On the Inspection of Initialization and Activations of Neural Networks

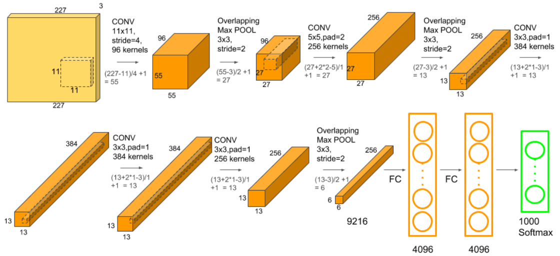
Presentor: Jinduk park

School of Mathematics and Computing (Computational Science and Engineering)
Yonsei Univ, Seoul, Korea

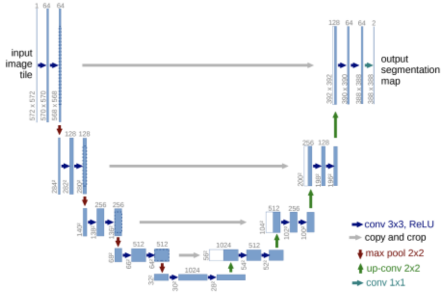
jindeok6@yonsei.ac.kr

Introduction

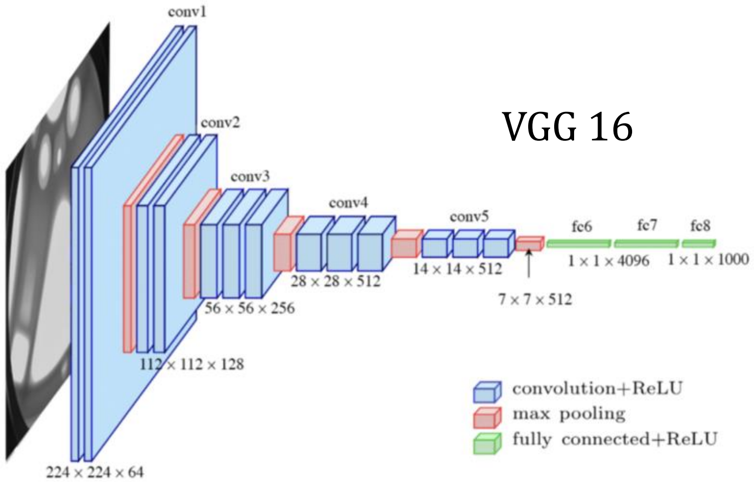
There are too many options to choice for designing neural network.



Alexnet



U-Net



VGG 16

Ronneberger et al. "U-net: Convolutional networks for biomedical image segmentation."

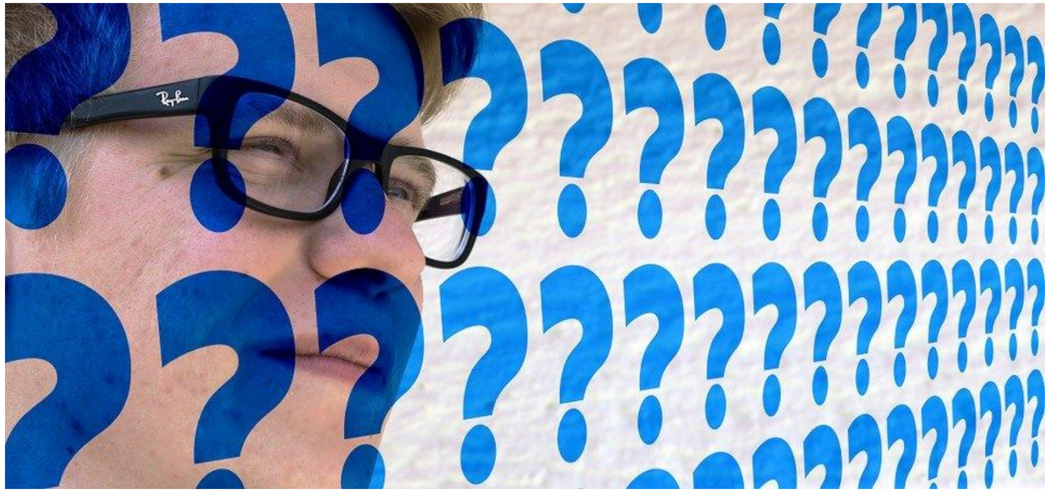
Krizhevsky et al. "Imagenet classification with deep convolutional neural networks."

Simonyan et al. "Very deep convolutional networks for large-scale image recognition."

Motivation

Which activation to use?
How many layers?
How to initialize weights?
...

Specific problem at hand

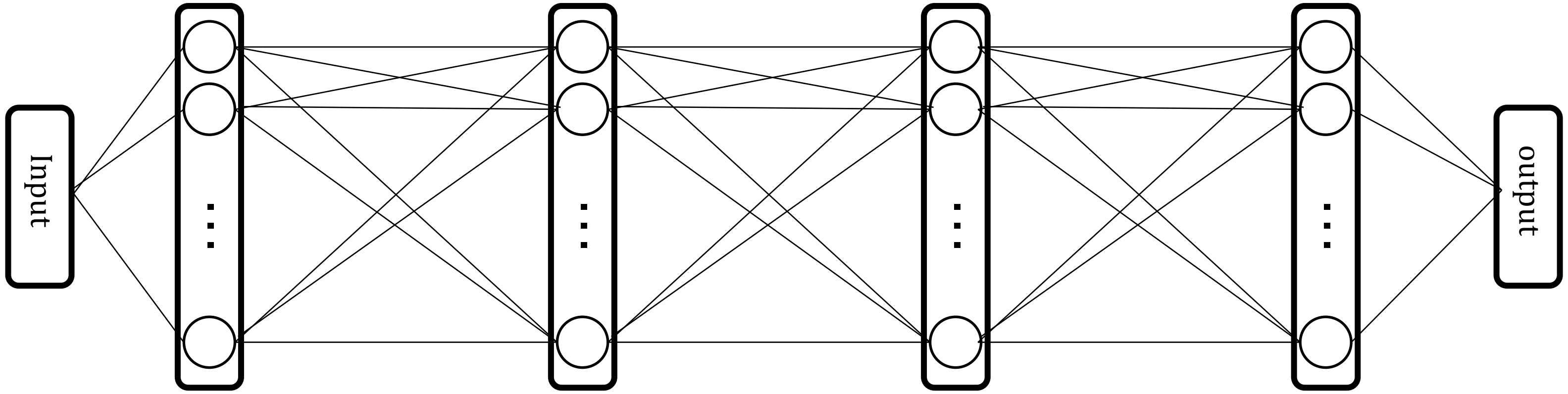


Many researcher usually do the puzzle game on those options.

Motivation

However,
those components are not mutually independent and they actucally affect each other.

The Neural Networks



Better understanding its **internal working** :
helpful to design neural network architectures

Two milestone initialization techniques

1. Xavier (Glorat) initialization

Glorot Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." JMLR Workshop and Conference Proceedings, 2010.

2. Kaiming (He) initialization

He Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." ICCV, 2015.

Two milestone initialization techniques

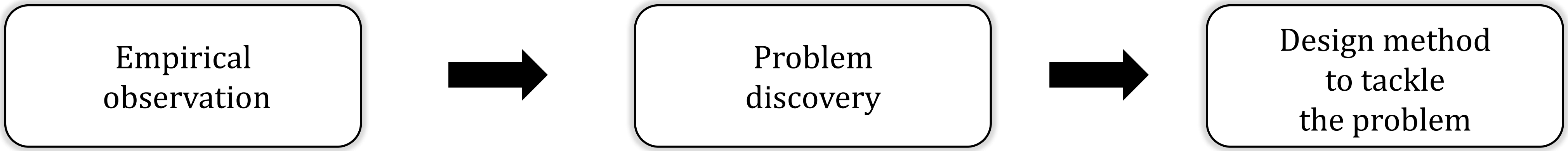
1. Xavier (Glorat) initialization

Glorot Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." JMLR Workshop and Conference Proceedings, 2010.

2. Kaiming (He) initialization

He Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." ICCV, 2015.

