DiracNets: Training Very Deep Neural Networks Without Skip-Connections

Sergey Zagoruyko
sergey.zagoruyko@enpc.fr
Nikos Komodakis
nikos.komodakis@enpc.fr

Université Paris-Est, École des Ponts ParisTech
Paris, France

Abstract

Deep neural networks with skip-connections, such as ResNet, show excellent performance in various image classification benchmarks. It is though observed that the initial motivation behind them - training deeper networks - does not actually hold true, and the benefits come from increased capacity, rather than from depth. Motivated by this, and inspired from ResNet, we propose a simple Dirac weight parameterization, which allows us to train very deep plain networks without skip-connections, and achieve nearly the same performance. This parameterization has a minor computational cost at training time and no cost at all at inference. We're able to achieve 95.5% accuracy on CIFAR-10 with 34-layer deep plain network, surpassing 1001-layer deep ResNet, and approaching Wide ResNet. Our parameterization also mostly eliminates the need of careful initialization in residual and non-residual networks. The code and models for our experiments are available at https://github.com/szagoruyko/diracnets

1 Introduction

There were many attempts of training very deep networks. In image classification, after the success of AlexNet [11] with 8 layers, the major improvement was brought by VGG [18] (16-19 layers) and later by Inception [21] (22 layers). In recurrent neural networks, LSTM [9] allowed training deeper networks by introducing gated memory cells, leading to major increase of parameter capacity and network performance. Recently, similar idea was applied to image classification, proposing Highway Networks [19], later improved by Residual Networks [6], resulting in simple architecture with skip-connections, which was shown to be generalizable to many other tasks. There were also proposed several other ways of adding skip-connections, such as DenseNet [10], which passed all previous activations to each new layer.

Despite the success of ResNet, a number of recent works showed that the original motivation of training deeper networks does not actually hold true, *e.g.* it might just be an ensemble of shallower networks [22], and ResNet widening is more effective that deepening [23], meaning that there's no benefit from increasing depth to more than 50 layers. It is also known that deeper networks can be more efficient than shallower and wider, so various methods were proposed to train deeper networks, such as well-designed initialization strategies and special nonlinearities [2, 5, 7], additional mid-network losses [12], better optimizers [20], knowledge transfer [4, 14] and layer-wise training [16].

To summarize, deep networks with skip-connections have the following problems:

- Feature reuse problem: upper layers might not learn useful representations given previous activations;
- Widening is more effective than deepening: there's no benefit from increasing depth;
- Actual depth is not clear: it might be determined by the shortest path.

But, the features learned by such networks are generic, and they are able to train with massive number of parameters without negative effects of overfitting. We're thus interested in better understanding of networks with skip-connections, which would allow us to train very deep *plain* (without skip-connections) networks and benefits they could bring, such as decreased number of parameters, better generalization, and improved computational efficiency.

Motivated by this, we propose a novel weight parameterization for neural networks, which we call Dirac parameterization, applicable to a wide range of network architectures. Furthermore, by use of the above parameterization, we propose novel plain VGG and ResNet-like architecture without skip-connections, which we call DiracNet. These networks are able to train with hundreds of layers, surpass 1001-layer ResNet while having only 34-layers, and approach Wide ResNet (WRN) accuracy. We should note that we train DiracNets end-to-end, without any need of layer-wise pretraining. We believe that our work is an important step towards simpler and more efficient deep neural networks.

Overall, our contributions are the following:

- We propose generic Dirac weight parametrization, applicable to a wide range of neural network architectures;
- Our plain Dirac parameterized networks are able to train end-to-end with hundreds of layers, and achieve nearly state-of-the-art Wide ResNet (WRN) accuracy. Furthermore, they are able to train with massive number of parameters and still generalize well without negative effects of overfitting;
- Plain deeper DiracNets with less than 50 layers need much less parameters than corresponding wider networks with the same accuracy; performance of deeper than 50 layer networks slowly decreases.
- Dirac parameterization can be used in combination with skip-connections like ResNet, in which case it eliminates the need of careful initialization.
- In a trained network Dirac-parameterized filters can be folded into a single tensor, resulting in a simple and easily interpretable VGG-like network

2 Dirac parameterization

Inspired from ResNet, we parametrize weights as a residual of Dirac function, instead of adding explicit skip connection. Because convolving any input with Dirac results in the same input, this helps propagate information deeper in the network. Similarly, on backpropagation it helps alleviate vanishing gradients problem.

