

Weight Parameterizations in Deep Neural Networks

Sergey Zagoruyko
Université Paris-Est, École des Ponts ParisTech

December 26, 2017

Outline

1. Motivation
2. Wide residual parameterizations
3. Dirac parameterizations
4. Symmetric parameterizations

Motivation

What changed in how we train deep neural networks since ImageNet?

- Optimization: SGD with momentum [Polyak, 1964] is still the most effective training method
- Regularization: still use basic l_2 -regularization
- Loss: still use softmax for classification
- Architecture: have **batch normalization** and **skip-connections**

Weight parameterization changed!

Single hidden layer MLP:

$$\mathbf{o} = \sigma(\mathbf{W}_1 \odot \mathbf{x}),$$
$$\mathbf{y} = \mathbf{W}_2 \odot \mathbf{o}$$

where \odot denotes linear operation, $\sigma(\mathbf{x})$ - nonlinearity.

Given enough neurons in hidden layer \mathbf{W}_1 MLP can approximate any function [Cybenko, 1989]. However:

- Empirically, deeper networks (2-3 hidden layers) are easier to train [Ba and Caruana, 2014]
- Suffer from overfitting, need regularization, e.g. weight decay, dropout, etc.
- Deeper networks suffer from vanishing/exploding gradients

Improvement #1

Batch Normalization

Reparameterize each layer as:

$$\hat{\mathbf{x}}^{(k)} = \frac{\mathbf{x}^{(k)} - \mathbb{E}[\mathbf{x}^{(k)}]}{\sqrt{\text{Var}[\mathbf{x}^{(k)}]}} \gamma^{(k)} + \beta^{(k)} \text{ for each feature plane } k,$$
$$\mathbf{o} = \sigma(\mathbf{W} \odot \hat{\mathbf{x}})$$

- + Alleviates vanishing/exploding gradients problem (dozens of layers), **does not solve it**
- + Trained networks generalize better (greatly increased capacity)
- + γ and β can be folded into weights at test time
- Weight decay loses its importance
- Struggles to work if samples are highly correlated (RL, RNN)

Improvement #2

skip connections - Highway / ResNet / DenseNet

Instead of single layer:

$$\mathbf{o} = \sigma(\mathbf{W} \odot \mathbf{x}) \quad (1)$$

Residual layer [He et al., 2015]:

$$\mathbf{o} = \mathbf{x} + \sigma(\mathbf{W} \odot \mathbf{x}) \quad (2)$$

- + Further alleviates vanishing gradients (thousands of layers), **does not solve it**
- **No improvement from depth:** - it comes from further increased capacity
 - Batch norm is essential

To summarize, deep residual networks:

- able to train with thousands of layers
- + simplify training
- + achieve state-of-the-art results in many tasks
- have diminishing feature reuse problem
- improving accuracy by a small fraction doubles computational cost

Wide residual parameterizations

1. Motivation
2. Wide residual parameterizations
3. Dirac parameterizations
4. Symmetric parameterizations

Wide Residual Networks,
Zagoruyko&Komodakis, in BMVC 2016

Can we answer these questions:

- is extreme depth important? does it saturate?
- how important is width? can we grow width instead?

Residual parameterization

Instead of single layer:

$$\mathbf{x}_{n+1} = \sigma(\mathbf{W} \odot \mathbf{x}_n)$$

Residual layer [He et al., 2015]:

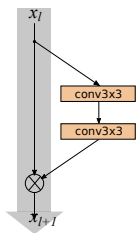
$$\mathbf{x}_{n+1} = \mathbf{x} + \sigma(\mathbf{W} \odot \mathbf{x}_n)$$

“basic” residual block:

