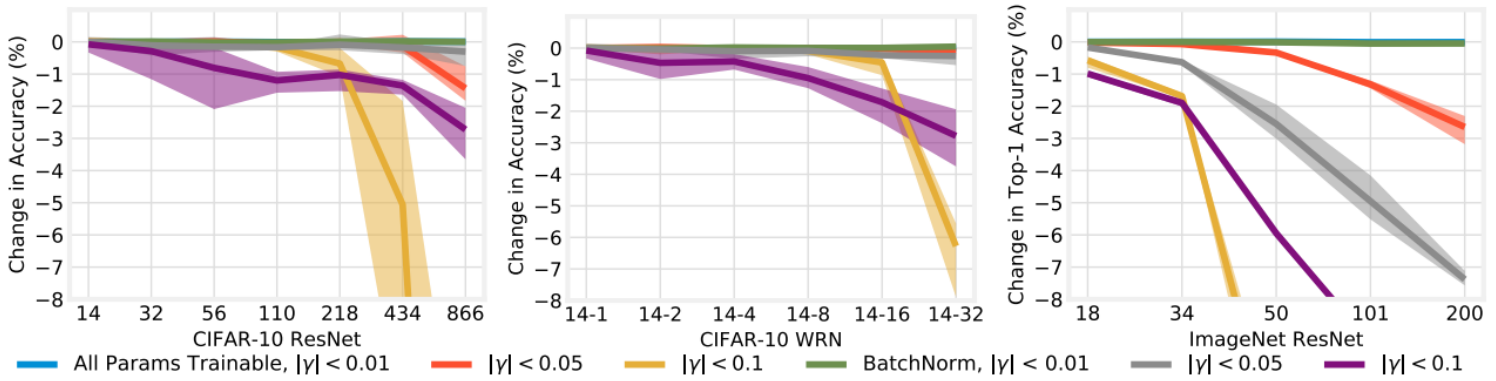


Figure 6: Accuracy change when clamping γ values with $|\gamma|$ below various thresholds to 0.

Sparsity of γ become increase when depth and width increase?
→ Hypothesis: Does BatchNorm prevent exploding activations?

Training only BatchNorm

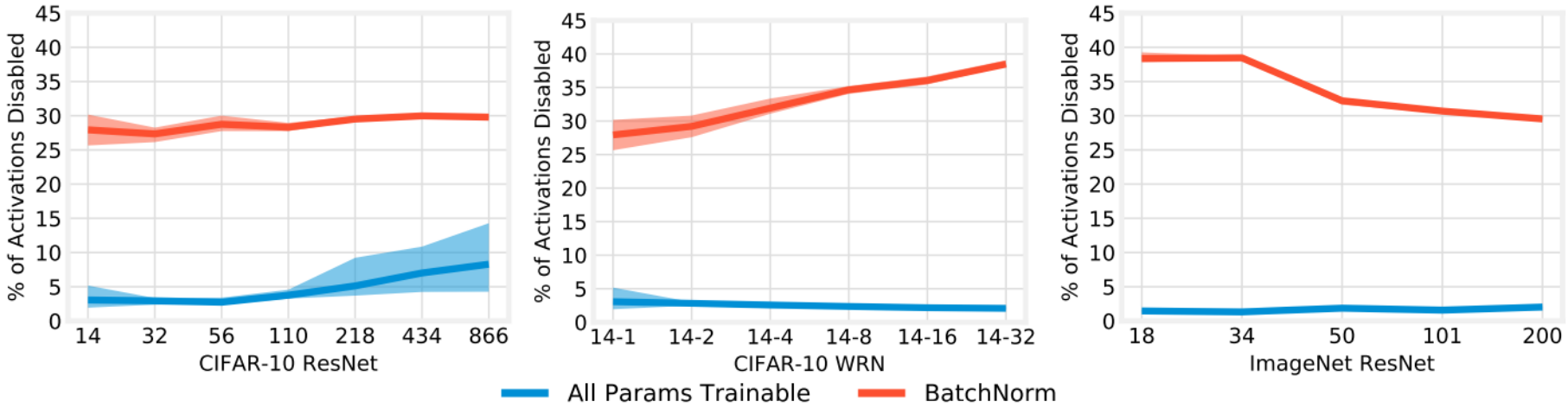
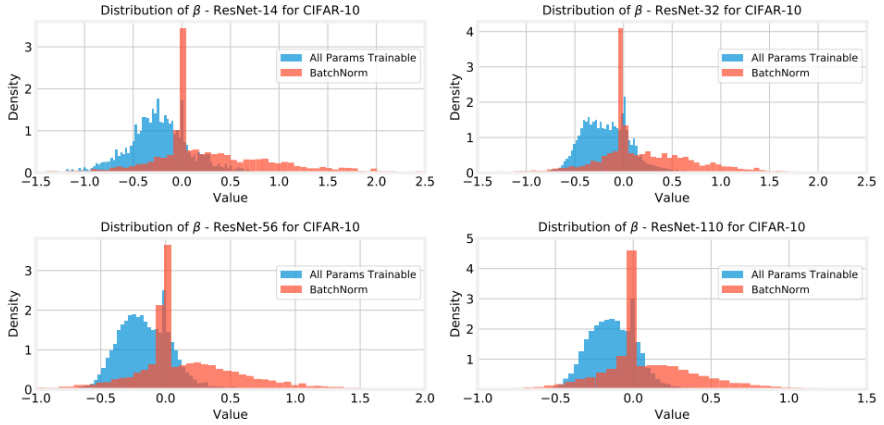


Figure 7: The fraction of ReLU activations for which $\Pr[\text{activation} = 0] > 0.99$.

= ratio of Dead ReLU = sparse activation

How about the effect of β?
→ We need to analyze activation values after ReLU

When training all parameters, activations were not disabled



Discussion and Conclusions

- BatchNorm and affine transformation
- Random features