Let δ be the identity in algebra of discrete convolutional operators, i.e. convolving it with input x results in the same output x ($\odot$ denotes convolution):

$$\delta \odot x = x \tag{1}$$

In two-dimensional case, when convolution might be expressed as matrix multiplication, δ is simply an identity matrix I (or a Kronecker delta). We generalize this operator to the case of a convolutional layer, where an input tensor $x \in \mathbb{R}^{M,N_1,N_2,\ldots,N_L}$ (that consists of M channels of spatial dimensions (N_1, N_2, ..., N_L)) is convolved with a weight tensor $\hat{W} \in \mathbb{R}^{M,M,K_1,K_2,\ldots,K_L}$ (combining M filters[1]) to produce an output tensor y of M channels, i.e. $y = \hat{W} \odot x$ (here we assume that the number of input channels and output channels are the same, and generalize to other shapes in section 2.1). In this case we define Dirac $\delta \in \mathbb{R}^{M,M,K_1,K_2,\ldots,K_L}$ as the following weight tensor that preserves eq. 1:

$$\delta(i,j,l_1,l_2,\ldots,l_L) = \begin{cases} 1 & \text{if } i = j \text{ and } l_m = \lfloor \frac{K_m}{2} \rfloor \text{ for } m = 1..L, \\ 0 & \text{otherwise;} \end{cases} \tag{2}$$

Given the above definition, for a convolutional layer $y = \hat{W} \odot x$ we propose the following parameterization for the weight tensor $\hat{W}$ (hereafter we omit bias for simplicity):

$$\begin{aligned} y &= \hat{W} \odot x, \\ \hat{W} &= a\delta + W, \end{aligned} \tag{3}$$

where a is scalar parameter learned during training, and W is a weight tensor. When $a = 0$ it reduces to a simple linear layer $W \odot x$. When $a > 1$ and W is small, Dirac dominates, and the output is close to be the same as input. During training, we initialize a to a higher value than 1 and add weight decay regularization on a, forcing W to contribute at later stages of training, when a drops to smaller values.

We also use weight normalization [15] for W, which we find useful for stabilizing training of very deep networks with more than 100 layers:

$$\hat{W} = a\delta + bW_{norm}, \tag{4}$$

where b is another scalar parameter (to be learned during training), and W_{norm} is a normalized weight tensor where each filter v is normalized by it's Euclidean norm:

$$v = \frac{v}{\max(\|v\|, \varepsilon)} \tag{5}$$

We initialize b to a small number, *e.g.* 10^{-3}, W from normal distribution $\mathcal{N}(0,1)$, and ε is a small constant to avoid division by zero. Gradients of (4) can be easily calculated via chain-rule. We rely on automatic differentiation, available in all major modern deep learning frameworks (PyTorch, Tensorflow, Theano), to implement it.

Overall, this adds a negligible number of parameters to the network (just a single scalar a and b per layer). We note the importance of using weight decay on all parameters, including a and b. Given targets z, the full loss function is:

$$L(W,a,b,x,z) = L_{orig}(W,a,b,x,z) + \gamma_W \|W\|_2^2 + \gamma_b \|b\|_2^2 + \gamma_a \|a\|_2^2 \tag{6}$$

In practice, we set $\gamma_W = \gamma_b = \gamma_a$.

[1]outputs are over the first dimension of $\hat{W}$, inputs are over the second dimension of $\hat{W}$

2.1 Generalization to non-square kernel shapes

In this section we explain how to apply delta parameterization to other convolutional filter shapes than square. We limit numbers of input channels and output channels N_i and N_o to be a multiple of each other $N_i = mN_o$ or $N_o = mN_i$, where $m \in \mathbb{N}$ is a natural number. First, we generate δ for input with equal numbers of input and output channels : $\min(N_i, N_o)$, and repeat it m times in either dimension to match the shape (we provide an example in fig. 1).