Logical flow of the research



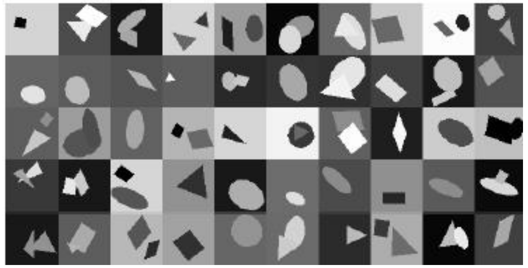
the behavior of backpropagation on various *activation functions*

Saturation problem over deeper layers

Proposing new initialization technique

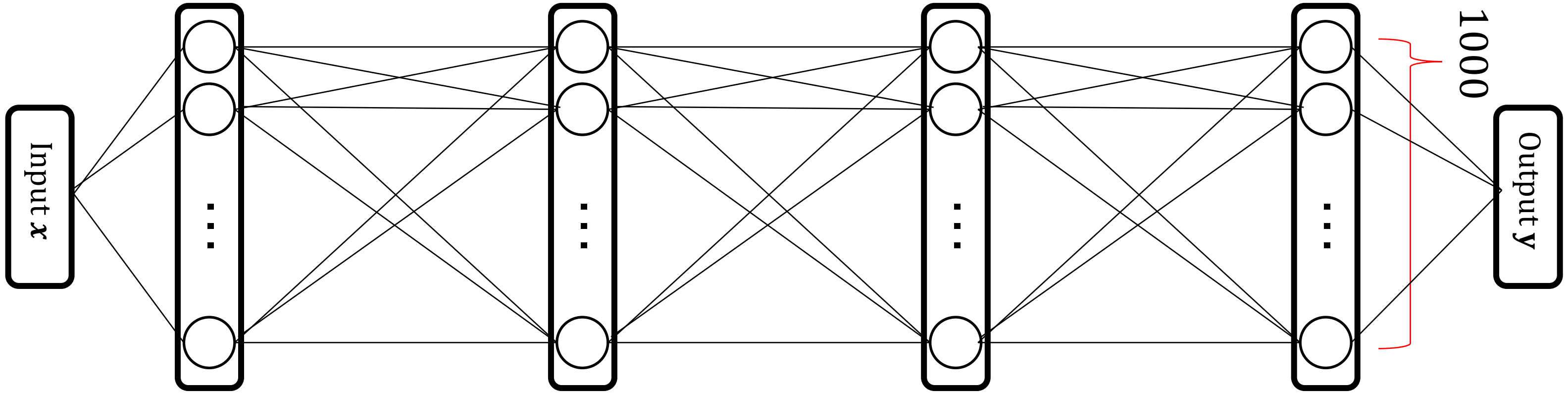
Basic Experimental Settings

Dataset



Shapeset-3x2 dataset
with small resolution ~ 64 x 64, gray-scale

The Neural Networks



1) 4 hidden layer with **n=1000** (same) neurons per each layer

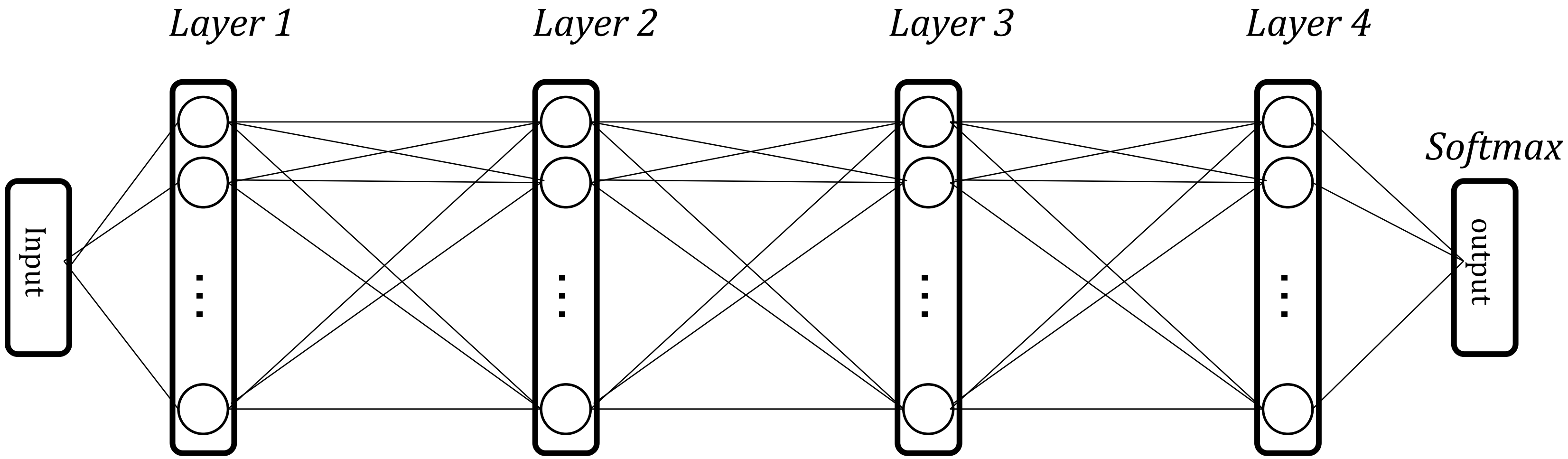
2) Different activations depending on the experiment

3) Weight initialization (standardly used heuristic)

$$W \sim U\left(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right)$$

(What conventionally used at that time)

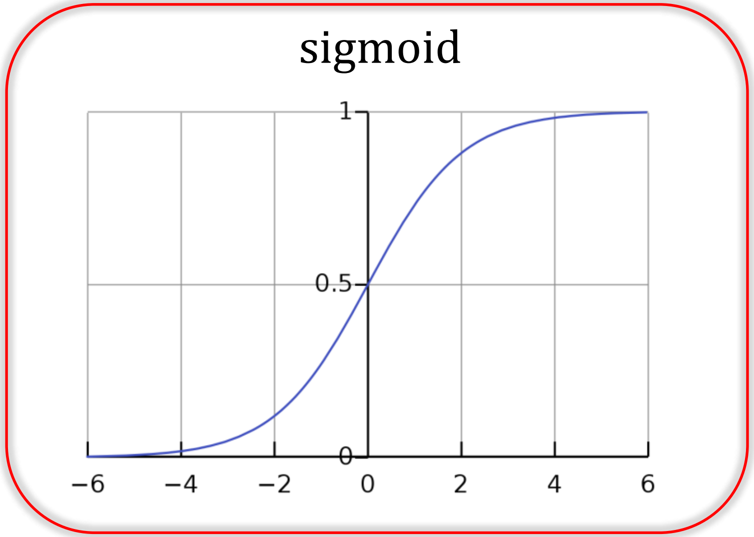
Basic Terms and Notations



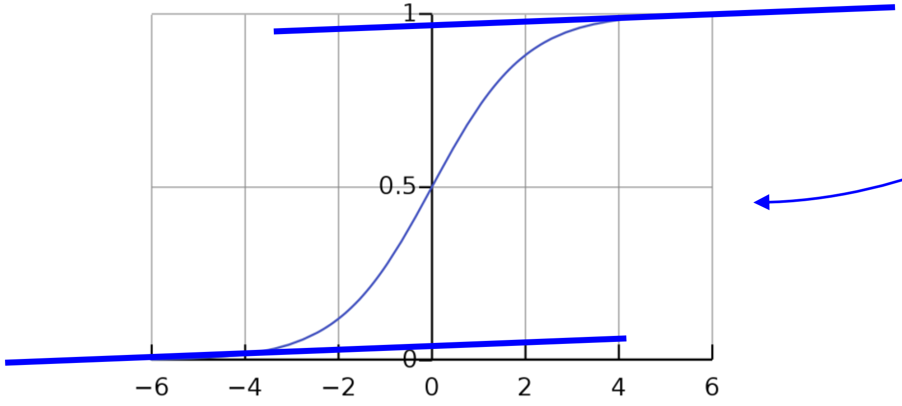
- ‘Layer i’ : the output of the i-th hidden layer
- $f(.)$: activation function
- $\boldsymbol{z^i}$: activation vector at layer i
- $\boldsymbol{s^i} = \boldsymbol{z^i W^i} + \boldsymbol{b^i}$
- $\boldsymbol{z}^{i+1} = f(\boldsymbol{s}^i)$