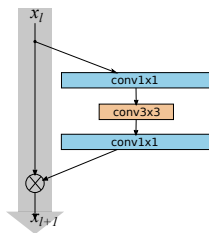
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \sigma(\mathbf{W}_2 \odot \sigma(\mathbf{W}_1 \odot \mathbf{x}_n))$$

where $\sigma(\mathbf{x})$ combines nonlinearity and batch normalization

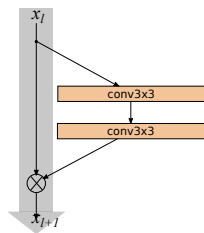
Residual blocks



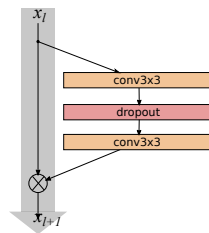
(a) basic



(b) bottleneck



(c) basic-wide



(d) wide-dropout

WRN architecture

group name	output size	block type = $B(3, 3)$
conv1	32×32	$[3 \times 3, 16]$
conv2	32×32	$\begin{bmatrix} 3 \times 3, 16 \times k \\ 3 \times 3, 16 \times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3 \times 3, 32 \times k \\ 3 \times 3, 32 \times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3 \times 3, 64 \times k \\ 3 \times 3, 64 \times k \end{bmatrix} \times N$
avg-pool	1×1	$[8 \times 8]$

Table: Structure of wide residual networks. Network width is determined by factor k .

CIFAR results

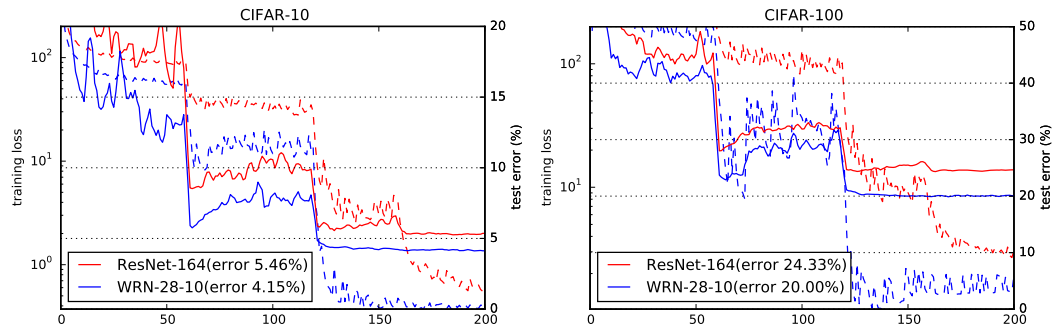


Figure: Training curves for thin and wide residual networks on CIFAR-10 and CIFAR-100. Solid lines denote test error (y-axis on the right), dashed lines denote training loss (y-axis on the left).

CIFAR computational efficiency

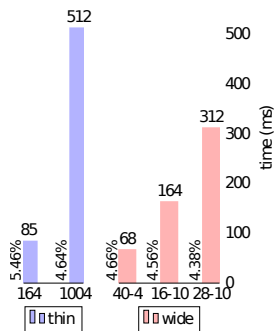


Figure: Time of forward+backward update per minibatch of size 32 for wide and thin networks(x-axis denotes network depth and widening factor).

Making network deeper **makes computation sequential**, we want it to be parallel!

ImageNet: basic block width

width		1.0	2.0	3.0	4.0
WRN-18	top1,top5	30.4, 10.93	27.06, 9.0	25.58, 8.06	24.06, 7.33
	#parameters	11.7M	25.9M	45.6M	101.8M
WRN-34	top1,top5	26.77, 8.67	24.5, 7.58	23.39, 7.00	
	#parameters	21.8M	48.6M	86.0M	

Table: ILSVRC-2012 validation error (single crop) of non-bottleneck ResNets with various width. Networks with the comparable number of parameters achieve similar accuracy, despite having 2 times less layers.

ImageNet: bottleneck block width

Model	top-1 err, %	top-5 err, %	#params	time/batch 16
ResNet-50	24.01	7.02	25.6M	49
ResNet-101	22.44	6.21	44.5M	82
ResNet-152	22.16	6.16	60.2M	115
WRN-50-2	21.9	6.03	68.9M	93
pre-ResNet-200	21.66	5.79	64.7M	154

Table: ILSVRC-2012 validation error (single crop) of bottleneck ResNets. Faster WRN-50-2 outperforms ResNet-152 having 3 times less layers, and stands close to pre-ResNet-200.

