All you need is a good init

Dmytro Mishkin and Jiří Matas
Center for Machine Perception Czech Technical University in Prague

TRAINING A VERY DEEP NET AND THE PER-LAYER GAIN

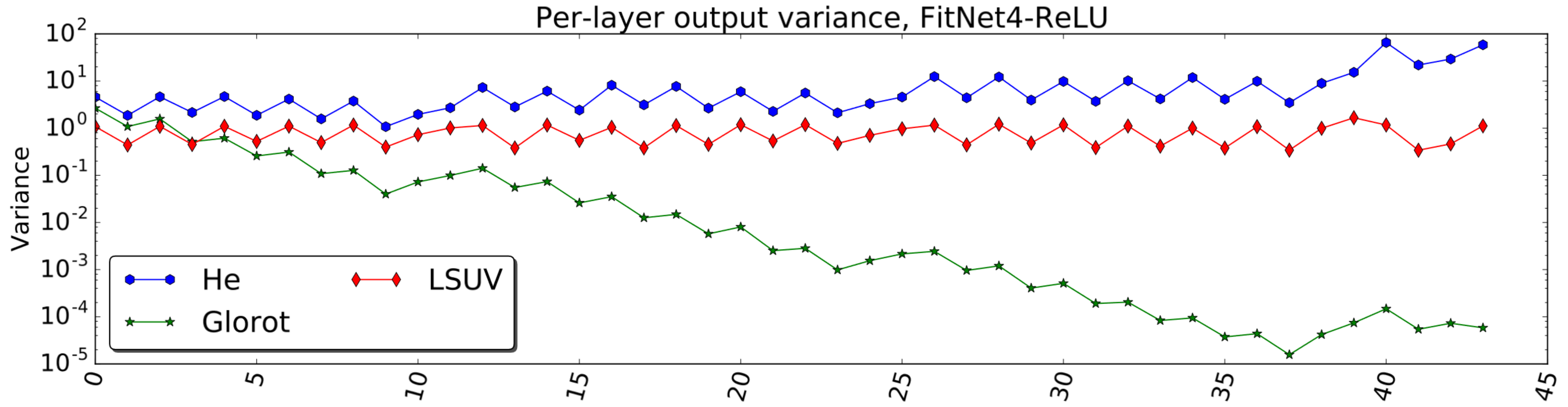
Layer gain $G_L = var(O_L)/var(I_L)$, where $var(I_L)$ – variance of layer input, $var(O_L)$ – variance of layer output.

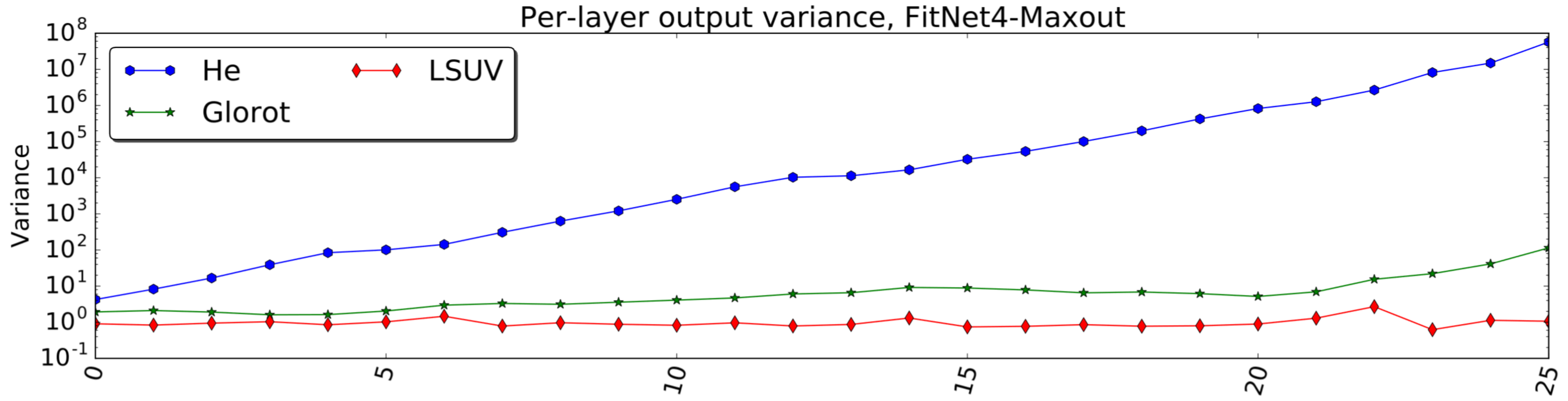
1. Very deep neural networks are powerful but hard to train.
2. Observation: regardless of the non-linearity used, deep net trains well, if its **product of per-layer gains** equals to one: $G_{\text{DNN}} = \prod_{i=1}^{n} G_{L_i} \approx 1$ (1)
3. Initialization satisfying Eq. (1) exists only for linear and ReLU networks. We propose an initialization algorithm applicable to any feedforward network.

