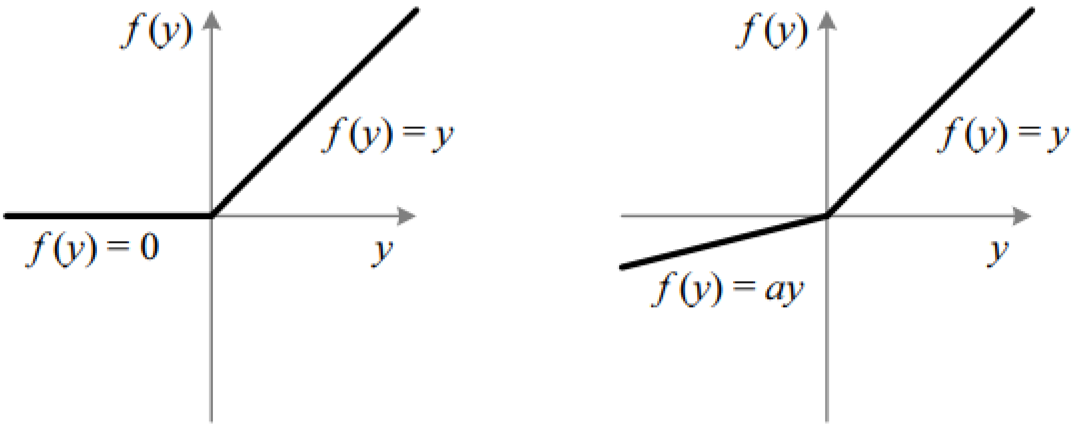
Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

URL	Delving Deep into Rectifiers.pdf
Status	Done

1. Introduction

- ReLU를 parametric하게 만들어서 사용
- ReLU에 맞는 weight 초기화 방법을 사용
- ImageNet으로 실험 진행

2. Parametric Rectifiers



- 기존 ReLU는 0으로 가기 때문에 dying ReLU, 가중치가 없는 문제
- PReLU는 0이 음수일 때 ay 값을 사용해서 성능을 더 좋게 만든 방법

- Generic form of Rectifier Linear Function:

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \le 0 \end{cases}.$$

- ReLU: $a_i = 0$, $f(x_i) = max(0, x_i)$
- PReLU: when a_i is a learnable parameter, $f(x_i) = max(0, x_i) + a_i max(0, x_i)$
  → 음수 부분에 학습 가능한 a_i
- LReLU(Leaky ReLU):$a_i = 0.01$, $f(x_i) = max(0, x_i) + 0.01max(0, x_i)$

Optimization

- PReLU can be trained using backpropagation and optimized simultaneously with other layers.

$$\frac{\partial \mathcal{E}}{\partial a_i} = \sum_{y_i} \frac{\partial \mathcal{E}}{\partial f(y_i)} \frac{\partial f(y_i)}{\partial a_i}$$

- The gradient of the activation is given by:

$$\frac{\partial f(y_i)}{\partial a_i} = \begin{cases} 0, & \text{if } y_i > 0 \\ y_i, & \text{if } y_i \le 0 \end{cases}$$

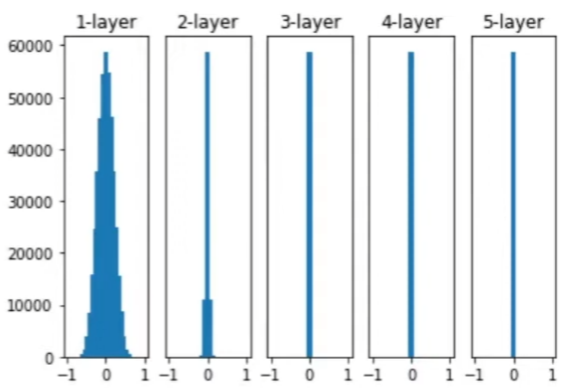
- Momentum method when updating a_i

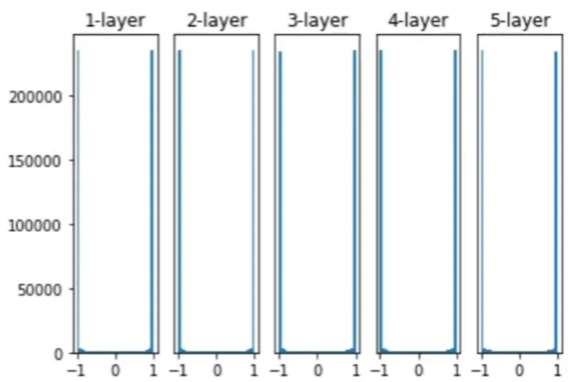
$$\Delta a_i := \mu \Delta a_i + \epsilon \frac{\partial \mathcal{E}}{\partial a_i}$$

3. Initialization of Filter Weights for Rectifiers

Weight initialization

- **Zero or same value**
    - It sends out the same output value of all neurons.
    - At the backward propagation stage, each neuron has the same gradient value.
    - If the weights are the same, they work as if they were one regardless of the number of neurons.