Supervised learning with Activations: 1) sigmoid

Experiment 1) All layer activations → set to *sigmoid*



- **None-zero mean**
- **Not symmetric**
- **~0 gradient near y=0, 1**



$$\frac{d}{dx} sigmoid(x) = sigmoid(x)(1 - sigmoid(x))$$

Supervised learning with Activations: 1) sigmoid

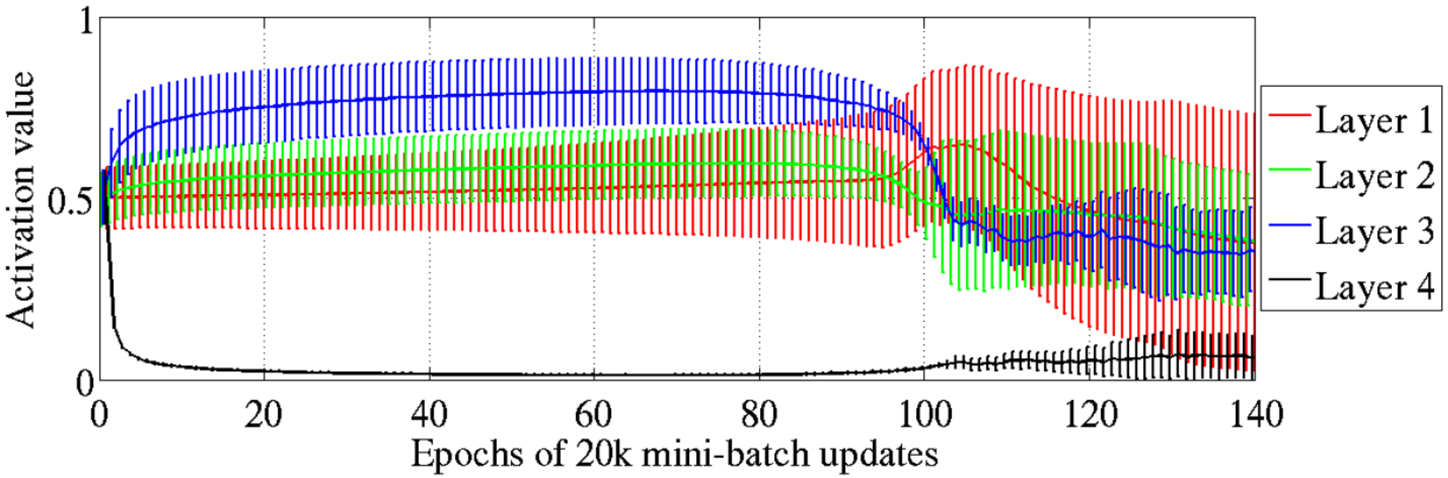


Figure: Mean and standard deviation (vertical bars) of the activation values (output of the sigmoid)

Empirical observation → Problem discovery → Design method to tackle the problem

Xavier et al, " Understanding the difficulty of training deep feedforward neural networks. " (2010).

Supervised learning with Activations: 1) sigmoid

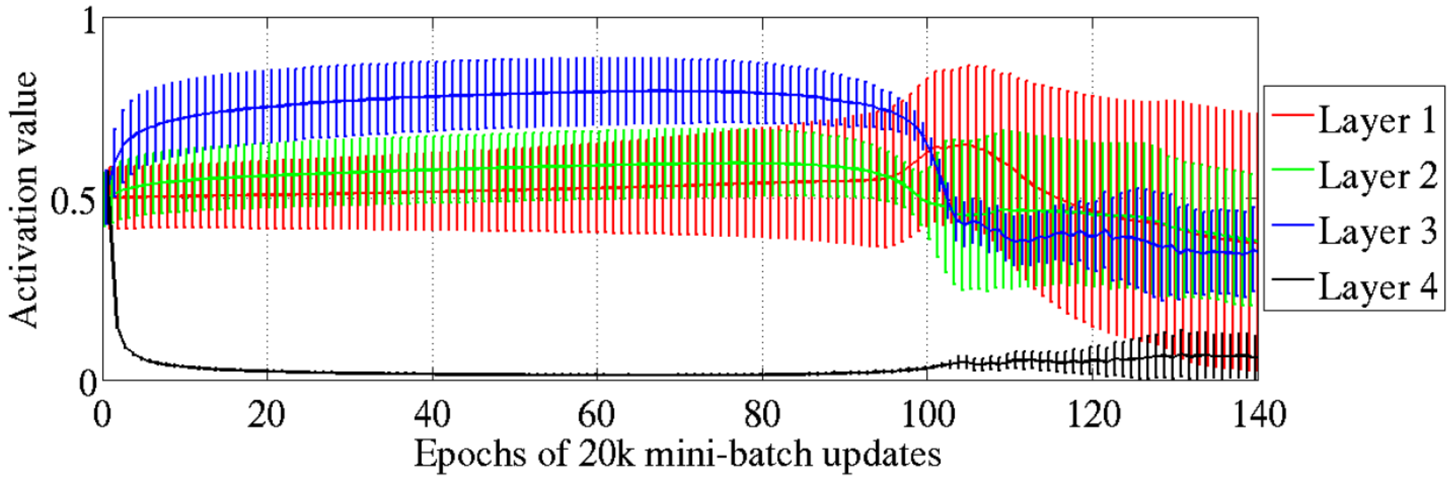


Figure: Mean and standard deviation (vertical bars) of the activation values (output of the sigmoid)

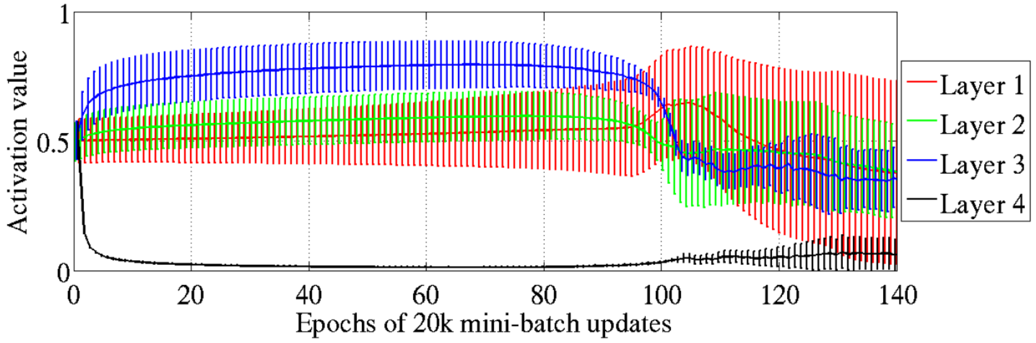
⬇

Observation: Excessive saturation

All the sigmoid activation at layer 4 : saturated to 0, **until ~100 epochs**

Xavier et al, " Understanding the difficulty of training deep feedforward neural networks. " (2010).

Supervised learning with Activations: 1) sigmoid



A Hypothesis:

" the transformation that the lower layers of the randomly initialized network computes initially is **not useful** to the classification task"

Output layer:

$$softmax(\boldsymbol{W}\boldsymbol{z}^{(4)} + \boldsymbol{b})$$

Initially ($W \sim U(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}})$),

The output layer is correlated mostly with dominant variations of x, not predictive of y

⟶ Rely mostly on $\boldsymbol{b}$ (learned very quickly) rather than $\boldsymbol{W}\boldsymbol{z}^{(4)}$

⟶ The network pushes $\boldsymbol{W}\boldsymbol{z}^{(4)}$ towards **0**,

which can be achieved by pushing $\boldsymbol{z}^{(4)}$ towards **0** ✓

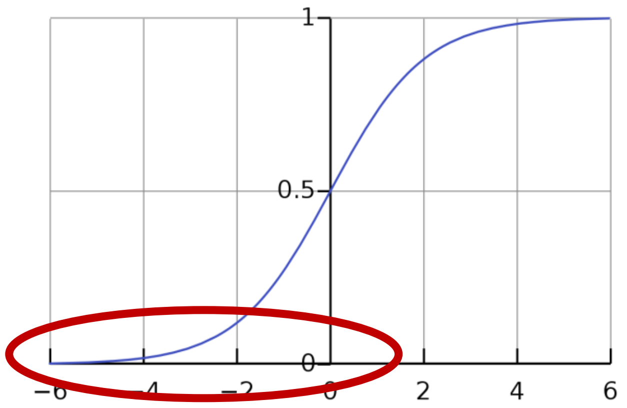
Xavier et al, " Understanding the difficulty of training deep feedforward neural networks. " (2010).

Supervised learning with Activations: 1) sigmoid

At the initial stage of training

pushing $z^{(4)}$ towards **0**

This is **not a good** thing for **sigmoid** function,
which is **not symmetric.**



Since the meaningful gradient cannot flow with this condition.

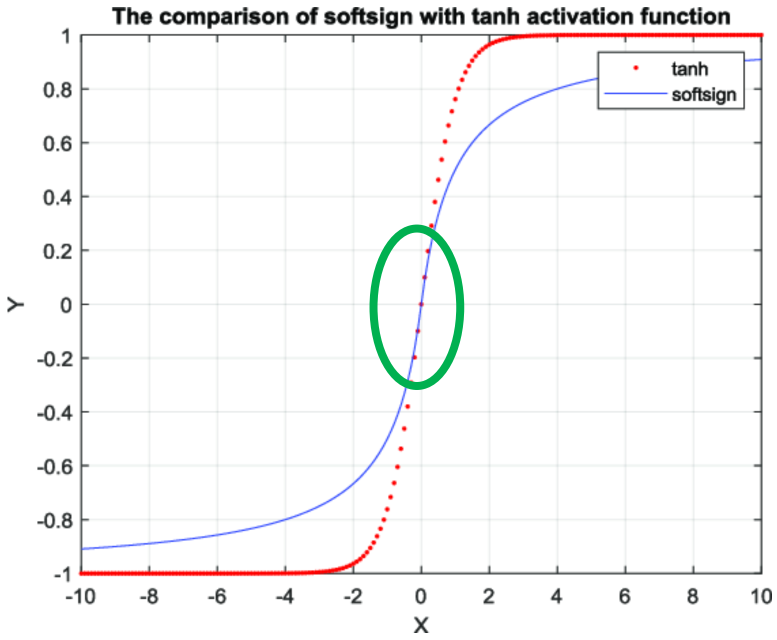
→ **Excessive saturation** problem

Xavier et al, " Understanding the difficulty of training deep feedforward neural networks. " (2010).