Conclusions

- Harder the task, more layers we need:
 - **MNIST**: 2 layers
 - **SVHN**: 8 layers
 - **CIFAR**: 20 layers
 - **ImageNet**: 50 layers
- ResNet **does not benefit from increased depth**, it benefits from increased capacity
- Deeper networks are not better for transfer learning
- After some point, only **number of parameters** matters: you can vary depth/width and get the same performance

Dirac parameterizations

1. Motivation
2. Wide residual parameterizations
3. Dirac parameterizations
4. Symmetric parameterizations

Training Very Deep Neural Networks Without Skip-Connections,
Zagoruyko&Komodakis, 2017, <https://arxiv.org/abs/1706.00388>

Do we need skip-connections?

Several issues with skip-connections in ResNet:

- Actual depth is not clear: might be determined by the shortest path
- Information can bypass nonlinearities, some blocks might not learn anything useful

Can we train a vanilla network without skip-connections?

Dirac parameterization

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

$$\mathbf{I} \odot \mathbf{x} = \mathbf{x}$$

In 2-d case: Kronecker delta, or identity matrix.

In N-d case:

$$\mathbf{I}(i, j, l_1, l_2, \dots, l_L) = \begin{cases} 1 & \text{if } i = j \text{ and } l_m \leq K_m \text{ for } m = 1..L, \\ 0 & \text{otherwise;} \end{cases}$$

Dirac parameterization

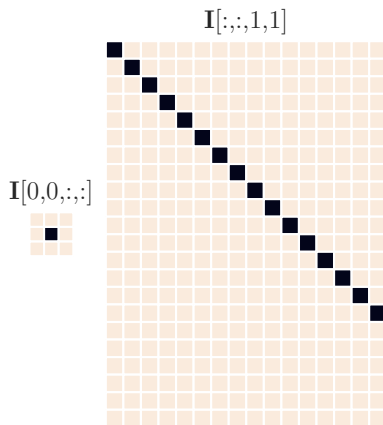


Figure: 4D-Dirac parameterized filters

Dirac parameterization

For a convolutional layer $\mathbf{y} = \hat{\mathbf{W}} \odot \mathbf{x}$ we propose the following parameterization for the weight tensor $\hat{\mathbf{W}}$:

$$\mathbf{y} = \hat{\mathbf{W}} \odot \mathbf{x},$$
$$\hat{\mathbf{W}} = \text{diag}(\mathbf{a})\mathbf{I} + \text{diag}(\mathbf{b})\mathbf{W}_{\text{norm}},$$

where:

- \mathbf{a} – scaling vector (init $\mathbf{a}_0 = 1$) [no weight decay]
- \mathbf{b} – scaling vector (init $\mathbf{b}_0 = 0.1$) [no weight decay]
- \mathbf{W}_{norm} – normalized weight tensor where each filter \mathbf{v} is normalized by it's Euclidean norm (init \mathbf{W} from normal distribution $\mathcal{N}(0, 1)$)

Connection to ResNet

Due to distributivity of convolution:

$$\mathbf{y} = \sigma((\mathbf{I} + \mathbf{W}) \odot \mathbf{x}) = \sigma(\mathbf{x} + \mathbf{W} \odot \mathbf{x}),$$

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

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

- Dirac parameterization and ResNet differ only by the order of nonlinearities
- Each delta parameterized layer adds complexity by having unavoidable nonlinearity
- Dirac parameterization can be folded into a single weight tensor on inference

DiracNet architecture

Same with ResNet, but with Dirac parametrization instead of residuals

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

Table: 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.

CIFAR results

	depth-width	# params	CIFAR-10	CIFAR-100
DiracNet	28-5	9.1M	4.93	23.39
	28-10	36.5M	4.73	21.59
ResNet	1001-1	10.2M	4.92	22.71
WRN	28-10	36.5M	4.00	19.25

Table: 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.

CIFAR results

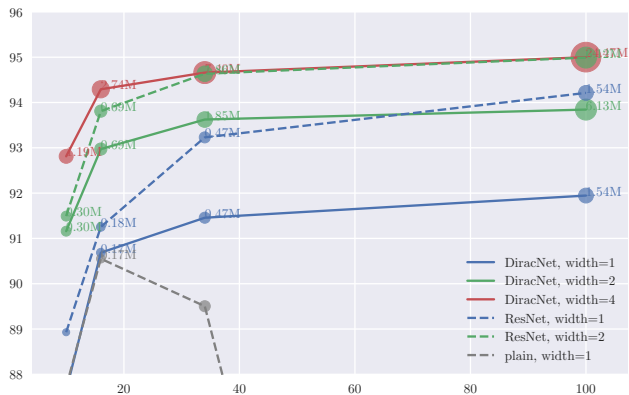


Figure: DiracNet and ResNet with different depth/width, each circle area is proportional to number of parameters.