Let's first consider the case when number of input channels is bigger, or $N_i = mN_o$. Dirac then is a block tensor of "square" δ tensors (where each δ-tensor has N_o input and N_o output channels). Given blocks x_i ($i \in \{1 \ldots m\}$) of input tensor x and corresponding filter tensors W_i ($i \in \{1 \ldots m\}$), the output is defined as a sum of per-block convolutions:

$$y = \begin{pmatrix} a\delta + W_1 \\ a\delta + W_2 \\ \ldots \\ a\delta + W_m \end{pmatrix}^{\intercal} \odot \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_m \end{pmatrix} = \sum_{i=1}^{m} (a\delta + W_i) \odot x_i \tag{7}$$

In the second case when the number of output channels is bigger (e.g. $m = 2$ in fig. 1), the output is a block matrix, where vertical blocks are convolutions with input and square filter blocks $a\delta + W_i$:

$$y = \begin{pmatrix} (a\delta + W_1) \odot x \\ (a\delta + W_2) \odot x \\ \ldots \\ (a\delta + W_m) \odot x \end{pmatrix} \tag{8}$$

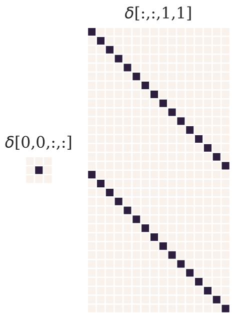


Figure 1: Slices of 3×3 filter size, 16 input and 32 output δ-tensor.

2.2 Negative CReLU

CReLU [17] was initially proposed as an alternative to ReLU as nonlinearity without loss of information, to improve reconstruction of inputs given activations in hidden layers. It was also found to be improving classification accuracy. LL-init, proposed in [1], combines CReLU and weights initialized with block-mirror structure, and aims to keep layers linear in the beginning. In our case, block-mirror structure, combined with delta-parameterization and small weights, becomes no longer necessary, and reduces to a modification in nonlinearity, which we call negative CReLU (NCReLU):

$$\text{NCReLU}(x) = \begin{pmatrix} \rho(x) \\ -\rho(-x) \end{pmatrix}, \tag{9}$$

a simple concatenation of non-negative and non-positive activations. As CReLU, it requires twice the number of filter input channels than ReLU, does not cause loss of information, so has the same reconstruction property. Combining NCReLU with Dirac parameterization (7) results in:

$$y = \hat{W} \odot \begin{pmatrix} \rho(x) \\ -\rho(-x) \end{pmatrix} = (a\delta + W_1) \odot \rho(x) - (a\delta + W_2) \odot \rho(-x), \tag{10}$$

where W_1 and W_2 are blocks of weight tensor W. Initially, $W \approx 0$ and NCReLU output is:

$$y \approx a\delta \odot \rho(x) - a\delta \odot \rho(-x) = ax, \tag{11}$$

which is a simple scalar scaling layer. As we have batch normalization layer after, scaling doesn't hurt information propagation in the network.

2.3 Connection to ResNet

Let us discuss the connection of Dirac parameterization to ResNet. Due to distributivity of convolution, eq. (3) can be rewritten to show that the skip-connection in Dirac parameterization is implicit:

$$y = \sigma\big((a\delta + W) \odot x\big) = \sigma\big(ax + W \odot x\big), \tag{12}$$

where $\sigma(x)$ is a function combining nonlinearity and batch normalization. The skip connection in ResNet is explicit:

$$y = x + \sigma(W \odot x) \tag{13}$$

This means that Dirac parameterization and ResNet differ only by the order of nonlinearities. Each delta parameterized layer adds complexity by having unavoidable nonlinearity, which is not the case for ResNet. Additionally, Dirac parameterization can be folded into a single weight tensor on inference.