Supervised learning with Activations: 1) sigmoid

pushing $z^{(4)}$ towards $\mathbf{0}$

But this is **good** thing **symmetric functions !**
(e.g., hyperbolic tangent, softsign)

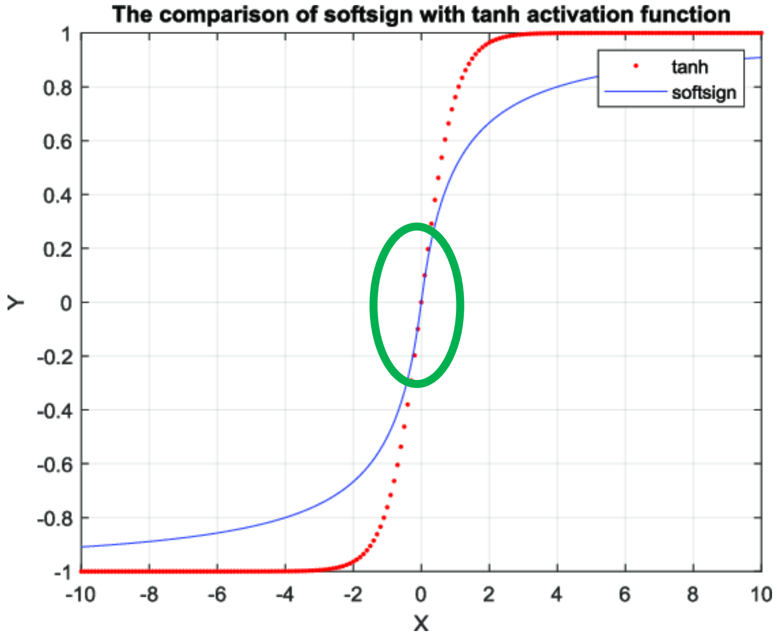


Effat et al, " Automatic Personality Traits Perception Using Asymmetric Auto-Encoder. “

Supervised learning with Activations: 1) sigmoid

pushing $\boldsymbol{z}^{(4)}$ towards **0**

But this is **good** thing **symmetric functions !**
(e.g., hyperbolic tangent, softsign)

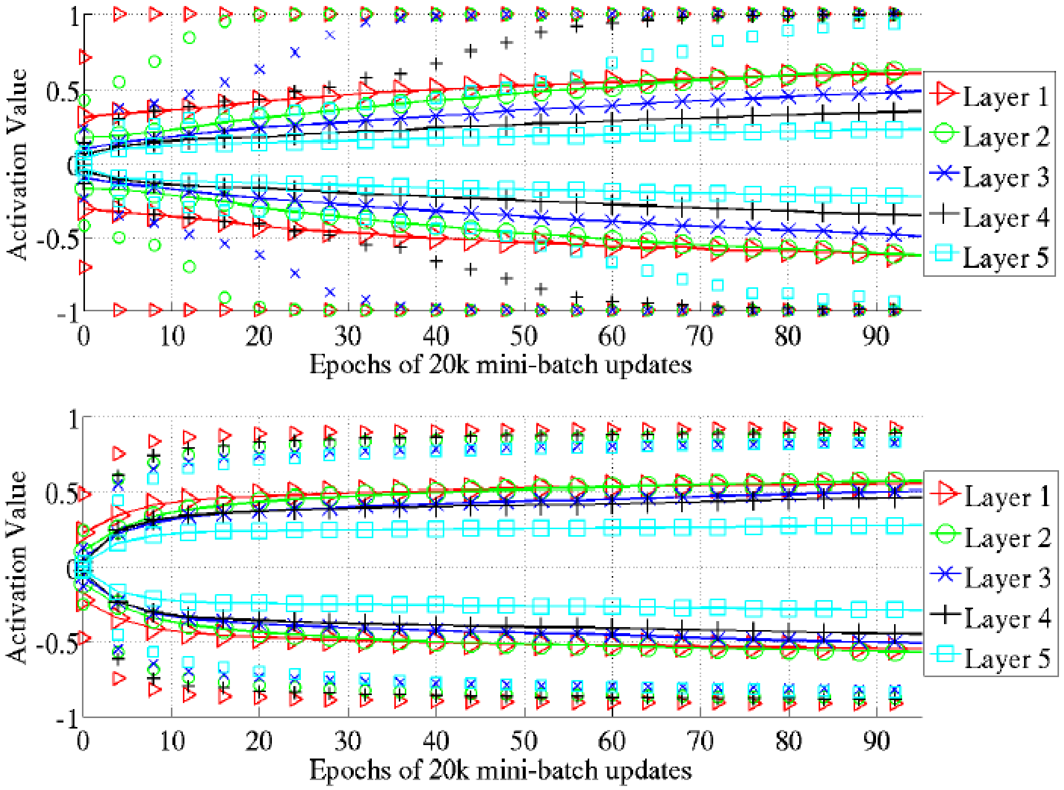


Do really those activations escape from such excessive saturation problem?

Effat et al, " Automatic Personality Traits Perception Using Asymmetric Auto-Encoder. "

Supervised learning with Activations: 2) tanh / softsign (symmetric)

Markers alone: activation value
With solid line: standard deviation
(Top: tanh // Down: softsign)



*** 5 layers are used in the experiments**

Using symmetric activation networks ..

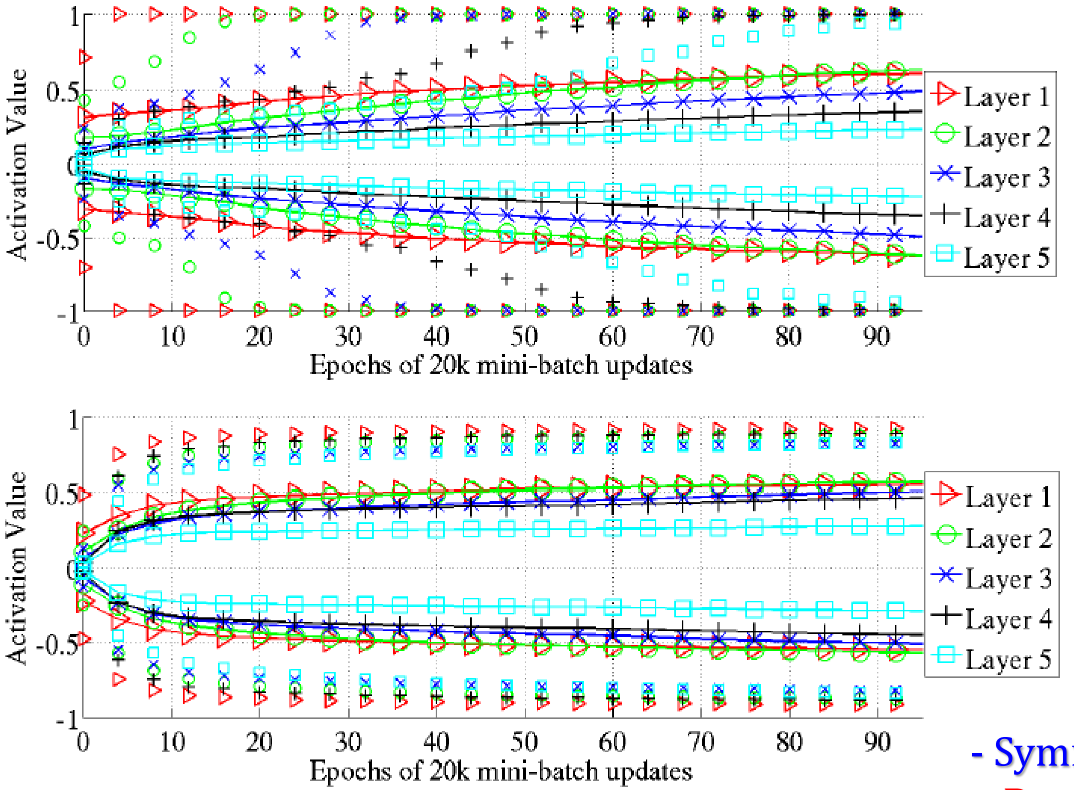
Observation1 :
The saturation behavior **is gone !!**

Observation2 :
Variation is getting smaller as go to deeper layers

Xavier et al, " Understanding the difficulty of training deep feedforward neural networks. " (2010).

Supervised learning with Activations: 2) tanh / softsign (symmetric)

Markers alone: activation value
With solid line: standard deviation
(Top: tanh // Down: softsign)



* **5 layers are used in the experiments**

Using symmetric activation networks ..