PROBLEM: HOW TO START TRAINING A VERY DEEP NET

Common initialization methods lead to the same layer gain G_L for each convolutional and fully-connected layer, which works if input variance is 1 and no other types of layers are present. If a significant number of other type of layers is present:

a) Layer gain $G_L < 1 \rightarrow$ **vanishing variance**
b) Layer gain $G_L > 1 \rightarrow$ **exploding variance**





Deriving layer gain G_L provably ensuring $G_{\text{DNN}}^{\#\text{Layer}} = \prod_{i=1}^{n} G_{L_i} = 1$ is a **hard task for general network** with various activation functions, poolings, skip connections, etc.

STATE OF THE ART

Machine learning basics: centered and normalized (mean = 0, var = 1) input is good.
Glorot & Bengio (2010): keep input (and output) of each layer normalized, propose weight initialization formula for linear net.
He et. al (2015): modifies the Glorot formula for ReLU net.
Batch Norm (2015): explicitly calculate mean and variance for each batch and use them for normalization. Do it every forward pass.
Recurring theme: many functions (Maxout, ELU, etc.) are superior to ReLU.

KEEPING PRODUCT OF PER-LAYER GAINS ≈ 1: LAYER-SEQUENTIAL UNIT-VARIANCE ORTHOGONAL INITIALIZATION

Algorithm 1. Layer-sequential unit-variance orthogonal initialization. $\boldsymbol{L}$ – convolution or fully-connected layer, W_L – its weights, O_L – layer output, ε – variance tolerance, T_i – iteration number, T_{max} – max number of iterations.

Pre-initialize network with orthonormal matrices as in Saxe et.al. (2013)
for each convolutional and fully-connected layer $\boldsymbol{L}$ **do**
  do forward pass with mini-batch
    calculate $var(O_L)$
    $W_L^{i+1} = W_L^i / \sqrt{var(O_L)}$
  until $|var(O_L) - 1.0| < \varepsilon$ **or** $(T_i > Tmax)$
end for

*The LSUV algorithm does not deal with biases and initializes them with zeros

CIFAR-10/100 RESULTS

Accuracy on CIFAR-10/100, with data augmentation

Network	CIFAR-10, [%]	CIFAR-100,[%]
Fitnet4-LSUV	**93.94**	70.04 (**72.34**†)
Fitnet4-OrthoInit	93.78	70.44 (72.30†)
Fitnet4-Hints	91.61	64.96
Fitnet4-Highway	92.46	68.09
ALL-CNN	92.75	66.29
DSN	92.03	65.43
NiN	91.19	64.32
Maxout	90.62	65.46
MIN	*93.25*	*71.14*
Extreme data augmentation		
Large ALL-CNN	95.59	n/a
Fractional MP (1 test)	95.50	68.55
Fractional MP (12 tests)	**96.53**	**73.61**

MNIST RESULTS

Error on MNIST w/o data augmentation

Network	layers	params	Error, %
FitNet-like networks			
HighWay-16	10	39K	0.57
FitNet-Hints	6	30K	0.51
FitNet-Ortho	6	30K	0.48
FitNet-LSUV	6	30K	0.48
FitNet-Ortho-SVM	6	30K	0.43
FitNet-LSUV-SVM	6	30K	**0.38**
State-of-art-networks			
DSN-Softmax	3	350K	0.51
DSN-SVM	3	350K	0.39
HighWay-32	10	151K	0.45
Maxout	3	420K	0.45
MIN	*9*	*447K*	*0.24*