- **Small random values**
    - If the initial value of the weights is large, gradient vanishing occurs because it converges to 0 and 1.
    - If ReLU is large, dead ReLU problem occurs when negative.
    - Therefore, the initial value of the weights must be initialized small to be random without having the same initial value.
    - Typically, we randomly initialize to a value that follows a normal distribution with an average initial value of 0, standard deviation of 0.01.
    - In shallow neural networks, there is no problem, but the deeper the problem becomes.
    - If the right plot randomly initializes to a normal distribution with an average of 0, standard deviation of 1, and tanh is used as an activation function, the gradient vanishing problem occurs as the output is concentrated to -1 and 1.
    - In conclusion, methods to initialize with small random numbers can also be said to be unsuitable in DNNs.

- Xavier Normal Initialization

$$W \sim N(0, Var(W)) \ , \ Var(W) = \sqrt{\frac{2}{n_{input} + n_{output}}}$$

- Xavier Uniform Initialization

$$W \sim U = \left(\sqrt{\frac{6}{n_{input} + n_{output}}}, + \sqrt{\frac{6}{n_{input} + n_{output}}}\right)$$

- “Understanding the difficulty of training deep feedforward neural networks”
- With fixed standard deviations (e.g., 0.01), very deep models (e.g., >8 conv layers) have difficulties to converge, as reported by the VGG team and also observed in our experiments.
- Its derivation is based on the assumption that the activations are linear. This assumption is invalid for ReLU and PReLU.

Forward Propagation

$y_l = W_l x_l + b_l$
$x_l = f(y_{l-1})$
$Y = W_1 x_1 + W_2 x_2 + \cdots + W_n x_n$

x_l : mutually independent, same distribution
x_l, W_l : independent
n : size of layer
$E[w_l] = 0$

$Goal : Var[y_l] = n_l Var[w_l x_l]$

$$Var[w_l x_l] = E[w_l^2]E[x_l^2] - [E[x_l]]^2 [\cancelto{0}{E[w_l]}]^2$$
$$Var[w_l x_l] = E[w_l^2]E[x_l^2]$$
$$Var[w_l] = E[w_l^2] - [\cancelto{0}{E[w_l]}]^2$$
$$Var[w_l] = E[w_l^2]$$
$$Var[w_l x_l] = E[w_l^2]E[x_l^2] = Var[w_l]E[x_l^2]$$

$Goal : Var[y_l] = n_l Var[w_l x_l] = n_l Var[w_l] E[x_l^2]$

$Goal : Var[y_l] = n_l Var[w_l x_l] = n_l Var[w_l] E[x_l^2]$

w_{l-1} = symmetric distribution around zero and $b_{l-1} = 0$
y_{l-1} = zero mean and symmetric distribution around zero

$$E(x_l^2) = \frac{1}{\sigma_{y_{l-1}}\sqrt{2\pi}} \int_0^{\infty} x^2 e^{-\frac{x^2}{2\sigma_{y_{l-1}}^2}} dx$$
$$= \left(\frac{1}{2}\right) \frac{1}{\sigma_{y_{l-1}}\sqrt{2\pi}} \int_{-\infty}^{\infty} x^2 e^{-\frac{x^2}{2\sigma_{y_{l-1}}^2}} dx$$
$$= \frac{\sigma_{y_{l-1}}^2}{2} = \frac{1}{2} Var(y_{l-1})$$

$Goal : Var[y_l] = n_l Var[w_l] E[x_l^2] = \frac{1}{2} n_l Var[w_l] Var[y_{l-1}]$

Backward Propagation

$$\Delta x_l = \widehat{W}_l \Delta y_l$$
$$\Delta y_l = f'(y_l)\ \Delta x_{l+1}$$