Observation1 :
The saturation behavior **is gone !!**

Observation2 :
Variation is getting smaller as go to deeper layers

- Symmetric activation : solve excessive saturation
- But **another problem** is observed
(variation is getting smaller) : Let's solve it with weight initialization !

Xavier et al, " Understanding the difficulty of training deep feedforward neural networks. " (2010).

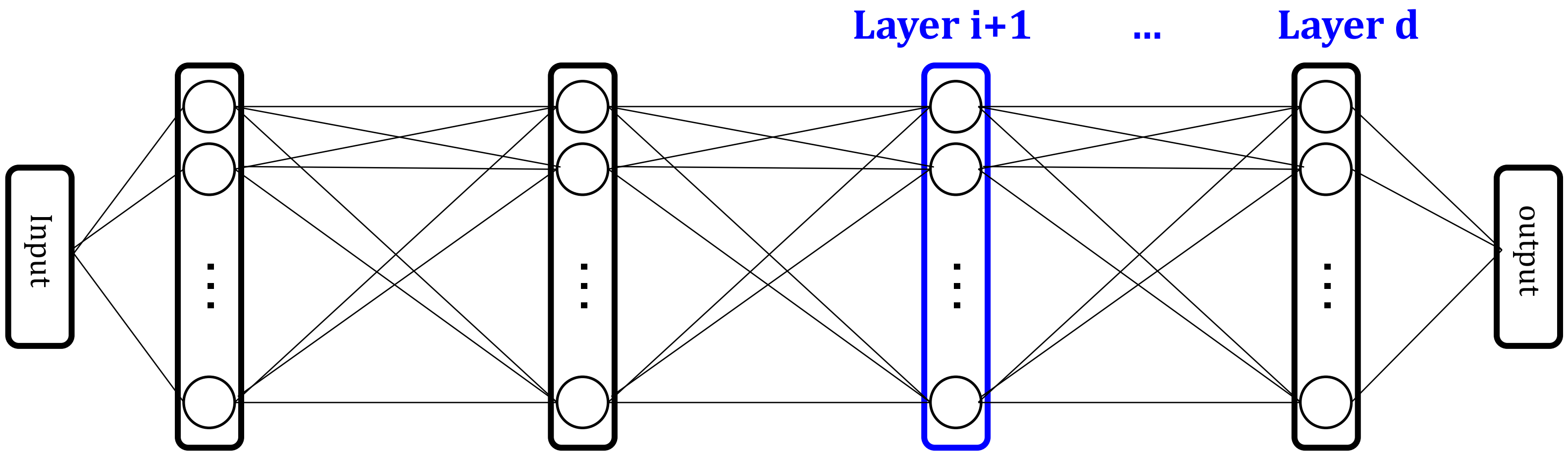
Studying Gradients and their Propagation

Some settings:

- Symmetric activation with unit derivative $(f'(0) = 1)$

- Considering initial stage : linear regime

- $\boldsymbol{s^i} = \boldsymbol{z^i W^i} + \boldsymbol{b^i}$
- $\boldsymbol{z}^{i+1} = f(\boldsymbol{s}^i)$
- i :layer index
  d: depth of the network



$\boldsymbol{s^i} = \boldsymbol{z^i W^i} + \boldsymbol{b^i}$
$\rightarrow \boldsymbol{z}^{i+1} = f(\boldsymbol{s}^i)$

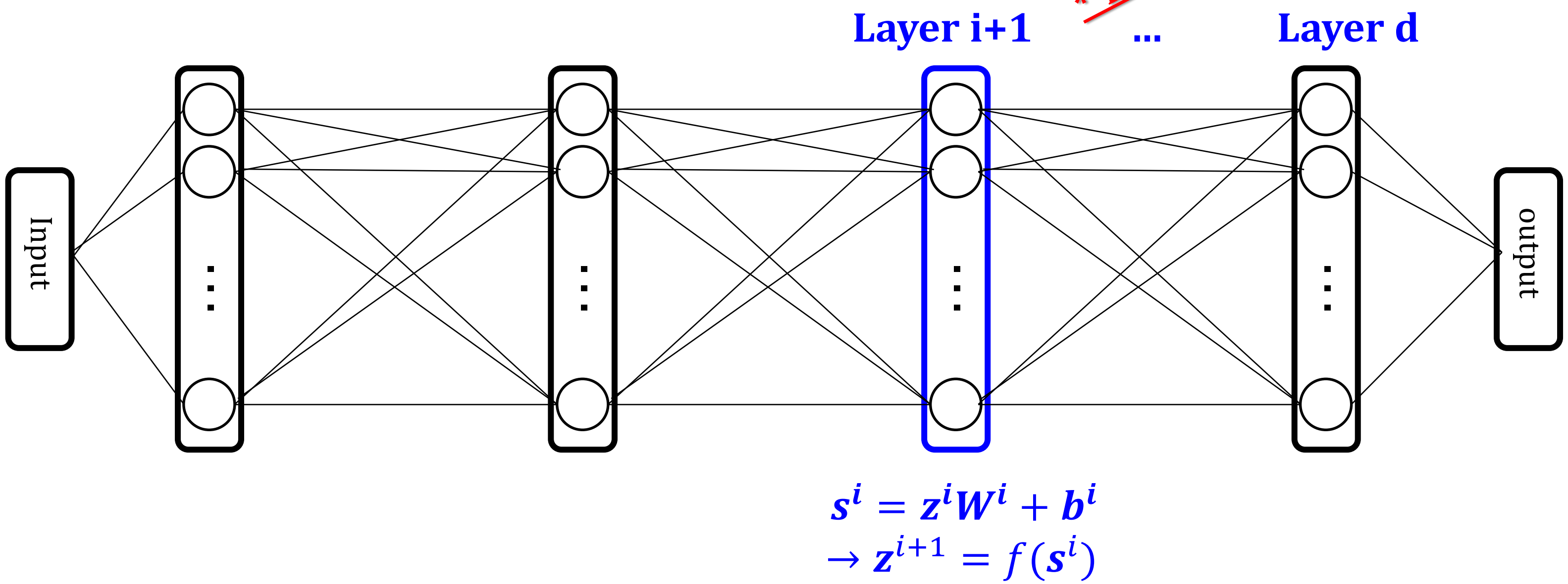
Studying Gradients and their Propagation

Some settings:

- Symmetric activation with unit derivative $(f'(0) = 1)$
- Considering initial stage : linear regime

- $\boldsymbol{s^i} = \boldsymbol{z^i W^i} + \boldsymbol{b^i}$
- $\boldsymbol{z}^{i+1} = f(\boldsymbol{s}^i)$
- i :layer index
  d: depth of the network

*** Do not stuck in a certain mathmatical derivation !**



Studying Gradients and their Propagation

Some settings:

- Symmetric activation with unit derivative $(f'(0) = 1)$
  - Considering initial stage : linear regime

- $\boldsymbol{s^i} = \boldsymbol{z^i W^i} + \boldsymbol{b^i}$
- $\boldsymbol{z}^{i+1} = f(\boldsymbol{s}^i)$
- i :layer index

d: depth of the network

$$\frac{\partial Cost}{\partial s_k^i} = f'(s_k^i) W_{k,\bullet}^{i+1} \frac{\partial Cost}{\partial \mathbf{s}^{i+1}}$$

$$\frac{\partial Cost}{\partial w_{l,k}^i} = z_l^i \frac{\partial Cost}{\partial s_k^i}$$

$$f'(s_k^i) \approx 1$$

in a linear regime at the initial stage

$$Var(z_j^l) = Var\left(\sum_{j=1}^{n_l} W_{jk}^l z_k^{l-1}\right)$$

$$Var[z^i] = Var[x] \prod_{i'=0}^{i-1} n_{i'} Var[W^{i'}]$$

Studying Gradients and their Propagation

Some settings:

- Symmetric activation with unit derivative $(f'(0) = 1)$
- Considering initial stage : linear regime

- $\boldsymbol{s^i} = \boldsymbol{z^i}\boldsymbol{W^i} + \boldsymbol{b^i}$
- $\boldsymbol{z^{i+1}} = f(\boldsymbol{s^i})$
- i :layer index

d: depth of the network

$$\frac{\partial Cost}{\partial s_k^i} = f'(s_k^i) W_{k,\bullet}^{i+1} \frac{\partial Cost}{\partial \mathbf{s}^{i+1}}$$

$$\frac{\partial Cost}{\partial w_{l,k}^i} = z_l^i \frac{\partial Cost}{\partial s_k^i}$$

$$f'(s_k^i) \approx 1$$

$$Var[z^i] = Var[x] \prod_{i'=0}^{i-1} n_{i'} Var[W^{i'}]$$

Expand with chain rule

$$Var\left[\frac{\partial Cost}{\partial s^i}\right] = Var\left[\frac{\partial Cost}{\partial s^d}\right] \prod_{i'=i}^{d} n_{i'+1} Var[W^{i'}] \qquad (1)$$