COMPARISON OF THE INITIALIZATIONS FOR DIFFERENT ACTIVATIONS

CIFAR-10 FITNET

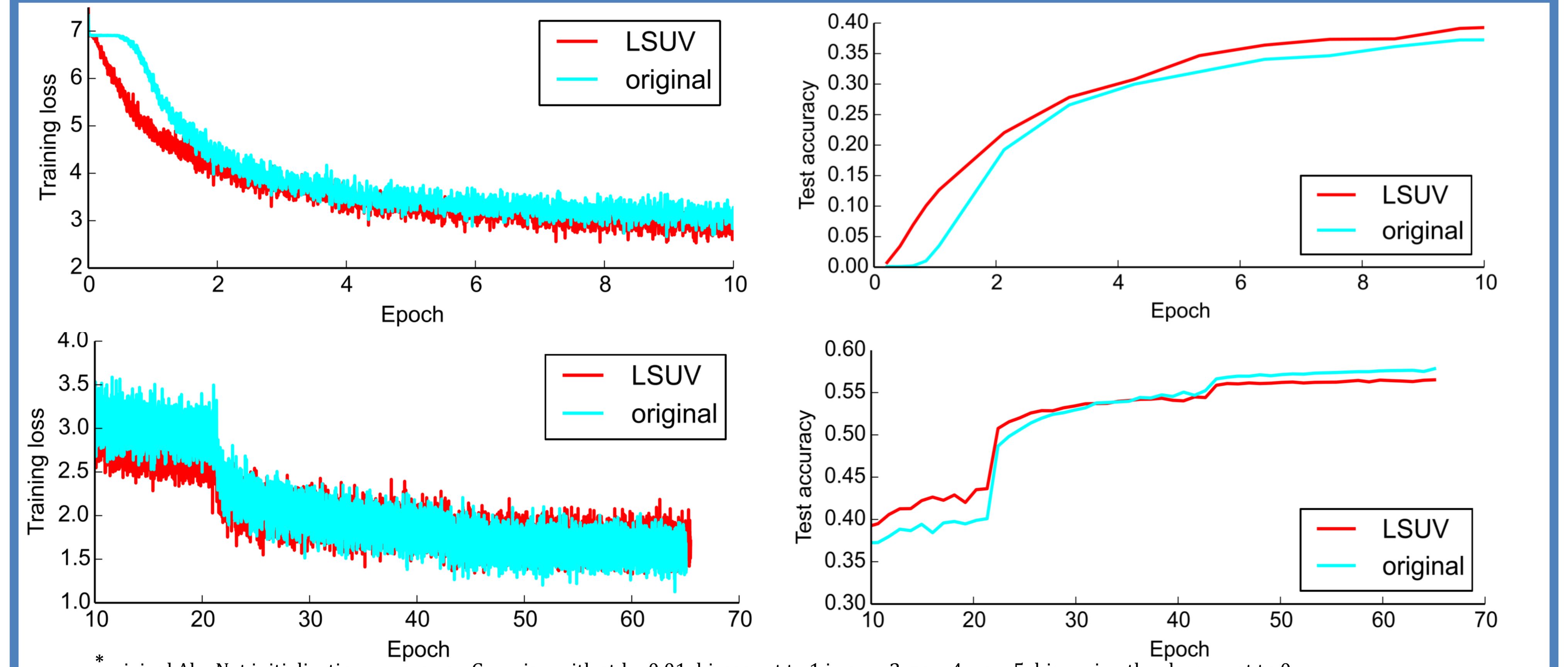
Init method	Maxout	ReLU	VLReLU	tanh
LSUV	**93.94**	**92.11**	92.97	89.28
OrthoNorm	93.78	91.74	92.40	89.48
Xavier	91.75	90.63	92.27	**89.82**
MSRA	n/c†	90.91	92.43	89.54
OrthoNorm MSRA-scaled	–	91.93	**93.09**	–

CIFAR-10 RESIDUAL FITNET

Init method	maxout	ReLU	VLReLU	tanh
LSUV	**94.16**	**92.82**	**93.36**	89.17
OrthoNorm	n/c	91.42	n/c	89.31
Xavier	n/c	92.48	**93.34**	**89.62**
MSRA	n/c	n/c	n/c	88.59
–	–	–	–	–

