All You Need Is A Good Init

HAI 소성민
2020.08.08

Why?

- initialize all the weight to 0?
  - symmetry problem

- initialize big random values?
  - exploding gradient problem

- initialize small random values?
  - vanishing gradient problem

Notes: https://towardsdatascience.com/why-better-weight-initialization-is-important-in-neural-networks-ff9acf01026d

First, pre-initialize weights of each convolution or inner-product layer with orthonormal matrices.

Second, proceed from the first to the final layer, normalizing the variance of the output of each layer to be equal to one.

Independently, Saxe et al. (2014) showed that ***orthonormal matrix initialization*** *works much better for linear networks than Gaussian noise, which is only approximate orthogonal.*

It also work for networks with non-linearities.

LSUV Algorithm

Algorithm 1 Layer-sequential unit-variance orthogonal initialization. L – convolution or full-connected layer, W_L - its weights, B_L - its output blob., Tol_{var} - variance tolerance, T_i – current trial, T_{max} – max number of trials.

Pre-initialize network with orthonormal matrices as in Saxe et al. (2014)
for each layer L **do**
  while $|Var(B_L) - 1.0| \geq Tol_{var}$ and $(T_i < T_{max})$ **do**
    do Forward pass with a mini-batch
    calculate $Var(B_L)$
    $W_L = W_L \,/\, \sqrt{Var(B_L)}$
  end while
end for

LSUV Algorithm

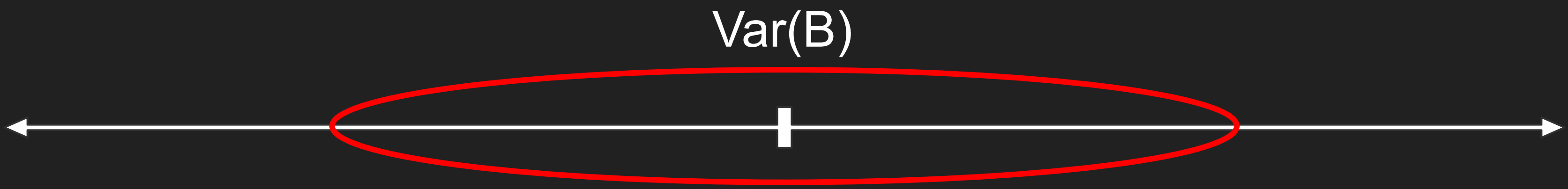
$$\textbf{while} \ \ |Var(B_L) - 1.0| \geq Tol_{var}$$

LSUV Algorithm

$$\textbf{while} \ |Var(B_L) - 1.0| \geq Tol_{var}$$

$$|Var(B_L) - 1.0| < Tol_{var}$$

LSUV Algorithm



$$|Var(B_L) - 1.0| < Tol_{var}$$

LSUV Algorithm

Algorithm 1 Layer-sequential unit-variance orthogonal initialization. L – convolution or full-connected layer, W_L - its weights, B_L - its output blob., Tol_{var} - variance tolerance, T_i – current trial, T_{max} – max number of trials.

Pre-initialize network with orthonormal matrices as in Saxe et al. (2014)
for each layer L **do**
  while $|Var(B_L) - 1.0| \geq Tol_{var}$ and $(T_i < T_{max})$ **do**
    do Forward pass with a mini-batch
    calculate $Var(B_L)$
    $W_L = W_L \,/\, \sqrt{Var(B_L)}$
  end while
end for

LSUV Algorithm

$$W_L = W_L \,/\, \sqrt{Var(B_L)}$$

LSUV Algorithm

$$W_l * X_i = B_i$$

$$\mathrm{Var}\left(\frac{W_l}{\sqrt{\mathrm{Var}\left(B_i\right)}} X_{i+1}\right)$$

LSUV Algorithm

$$\left(\frac{1}{\sqrt{\mathrm{Var}\left(B_i\right)}}\right)^2 \mathrm{Var}\left(W_l \; X_{i+1}\right)$$

$$\frac{1}{\mathrm{Var}\left(B_i\right)} \mathrm{Var}\left(W_l X_{i+1}\right) = \mathrm{Var}\left(B_{i+1}^{scale}\right)$$

Experimental Validation

Accuracy on CIFAR-10/100, with data augmentation

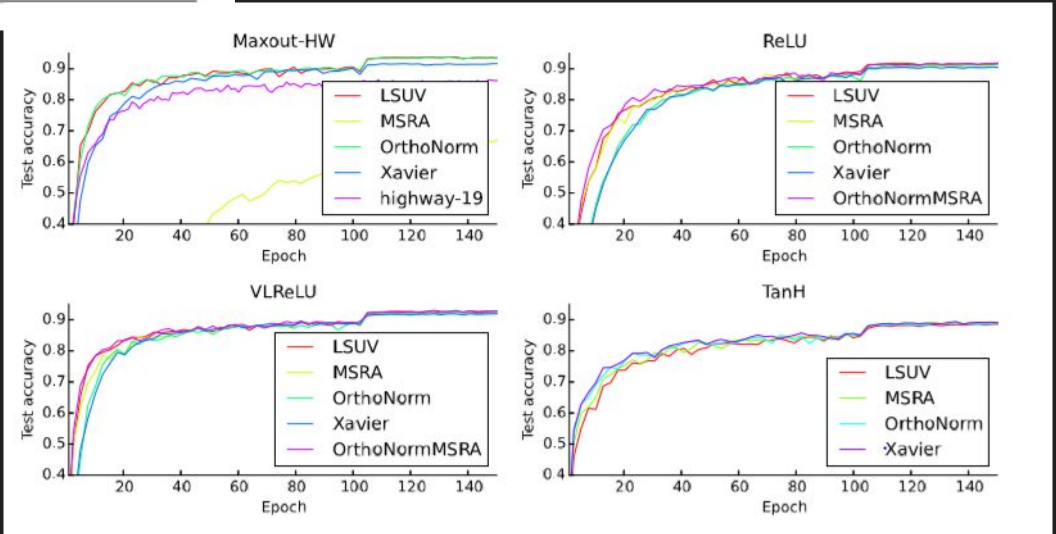
Network	CIFAR-10, [%]	CIFAR-100,[%]
Fitnet4-LSUV	**93.94**	70.04 (**72.34**†)
Fitnet4-OrthoInit	93.78	70.44 (72.30†)
Fitnet4-Hints	91.61	64.96
Fitnet4-Highway	92.46	68.09
ALL-CNN	92.75	66.29
DSN	92.03	65.43
NiN	91.19	64.32
maxout	90.62	65.46
MIN	*93.25*	*71.14*
Extreme data augmentation		
Large ALL-CNN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