3 Experimental results

We adopt architecture similar to ResNet and VGG, and instead of skip-connections use Dirac parameterization (see table 1). The architecture consists of three groups, where each group has $2N$ convolutional layers ($2N$ is used for easier comparison with basic-block ResNet and WRN, which have N blocks of pairs of convolutional layers per group). For simplicity we use max-pooling between groups to reduce spatial resolution. We also define width k as in WRN to control number of parameters. Due to NCReLU at given depth and width our architecture has $\sqrt{2}$ more parameters than WRN. VGG is a special case of our architecture.

We chose CIFAR and ImageNet for our experiments. As for baselines, we chose Wide ResNet with identity mapping in residual block [8] and basic block (two 3×3 convolutions per block). We used the same training hyperparameters as WRN [23] for both CIFAR and ImageNet.

The experimental section is composed as follows. First, we provide a detailed experimental comparison between plain and plain-Dirac networks, and compare them with ResNet and WRN on CIFAR. Then, we show the importance of various ingredients of our parameterization: scaling parameter a, NCReLU and weight normalization, and present the results on ImageNet. Lastly, we apply Dirac parametrization to ResNet and show that it eliminates the need of careful initialization.

name	output size	layer type
conv1	32×32	[3×3, 16]
group1	32×32	$[3\times3,\ 32k \times 16k]\times 2N$
max-pool	16×16	
group2	16×16	$[3\times3,\ 64k \times 32k]\times 2N$
max-pool	8×8	
group3	8×8	$[3\times3,\ 128k \times 64k]\times 2N$
avg-pool	1×1	$[8 \times 8]$

Table 1: Structure of DiracNets. Network width is determined by factor k. Groups of convolutions are shown in brackets as [kernel shape, number of input channels, number of output channels] where $2N$ is a number of layers in a group. Final classification layer and dimensionality changing layers are omitted for clearance.

3.1 Plain networks with Dirac parameterization

In this section we compare plain networks with plain DiracNets. To do that, we trained both with 10-52 layers and the same number of parameters at the same depth, keeping NCReLU nonlinearity (fig. 2). As expected, at 10 and 16 layers there's no difficulty in training plain networks, and both plain and plain-Dirac networks achieve the same accuracy. After that, accuracy of plain networks quickly drops, and with 52 layers only achieves 88%, whereas for Dirac parametrized networks it keeps growing until 34 layers, and then slowly drops. DiracNet with 34 layers achieves 92.8% validation accuracy, whereas simple plain only 91.2%. Plain 100-layer network does not converge and only achieves 40% train/validation accuracy, whereas DiracNet achieves 92.4% validation accuracy. We are able to train DiracNet even with 400 layers and achieve 91.6% accuracy, about 1% less than DiracNet-34, while 400-layer plain network without Dirac parameterization does not even start converging.

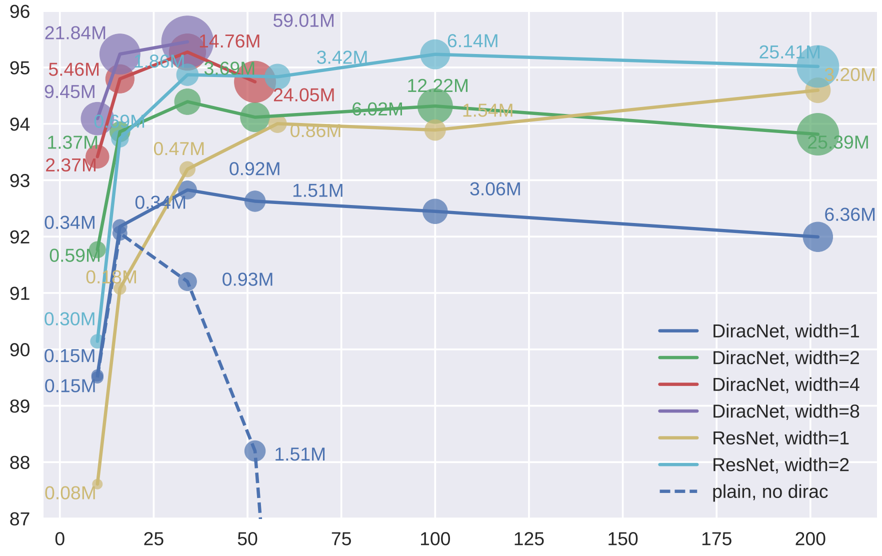


Figure 2: DiracNet and ResNet with different depth/width, each circle area is proportional to number of parameters. Plain networks without Dirac parameterization have the same architecture as DiracNet and have NCReLU nonlinearity. Note that both ResNet and DiracNet after 34-50 layers give either small or no benefit from increased depth.