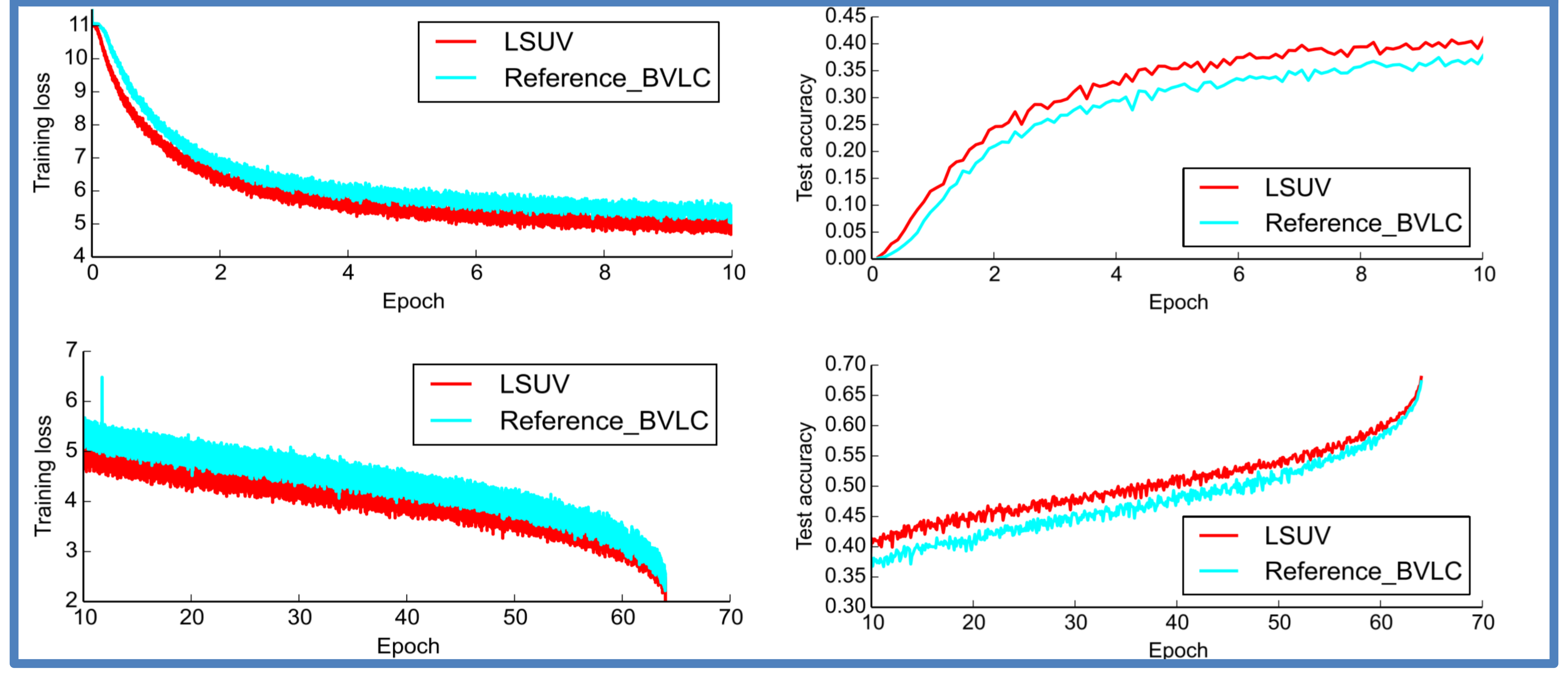
The authors were supported by The Czech Science Foundation Project GACR P103/12/G084 and CTU student grant SGS15/155/OHK3/2T/13

CAFFENET TRAINING



*original AlexNet initialization, zero-mean Gaussian with std = 0.01; biases set to 1 in conv2, conv4, conv5; biases in other layers set to 0.

GOOGLENET TRAINING



REFERENCES

[1] Saxe, Andrew M., McClelland, James L., and Ganguli, Surya. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In Proceedings of ICLR, 2014.
[2] Romero, Adriana, Ballas, Nicolas, Kahou, Samira Ebrahimi, Chassang, Antoine, Gatta, Carlo, and Bengio, Yoshua. Fitnets: Hints for thin deep nets. In Proceedings of ICLR, May 2015.
[3] Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the AISTATS 2010.
[4] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the ICCV, 2015
[5] Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training b reducing internal covariate shift. In Proceedings of the ICML, 2015.