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 [2]	*9*	*447K*	*0.24*

Experimental Validation

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



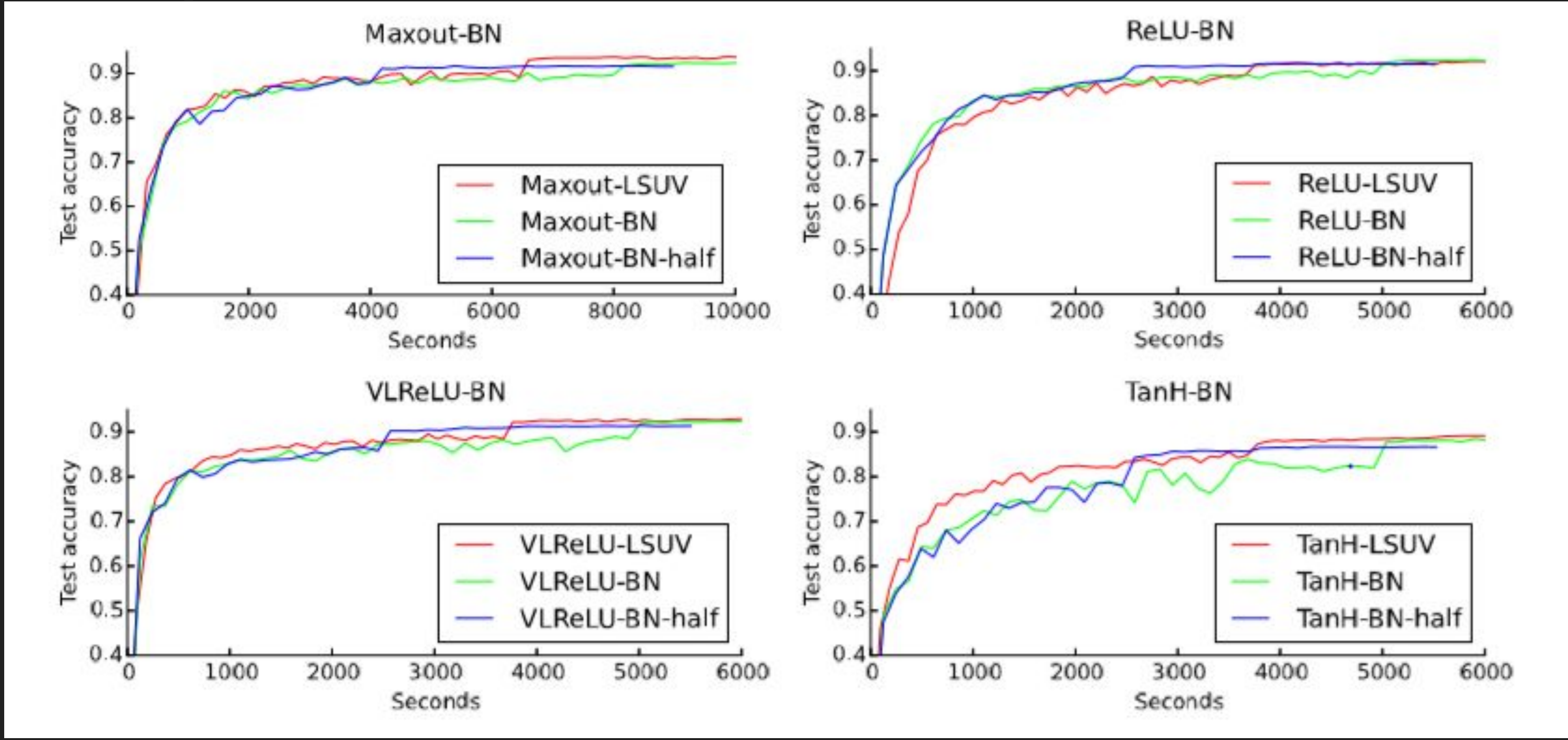
Compare with Batch Normalization

- LSUV Algorithm -> weight values
- Batch Normalization -> input data

Paper: https://arxiv.org/pdf/1502.03167.pdf

Notes: https://www.facebook.com/groups/TensorFlowKR/permalink/1022513991422992/

Compare with Batch Normalization



Pros and Cons

Table 6: Time needed for network initialization on top of random Gaussian (seconds).

Network	Init	
	OrthoNorm	LSUV
FitNet4	1	4
CaffeNet	188	210
GoogLeNet	24	60