ImageNet results

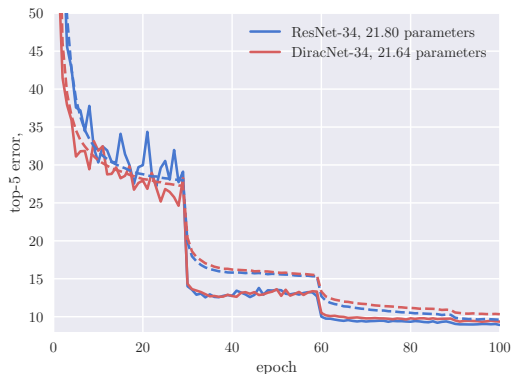
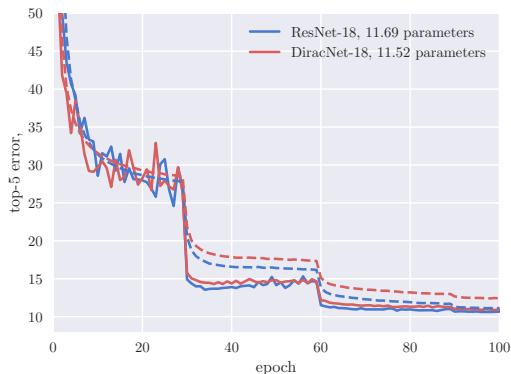


Figure: Convergence of DiracNet and ResNet on ImageNet. Training top-5 error is shown with dashed lines, validation - with solid. All networks are trained using the same optimization hyperparameters.

ImageNet results

	Network	# parameters	top-1 error	top-5 error
plain	VGG-CNN-S	102.9M	36.94	15.40
	VGG-16	138.4M	29.38	-
	DiracNet-18	11.7M	30.37	10.88
	DiracNet-34	21.8M	27.79	9.34
residual	ResNet-18 [our baseline]	11.7M	29.62	10.62
	ResNet-34 [our baseline]	21.8M	27.17	8.91

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

In a trained network Dirac parameterization and batch normalization fold into filters:

$$\hat{\mathbf{W}} = \text{diag}(\mathbf{a})\mathbf{I} + \text{diag}(\mathbf{b})\mathbf{W}_{\text{norm}},$$

Resulting in MLP-like architecture (for n -th layer):

$$\mathbf{x}_{n+1} = \text{ReLU}(\hat{\mathbf{W}}_n \odot \mathbf{x}_n)$$

Conclusions

- + Trained network is very simple: Dirac parametrization folds into weights, resulting in a plain feed-forward network like VGG
- + Can match ResNet accuracy on ImageNet
- Worse parameter efficiency and top accuracy on CIFAR (probably due to weight decay)

DiracNets do not solve the depth issues yet, but significantly simplifies deep networks.

Symmetric parameterizations

1. Motivation
2. Wide residual parameterizations
3. Dirac parameterizations
4. Symmetric parameterizations

Exploring Weight Symmetry in Deep Neural Networks,

Sergey Zagoruyko, Shell Hu, Nikos Komodakis, under review at CVPR 2018

Networks that achieve top accuracy are massively overparameterized, e.g. 50M-100M parameters for top ImageNet and seq2seq models.

Can we somehow introduce structure in linear layers to reduce the number of parameters, keeping the network capacity?

We propose to introduce symmetry:

- Channelwise symmetry: over feature dimension
- Spatial symmetry: over spatial dimensions

Example: in $32 \times 32 \times 3 \times 3$ filters channelwise over 32×32 , spatial over 3×3 ¹.

¹require filters to have equal numbers of input and output channels

Ways to impose symmetry

Soft constraint (additional loss):

$$\mathbb{E} \left[\mathcal{L}(\tilde{\mathbf{W}}, x) \right] + \rho \sum_{l=1}^L \sum_{i \in I} \left\| \text{vec}(\mathbf{W}_i^l) - \text{vec}(\mathbf{W}_i^{l\top}) \right\|_p \quad (3)$$

At test time we use upper triangular part for lower triangular part for each layer (similar to pruning).