3.2 Plain Dirac networks and residual networks

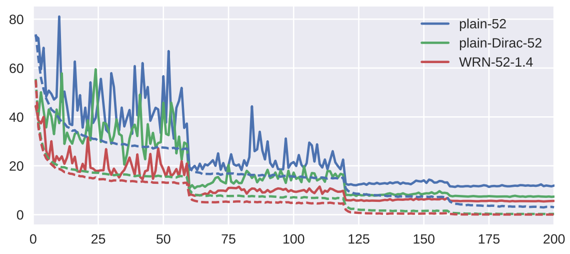
To compare plain Dirac parameterized networks with Wide ResNet we trained them with different width k from 1 to 4 and depth from 10 to 200 (fig. 2). As observed by WRN authors [23], accuracy of ResNet is mainly determined by the number of parameters, and we even notice that wider networks achieve better performance than deeper. DiracNets, however, benefit from depth, and deeper networks with the same accuracy as wider have less parameters. In general, DiracNets need more parameters than ResNet to achieve top accuracy, and we were able to achieve 95.5% with DiracNet-34-8 with 59M parameters, which is very close to WRN-28-10 with 96.0% and 36M parameters. We don't observe validation accu-

	depth-width	# params	CIFAR-10	CIFAR-100
NIN [13]			8.81	35.67
DSN [12]			8.22	34.57
FitNet [14]			8.39	35.04
ELU [5]			6.55	24.28
DiracNet (ours)	34-2	3.7M	5.6	26.72
	34-8	59M	**4.54**	**20.96**
ResNet[8]	1001-1	10.2M	4.92	22.71
WRN [23]	28-10	36.5M	**4.00**	**19.25**

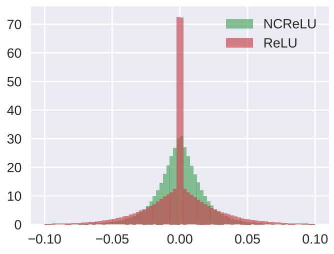
Table 2: CIFAR performance of plain (top part) and residual (bottom part) networks on with horizontal flips and crops data augmentation. DiracNets outperform all other plain networks by a large margin, and approach residual architectures. No dropout it used.

racy degradation when increasing width, the networks still perform well despite the massive number of parameters, just like WRN. Interestingly, plain DiracNet with only 34 layers is able to outperform original ResNet with 1001 layers (table 2)

We also visualize training/validation error of plain network without Dirac parameterization, WRN and plain DiracNet at fig. 3(a) with the same depth and number of parameters. Dirac parameterization greatly improves convergence of plain networks, but ResNet has better final accuracy. Interestingly, during training we observe low oscillating validation accuracy at high learning rates, despite the high training accuracy. The oscillation is higher at deeper networks. This is likely due to quickly changing bottom of the network. The effect vanishes if we use adaptive learning rate algorithms such as Adam, but we observe that in general these algorithms estimate lower learning rate than SGD and tend to fall to the closest local minima, converging to significantly worse accuracy.



(a)



(b)

Figure 3: **a:** Convergence of a plain network without Dirac parameterization, WRN and plain DiracNet with the same number of parameters. Plain network without Dirac parameterization has the same architecture as DiracNet and has NCReLU. **b:** Normalized histograms of all weights in a plain Dirac parameterized network with ReLU and NCReLU activations. ReLU makes weights sparser.

3.3 Importance of scaling

We plot how a changes during training (fig. 4). We initialize a for all layers with $a_0 = 5$, such that δ dominates and all layers are in linear zone. Then, due to weight decay on a, it decreases and the network is forced to learn better representations. We also tried to remove weight decay from a, and initialize from $a_0 = 1$, then it quickly grows to values around 5, and slowly decreases after the first learning rate drop. This has slightly worse final accuracy than keeping weight decay and $a_0 = 5$.