$x_l, \Delta y_l$: *independent*
Δx_l : *zero mean for all l*
$f'(y_l)$: *For the ReLU case, zero or one and their probabilities are equal*
$f'(y_l), \Delta x_{l+1}$: *independent of each other*

$$Goal: Var\Delta x_l = \hat{n}_l Var[x_l] Var[\Delta y_l]$$

$$E[\Delta y_l] = E[f'(y_l)\Delta x_{l+1}] = E[f'(y_l)]E[\Delta x_{l+1}]$$
$$= \frac{1}{2}E[\Delta x_{l+1}] = 0$$

$$Var(w_l \cdot \Delta y_l) = [\cancelto{0}{E(w_l)}]^2 Var(\Delta y_l) + [\cancelto{0}{E(\Delta y_l)}]^2 Var(w_l) + Var(w_l)Var(\Delta y_l)$$

$$Var[\Delta y_l] = Var[f'(y_l)\ \Delta x_{l+1}] = E[(f'(y_l))^2]E[(\Delta x_{l+1})^2] - [E[f'(y_l)]]^2[\cancelto{0}{E[\Delta x_{l+1}]}]^2 = E\left[(f'(y_l))^2\right]E[(\Delta x_{l+1})^2] - 0$$
$$= E\left[(f'(y_l))^2\right]E[(\Delta x_{l+1})^2] = \frac{1}{2}E[(\Delta x_{l+1})^2]$$

$$Goal: Var[\Delta x_l] = \hat{n}_l Var[x_l] Var[\Delta y_l]$$
$$Var(\Delta y_l) = \frac{1}{2}E[(\Delta x_{l+1})^2] = Var[x_{l+1}]$$

$$Goal: Var\Delta x_l = \frac{1}{2}\hat{n}_l Var[x_l] Var[x_{l+1}]$$
$$\frac{1}{2}\hat{n}_l Var[x_l] = 1, \forall l.$$

Normal Initialization(backward)

$$W \sim N(0, \sqrt{\frac{2}{n_l}})\ and\ b = 0$$

Normal Initialization(forward)

$$W \sim N(0, \sqrt{\frac{2}{n_l}})\ and\ b = 0$$

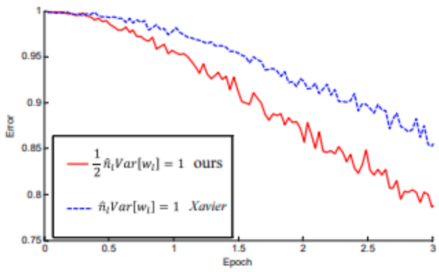


Figure 2. The convergence of a **22-layer** large model (B in Table 3). The x-axis is the number of training epochs. The y-axis is the top-1 error of 3,000 random val samples, evaluated on the center crop. We use ReLU as the activation for both cases. Both our initialization (red) and "*Xavier*" (blue) [7] lead to convergence, but ours starts reducing error earlier.

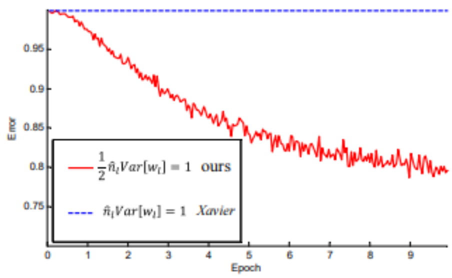


Figure 3. The convergence of a **30-layer** small model (see the main text). We use ReLU as the activation for both cases. Our initialization (red) is able to make it converge. But "*Xavier*" (blue) [7] completely stalls - we also verify that its gradients are all diminishing. It does not converge even given more epochs.

- Relu를 사용할 때, Xavier보다 더 좋은 성능이 나옴

4. Experiments on ImageNet

- Dataset
    - 100-calss ImageNet 2012 dataset
    - 1.2M training images, 40K validation images, 100K test images

- Results measured by top-1/top-5 error rates

	team	top-1	top-5
in competition ILSVRC 14	MSRA [11]	27.86	9.08†
	VGG [25]	-	8.43†
	GoogLeNet [29]	-	7.89
post-competition	VGG [25] (arXiv v2)	24.8	7.5
	VGG [25] (arXiv v5)	24.4	7.1
	Baidu [32]	24.88	7.42
	MSRA (A, ReLU)	24.02	6.51
	MSRA (A, PReLU)	22.97	6.28
	MSRA (B, PReLU)	22.85	6.27
	MSRA (C, PReLU)	**21.59**	**5.71**

Table 6. The **single-model** results for ImageNet 2012 val set. †: Evaluated from the test set.

	team	top-5 (**test**)
in competition ILSVRC 14	MSRA, SPP-nets [11]	8.06
	VGG [25]	7.32
	GoogLeNet [29]	6.66
post-competition	VGG [25] (arXiv v5)	6.8
	Baidu [32]	5.98
	MSRA, PReLU-nets	**4.94**

Table 7. The **multi-model** results for the ImageNet 2012 test set.