- Same number of parameters at train time, $2\times$ less at test time
- + More freedom during training

Ways to impose symmetry

Hard parameterization:

$$\hat{\mathbf{W}} = f(\mathbf{W}, \mathbf{v}) := \text{diag}(\mathbf{v}) + \text{triu}(\mathbf{W}) + \text{triu}(\mathbf{W})^\top,$$

\mathbf{W} is an upper triangular matrix. We call the above *triangular parameterization*.

- + 2× less parameters both at train and test time
- + Potential speed-up both at train and test time
- Less freedom in linear layers

Other hard parameterizations:

- *average parameterization:*

$$\hat{\mathbf{W}} = f(\mathbf{W}) := \frac{1}{2}(\mathbf{W} + \mathbf{W}^\top)$$

- *Eigen parameterization:*

$$\hat{\mathbf{W}} = f(\mathbf{V}, \lambda) := \mathbf{V} \text{diag}(\lambda) \mathbf{V}^\top$$

- *LDL parameterization:*

$$\hat{\mathbf{W}} = f(\mathbf{L}, \mathbf{D}) := \mathbf{L} \mathbf{D} \mathbf{L}^\top, \tag{4}$$

N-way parameterizations:

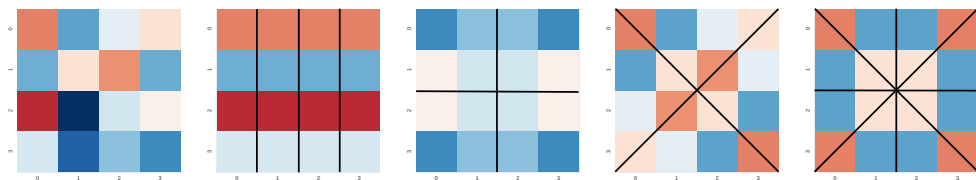


Figure: N-way parameterizations. (a) Original 4×4 weight matrix. (b) 4-way chunking: V is the first strip; $\hat{W} = \text{tile}_{4 \times}(V)$. (c) 4-way blocking: V is the bottom-right block; $\hat{W} = \text{reflect}_-(\text{reflect}_\downarrow(V))$. (d) 4-way triangulizing: V is the top triangle; $\hat{W} = \text{reflect}_\downarrow(\text{reflect}_\setminus(V))$. (e) 8-way triangulizing: V is the top-left triangle; $\hat{W} = \text{reflect}_\downarrow(\text{reflect}_\setminus(\text{reflect}_\downarrow(V)))$.

CIFAR results

symmetrization	#parameters		CIFAR-10
	train	test	
baseline	0.219M	0.219M	8.49
L1 soft	0.219M	0.172M	8.61
channel-triangular	0.172M	0.172M	8.84
channel-average	0.219M	0.172M	8.83
channel-eigen	0.173M	0.173M	10.23
channel-LDL	0.172M	0.172M	9.15
spatial-average	0.219M	0.187M	9.70
spatial&channel-average	0.219M	0.156M	10.20

Table: Various parameterizations on CIFAR-10 with WRN-16-1-bottleneck.

Basic or bottleneck

Basic: 3×3 , 3×3 , imposing symmetry on both is very restrictive, 50% parameter reduction.

Bottleneck: 1×1 , 3×3 , 1×1 , imposing symmetry on 3×3 only, 1×1 are “free” layers, 25% parameter reduction.