Variance of the back-propagated gradient

$$Var\left[\frac{\partial Cost}{\partial w^i}\right] = \prod_{i'=0}^{i-1} n_{i'} Var[W^{i'}] \prod_{i'=i}^{d-1} n_{i'+1} Var[W^{i'}] \times Var[x] Var\left[\frac{\partial Cost}{\partial s^d}\right]. \qquad (2)$$

Variance of the gradient on the weights

Remember these

Studying Gradients and their Propagation

$$Var\left[\frac{\partial Cost}{\partial s^i}\right] = Var\left[\frac{\partial Cost}{\partial s^d}\right] \prod_{i'=i}^{d} \boxed{n_{i'+1}} Var[W^{i'}] \quad \textbf{(1)}$$

$$Var\left[\frac{\partial Cost}{\partial w^i}\right] = \prod_{i'=0}^{i-1} n_{i'} Var[W^{i'}] \prod_{i'=i}^{d-1} \boxed{n_{i'+1}} Var[W^{i'}] \times Var[x] Var\left[\frac{\partial Cost}{\partial s^d}\right]. \quad \textbf{(2)}$$

Now let's consider the case when **all layers have same width *n*, and weight variance is shared.** then **(1), (2)** become:

$$\forall i, Var\left[\frac{\partial Cost}{\partial s^i}\right] = \left[n Var[W]\right]^{d-i} Var[x]$$

$$\forall i, Var\left[\frac{\partial Cost}{\partial w^i}\right] = \left[n Var[W]\right]^{d} Var[x] Var\left[\frac{\partial Cost}{\partial s^d}\right]$$

Studying Gradients and their Propagation

$$\forall i, Var\left[\frac{\partial Cost}{\partial w^i}\right] = \left[nVar[W]\right]^d Var[x] Var\left[\frac{\partial Cost}{\partial s^d}\right]$$

$$\forall i, Var\left[\frac{\partial Cost}{\partial s^i}\right] = \left[nVar[W]\right]^{d-i} Var[x]$$

Interesting properties:
with symmetric activations,

1) *Variance of the gradient on the weights* is the same for all layers
2) *Variance of the back propagated gradient* still **vanish or explode** as for **different layers.**

Studying Gradients and their Propagation

REMIND : Our aim (what we would like to be):

- From a **forward-propagation** view, to keep information flowing,

$$\forall (i, i'), Var[z^i] = Var[z^{i'}].$$

- From a **back-propagation** view, to keep information flowing

$$\forall (i, i'), Var\left[\frac{\partial Cost}{\partial s^i}\right] = Var\left[\frac{\partial Cost}{\partial s^{i'}}\right].$$

$$Var[z^i] = Var[x] \prod_{i'=0}^{i-1} n_{i'} Var[W^{i'}]$$

$$Var\left[\frac{\partial Cost}{\partial s^i}\right] = Var\left[\frac{\partial Cost}{\partial s^d}\right] \prod_{i'=i}^{d} n_{i'+1} Var[W^{i'}]$$

The both conditions transform to

$$\forall i, \quad n_i Var[W^i] = 1$$

$$\forall i, \quad n_{i+1} Var[W^i] = 1$$

The objective

A compromise between these

$$Var[W^i] = \frac{2}{n_i + n_{i+1}}$$

Studying Gradients and their Propagation

Wait ! Small note:

Name of the probability distribution	Probability distribution function	Mean	Variance
Binomial distribution	$\Pr(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$	np	$np(1-p)$
Geometric distribution	$\Pr(X=k) = (1-p)^{k-1} p$	$\frac{1}{p}$	$\frac{(1-p)}{p^2}$
Normal distribution	$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	μ	σ^2
Uniform distribution (continuous)	$f(x \mid a, b) = \begin{cases} \frac{1}{b-a} & \text{for } a \le x \le b, \\ 0 & \text{for } x < a \text{ or } x > b \end{cases}$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
Exponential distribution	$f(x \mid \lambda) = \lambda e^{-\lambda x}$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$
Poisson distribution	$f(k \mid \lambda) = \frac{e^{-\lambda}\lambda^k}{k!}$	λ	λ

https://en.wikipedia.org/wiki/Variance

Studying Gradients and their Propagation

The conventional initialization

$$W \sim U\left(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right)$$

→

$$nVar[W] = \frac{1}{3}$$

The objective

$$Var[W^i] = \frac{2}{n_i + n_{i+1}}$$

Match !

The normalized (Xavier/glorot) initialization

$$W \sim U\left[- \; ? \;, \; ? \;\right]$$

→

$$Var[W^i] = \frac{2}{n_i + n_{i+1}}$$

Studying Gradients and their Propagation

The conventional initialization

$$W \sim U\left(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right)$$

→

$$nVar[W] = \frac{1}{3}$$

The objective

$$Var[W^i] = \frac{2}{n_i + n_{i+1}}$$

Match !

The normalized (Xavier/glorot) initialization

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

→

$$Var[W^i] = \frac{2}{n_i + n_{i+1}}$$

Studying Gradients and their Propagation

The normalized (Xavier/glorot) initialization

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right] \rightarrow Var[W^i] = \frac{2}{n_i + n_{i+1}}$$

Now, using the normalized initialization,

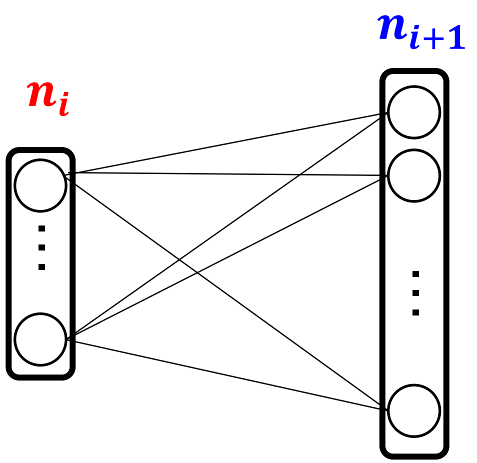
Objective of **maintaining** activation variances and back-propagated gradients variance satisfied.

→ Now the meaningful gradient still flows even in the deeper networks without losing variance

Studying Gradients and their Propagation

The normalized (Xavier/glorot) initialization

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right] \rightarrow Var[W^i] = \frac{2}{n_i + n_{i+1}}$$



Another small note!

Here, n_i is called as **fan-in**
& n_{i+1} is called as **fan-out**

Evaluation

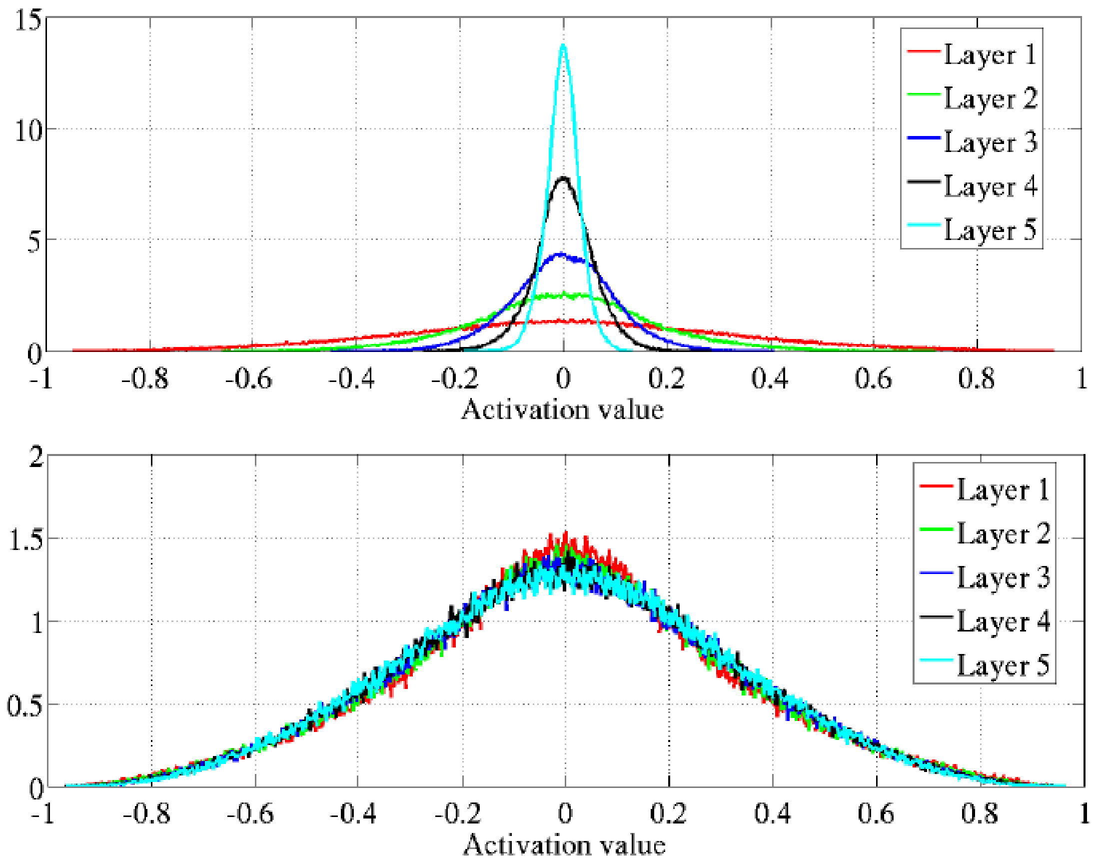


Figure 6: Activation values normalized histograms with hyperbolic tangent activation, with standard (top) vs normalized initialization (bottom). Top: 0-peak increases for higher layers.