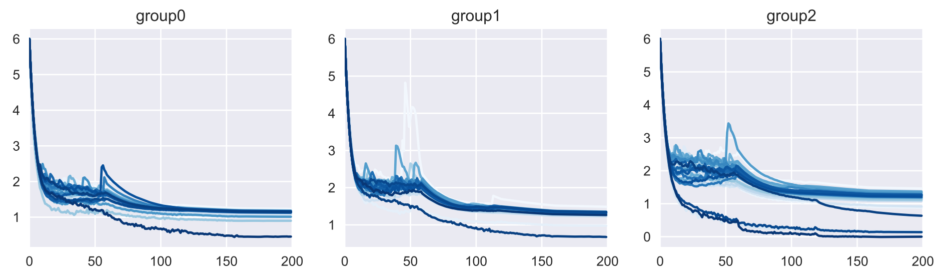


Figure 4: Dynamics of a values changing during training per group in DiracNet-100. Layers higher in a group have deeper colors. Initially $a = 5$.

3.4 Importance of NCReLU

We experimentally find that NCReLU in DiracNet works better than ReLU for deeper networks. To investigate, we visualize slices of weights of DiracNet with trained with NCReLU and with ReLU (fig. 5) and notice that ReLU weights are a lot sparser, and some rows contain only zeros. We plot distributions of both in fig. 3(b). Deeper networks without NCReLU and weight normalization in (4) diverge in the middle of training, likely due to instabilities caused by this.

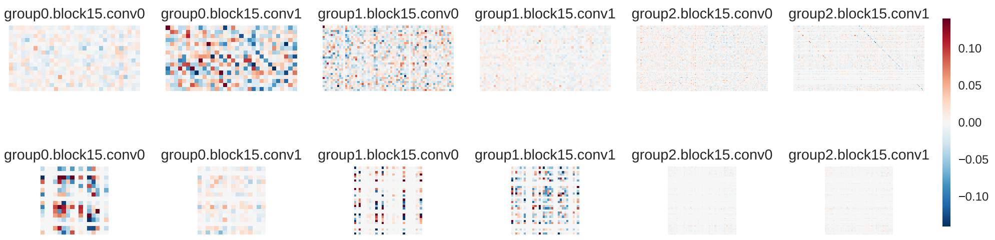


Figure 5: Central slices of subsets of weights of DiracNet-100 with NCReLU (top row) and ReLU (bottom row). ReLU network weights have many more zeros, and even full rows vanish.

3.5 ImageNet results

We trained DiracNets with 18 and 34 layers of different width on ILSVRC2012 dataset. We used the same setup as for ResNet training, and kept the same number of blocks per groups. To compare with ResNet-18, we trained DiracNet-18-0.75, which due to NCReLU has to have $\sqrt{2}$ more parameters than ResNet at the same depth, so we reduce width to have comparable number of parameters. As in CIFAR, DiracNet is very close to ResNet, but doesn't reach it's accuracy with the same number of parameters (fig. 6). Deeper DiracNet-

	Network	# parameters	top-1 error	top-5 error
plain	VGG-CNN-S [3]	102.9M	36.94	15.40
	VGG-16 [18]	138.4M	29.38	-
	DiracNet-18-0.75	12.8M	32.30	12.20
	DiracNet-34-0.5	15.5M	32.70	12.10
	DiracNet-34-0.75	34.8M	28.90	10.00
residual	ResNet-18 [6]	11.7M	30.50	10.40
	ResNet-34 [6]	21.8M	26.70	8.70

Table 3: Single crop top-1 and top-5 error on ILSVRC2012 validation set for plain (top) and residual (bottom) networks.