CIFAR results

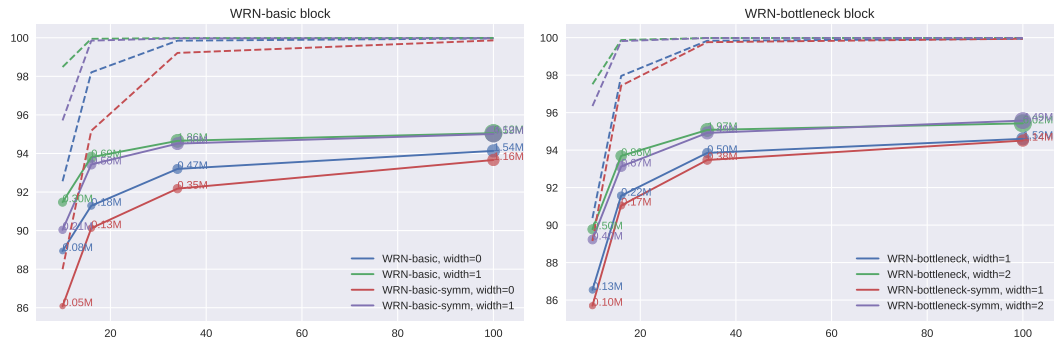
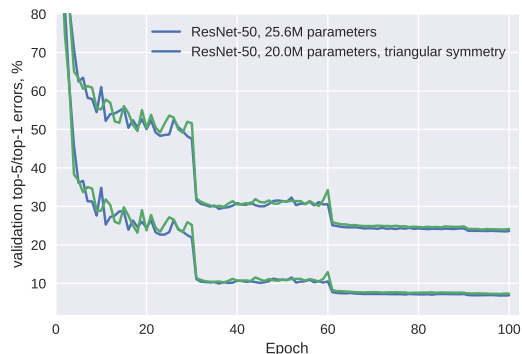


Figure: WRN of various depth and width with basic (left) and bottleneck (right) blocks and triangular symmetry. Dashed lines denote training accuracy, solid - validation (median of 5 runs). Accuracy reduction in bottleneck block is much lower due to supporting 1×1 convolutions without symmetry constraint.

ImageNet results

network	sym	#params	top-1	top-5
MobileNet		4.2M	28.18	9.8
MobileNet	✓	3.0M	30.57	11.6
ResNet-18		11.8M	30.54	10.93
ResNet-18	✓	8.6M	31.44	11.55
ResNet-50		25.6M	23.50	6.83
ResNet-50	✓	20.0M	23.98	7.25
ResNet-101		44.7M	22.14	6.09
ResNet-101	✓	34.0M	22.36	6.35



Channelwise symmetry

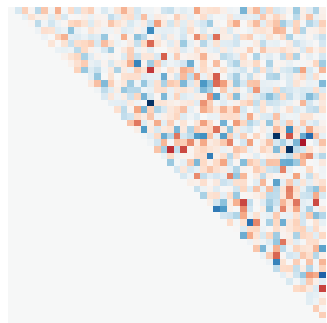
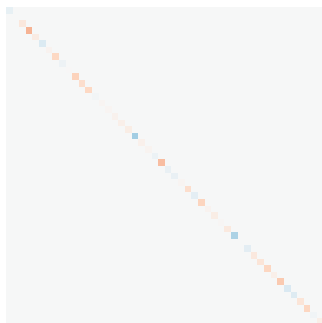
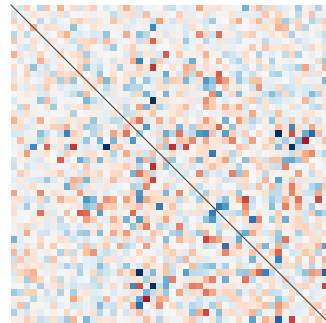
(a) \mathbf{W} (b) \mathbf{v} (c) $\hat{\mathbf{W}}$

Figure: Visualization of a channel slice of weights from ResNet-50 trained with triangular parameterization. (a) and (b) show triangular parameterization weights, upper triangular and diagonal, (c) shows resulting symmetric weight matrix.





Conclusions

- Weights in deep neural networks can be constrained to be symmetric without significant loss in accuracy, as long as they are able to closely fit into training data.
- Networks with 1×1 layers such as MobileNet can benefit from specialized SYMM routines on CPU and GPU, and convolutional layers could be potentially made faster too.

Conclusions

Need to continue looking for better parameterizations:

- Automatic architecture search?
- Issues of weight decay combined with batch norm?

-  Ba, J. and Caruana, R. (2014).
Do deep nets really need to be deep?
In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2654–2662. Curran Associates, Inc.
-  Cybenko, G. (1989).
Approximations by superpositions of sigmoidal functions.
In *Mathematics of Control, Signals, and Systems*, volume 2 (4), pages 303–314.
-  He, K., Zhang, X., Ren, S., and Sun, J. (2015).
Deep residual learning for image recognition.
CoRR, abs/1512.03385.
-  Polyak, B. (1964).
Some methods of speeding up the convergence of iteration methods.
4:1–17.