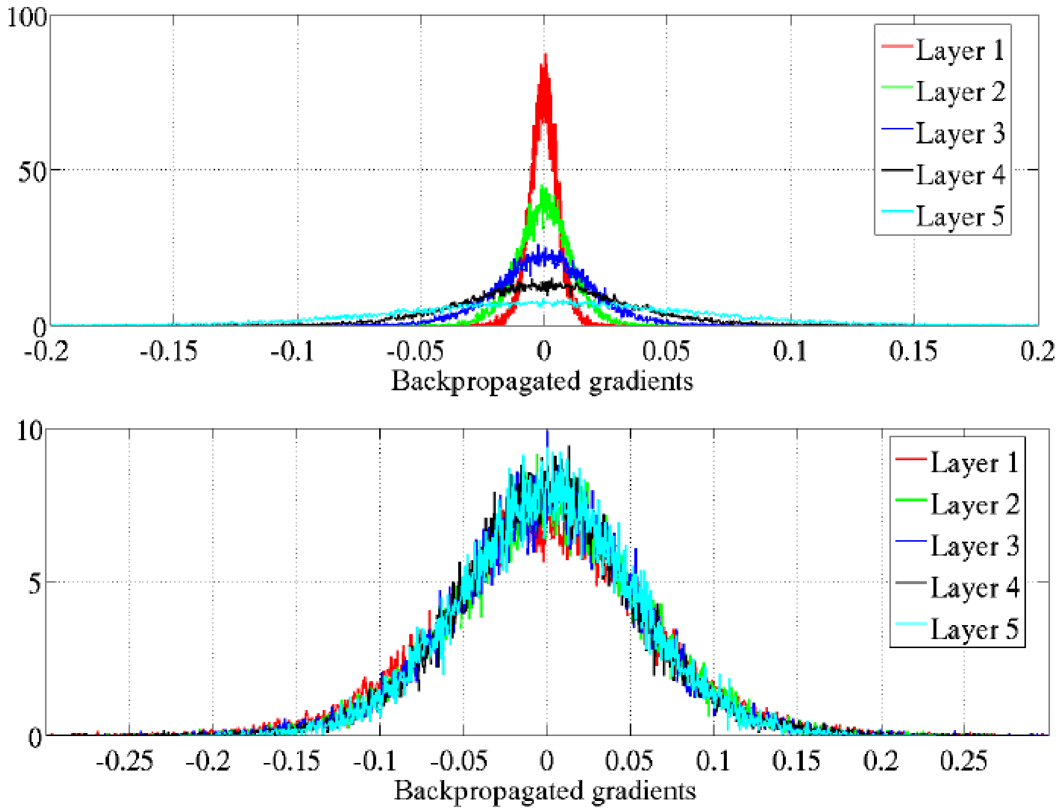


Figure 7: Back-propagated gradients normalized histograms with hyperbolic tangent activation, with standard (top) vs normalized (bottom) initialization. Top: 0-peak decreases for higher layers.

Xavier et al, " Understanding the difficulty of training deep feedforward neural networks. " (2010).

Two milestone initialization techniques

1. Xavier (Glorat) initialization

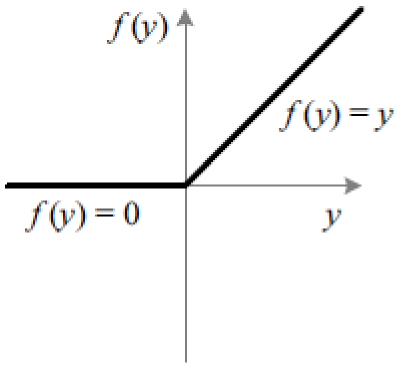
Glorot Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." JMLR Workshop and Conference Proceedings, 2010.

2. Kaiming (He) initialization

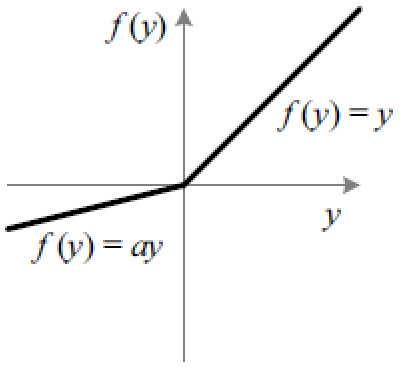
He Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." ICCV, 2015.

Propositions of the paper

Proposition 1.
Parametric Rectifier



ReLU



PReLU

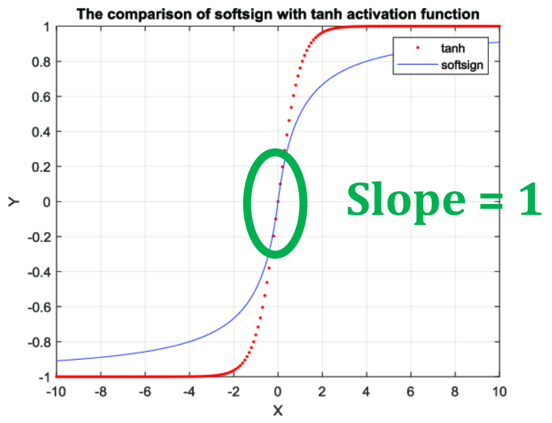
Proposition 2.
Initialization technique for ReLU

*** What we are to focus on in this presentation**

Motivation

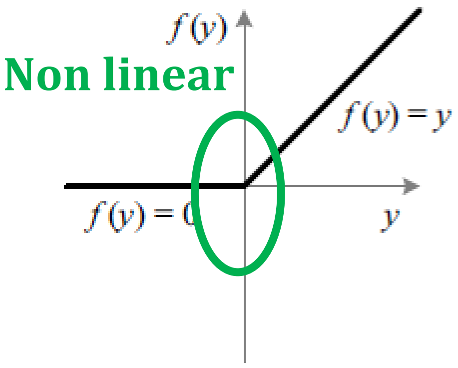
Xavier derivation is based on linear assumption and initial stage

Their symmetric activations



<u>However, It is invalid for ReLU</u>

ReLU



Derivation

- Derivation mainly follows Xavier init.

- But they start from deep **CNN** whose weights drawn from **Gaussian** distribution

Basic notations

$$\mathbf{y}_l = \mathrm{W}_l \mathbf{x}_l + \mathbf{b}_l$$

$$\mathbf{x}_l = f(\mathbf{y}_{l-1})$$

c :input channels
k: filter size
d: filter number ($c_l = d_{l-1}$)
$n = k^2 c$

R. L. Watrous and G. M. Kuhn. Some Considerations on the Training of Recurrent Neural Networks for Time-Varying Signals. In M. Gori (ed.), Second Workshop on Neural Networks for Speech Processing, pp. 5{17, Trieste, Italy, 1993. Universit?a di Firenze, Edizioni LINT Trieste S.r.l.

Derivation

- Derivation mainly follows Xavier init.

- But they start from deep **CNN** whose weights drawn from **Gaussian** distribution

***B*asic notations**

$$\mathbf{y}_l = \mathrm{W}_l \mathbf{x}_l + \mathbf{b}_l$$

$$\mathbf{x}_l = f(\mathbf{y}_{l-1})$$

c :input channels
k: filter size
d: filter number ($c_l = d_{l-1}$)
$\mathrm{n} = \mathrm{k}^2\mathrm{c}$

*** As derivation is basically same with Xavier, will further skip details here.**

R. L. Watrous and G. M. Kuhn. Some Considerations on the Training of Recurrent Neural Networks for Time-Varying Signals. In M. Gori (ed.), Second Workshop on Neural Networks for Speech Processing, pp. 5{17, Trieste, Italy, 1993. Universit?a di Firenze, Edizioni LINT Trieste S.r.l.