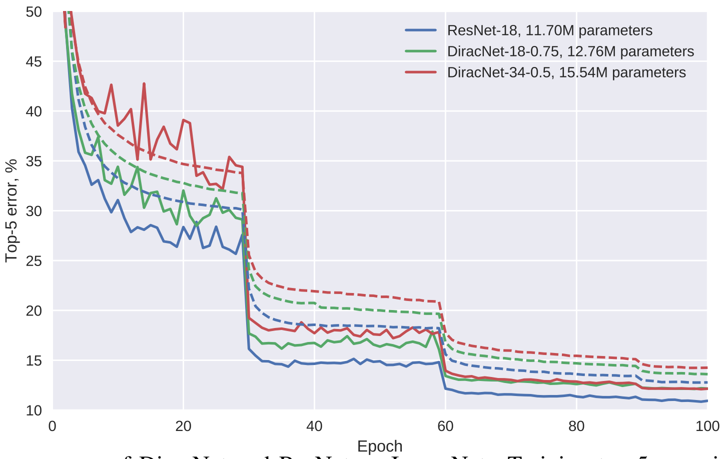


Figure 6: Convergence of DiracNet and ResNet on ImageNet. Training top-5 error is shown with dashed lines, validation - with solid. All networks are trained using the same optimization hyperparameters.

34-0.75 achieves much better accuracy than VGG-CNN-S, and is very close to also plain VGG-16, having only a fraction of it's parameters (table 3).

We also trained deeper and thinner DiracNet-34-0.5 (fig. 6) with the number of parameters comparable to DiracNet-18-0.75 (15.5M to 12.8M). Unlike on CIFAR, where deeper networks need less parameters to achieve the same accuracy, these networks have the same validation accuracy on ImageNet. However, training accuracy of DiracNet-34-0.5 is lower, suggesting that depth in DiracNet has a regularization effect, unlike in ResNet, so it would be interesting to see how DiracNets perform with many more parameters.

3.6 Dirac parameterization for ResNet weight initialization

As expected, Dirac parameterization does not bring accuracy improvements to ResNet on CIFAR, although eliminates the need of initialization. To test that, instead of usually used MSRA init [7], we parameterize weights as:

$$\hat{W} = \delta + W, \tag{14}$$

omitting other terms of eq. 4 for simplicity, and initialize all weights from a normal distribution $\mathcal{N}(0, \sigma^2)$, ignoring filter shapes. Then, we vary σ and observe that ResNet-28 converges to the same validation accuracy with statistically insignificant deviations, even for very small values of σ such as 10^{-8}, and only gives slightly worse results when σ is around 1. It does not converge when all weights are zeros, as expected. Additionally, we tried to use the same orthogonal initialization as for DiracNet and vary it's scaling, in which case the range of the scaling gain is even wider.

4 Discussion

We presented Dirac-parametrized networks, a simple and efficient way to train very deep networks with nearly state-of-the-art accuracy. Even though they are able to successfully train with hundreds of layers, after a certain number of layers there seems to be very small or no benefit in terms of accuracy for both ResNets and DiracNets. This is likely caused by underuse of parameters in deeper layers, and both architectures are prone to this issue to a different extent.

We also observe that DiracNets share the same property as WRN to train with massive number of parameters and still generalize well without negative effects of overfitting, which was initially thought was due to residual connections. We now hypothesize that it is due to a combination of SGD with momentum at high learning rate, which has a lot of noise, and stabilizing factors, such as residual or Dirac parameterization, batch normalization, etc.

References

[1] David Balduzzi, Marcus Frean, Lennox Leary, J. P. Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? *CoRR*, abs/1702.08591, 2017.

[2] Yoshua Bengio and Xavier Glorot. Understanding the difficulty of training deep feed-forward neural networks. In *Proceedings of AISTATS 2010*, volume 9, pages 249–256, May 2010.

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[4] T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representation*, 2016.

[5] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory, 1997.

[10] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[11] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[12] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-Supervised Nets. 2014.

[13] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[14] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. Technical Report Arxiv report 1412.6550, arXiv, 2014.

[15] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Neural Information Processing Systems 2016*, 2016.

[16] J. Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[17] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. *CoRR*, abs/1603.05201, 2016.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[19] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

[20] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1139–1147. JMLR Workshop and Conference Proceedings, May 2013.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[22] Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks are exponential ensembles of relatively shallow networks. *CoRR*, abs/1605.06431, 2016.

[23] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.