Forward propagation case

$$\mathbf{y}_l = \mathrm{W}_l \mathbf{x}_l + \mathbf{b}_l$$
$$\mathbf{x}_l = f(\mathbf{y}_{l-1})$$

***Step* 1.**

As elements in x_l, W_l is mutually independent

$$Var[y_l] = n_l Var[w_l x_l] \quad (1)$$

Zero mean w, *variance of product of independent variables* gives

$$Var[y_l] = n_l Var[w_l] E[x_l^2] \quad (2)$$

***Step* 2.**

Let $\mathrm{w}_{\mathrm{l}-1}$ have symmetric dist. around zero and $\mathrm{b}_{\mathrm{l}-1} = 0$
(can be achieved by the initialization), if f(.) is ReLU,

$$E[x_l^2] = \frac{1}{2} Var[y_{l-1}] \quad (3)$$

***Step* 3.**

Put (3) into (2) and consider L layer

$$Var[y_L] = Var[y_1] \left(\prod_{l=2}^{L} \frac{1}{2} n_l Var[w_l] \right)$$

Forward propagation case

$$Var[y_L] = Var[y_1]\left(\prod_{l=2}^{L}\frac{1}{2}n_l Var[w_l]\right)$$

If this term is 1, variance is consistent over layers = $\frac{1}{2}n_l Var[w_l] = 1$

It leads to Gaussian with 0 mean, $\sqrt{\frac{2}{n_l}}$ std. (bias=0)

This is the Kaiming initialization !

(derived from forward-propagation)

Backward propagation case

Consider gradients with notations

$$\Delta \mathbf{x}_l = \hat{\mathrm{W}}_l \Delta \mathbf{y}_l.$$

Skip derivational details
(The procedure is similar)

$$Var[\Delta x_l] = Var[\Delta x_{L+1}] \left(\prod_{l=2}^{L} \frac{1}{2} \hat{n}_l Var[w_l] \right)$$

Should be '1'

It leads to Gaussian with 0 mean, $\sqrt{\frac{2}{\hat{n}_l}}$ std. (bias=0)

This is also the Kaiming initialization !
(derived from backward-propagation)

Forward eq. vs Backward eq.

Kaiming He init.

	Forward case	**Backward case**	
Standar deviation in Gaussian	$\sqrt{\frac{2}{n_l}}$	$\sqrt{\frac{2}{\hat{n}_l}}$	* $c_l = d_{l-1}$
	$(\mathrm{n}_l = k_l^2 c_l)$	$(\hat{\mathrm{n}}_l = k_l^2 d_l)$	

Forward eq. vs Backward eq.

Kaiming He init.

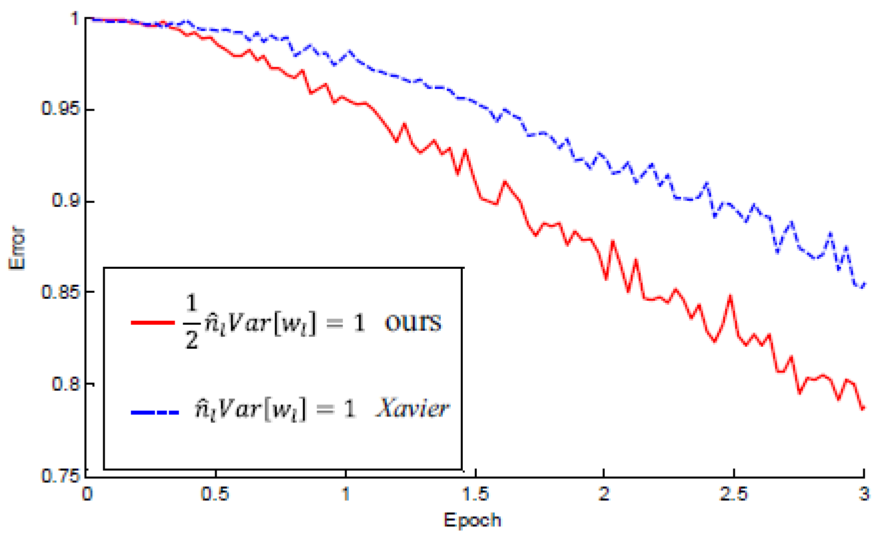
	Forward case	**Backward case**	
Standar deviation in Gaussian	$\sqrt{\frac{2}{n_l}}$	$\sqrt{\frac{2}{\hat{n}_l}}$	* $c_l = d_{l-1}$
	$(\mathrm{n}_l = k_l^2 c_l)$	$(\hat{\mathrm{n}}_l = k_l^2 d_l)$	

It is sufficent to use **one of the two.**

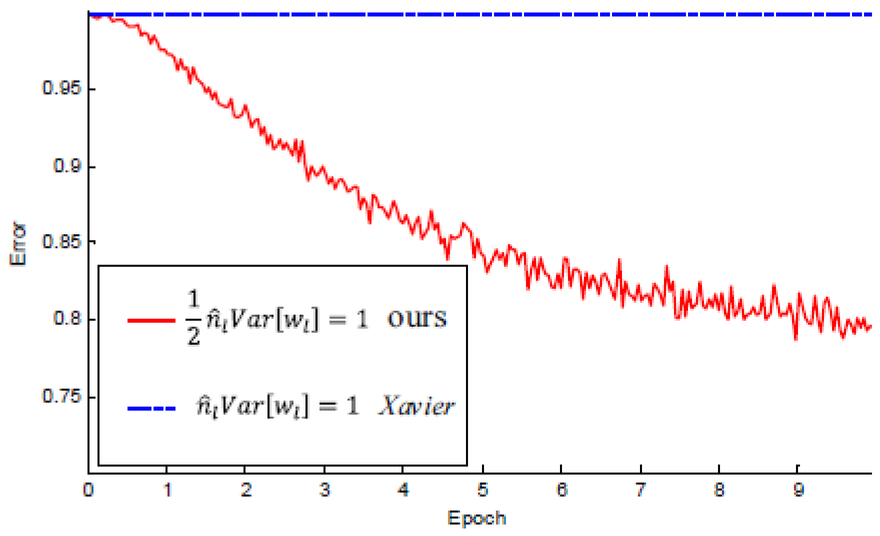
using $\sqrt{\frac{2}{\hat{n}_l}}$ as std for example, $\prod_{l=2}^{L} \frac{1}{2} n_l Var[w_l]$ in forward case equation become $\frac{c_2}{d_L}$, which is not a diminishing number

The results

ImageNet classification task



CNN with 22 layer



CNN with 30 layer

➡ Gradient diminishing in Xavier

Discussion

- Initialization and activation should be PAIRED

Xavier : symmetric (tanh, softsign)
Kaiming : ReLU-like

- Use Kaiming for extremely deeper networks.

- For Kaiming initialization, excessive increase/decrease of number of filters (or channels) in CNN may be undesirable (as variance preservation doesn't hold for back and forward at the same time)

Thank you !

Appendix A. variance of product

If two variables X and Y are independent, the variance of their product is given by

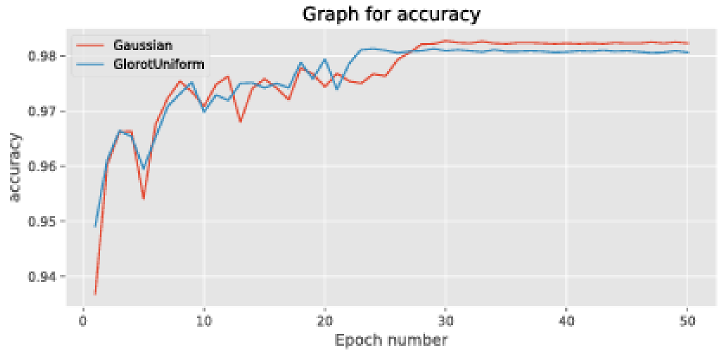
$$\operatorname{Var}(XY) = [\operatorname{E}(X)]^2 \operatorname{Var}(Y) + [\operatorname{E}(Y)]^2 \operatorname{Var}(X) + \operatorname{Var}(X) \operatorname{Var}(Y).$$

Equivalently, using the basic properties of expectation, it is given by

$$\operatorname{Var}(XY) = \operatorname{E}\left(X^2\right) \operatorname{E}\left(Y^2\right) - [\operatorname{E}(X)]^2 [\operatorname{E}(Y)]^2.$$

Appendix B. Gaussian vs Uniform

(a) validation error



(b) validation accuracy

Note 1. [1] compared a Gaussian distribution to a uniform distribution and found differences on the conditioning of the Jacobian matrix of a neural network, **but found no relation to the convergence speed**

Note 2. Extensive experiments are given in [2] (seems no difference for me)

[1] R. L. Watrous and G. M. Kuhn. Some Considerations on the Training of Recurrent Neural Networks for Time-Varying Signals. In M. Gori (ed.), Second Workshop on Neural Networks for Speech Processing, pp. 5{17, Trieste, Italy, 1993. Universit?a di Firenze, Edizioni LINT Trieste S.r.l.

[2] Pedamonti, Dabal. "Comparison of non-linear activation functions for deep neural networks on MNIST classification task." arXiv preprint arXiv:1804.02763 (2018).