



# Delving deep into rectifiers: Surpassing human-level performance on imagenet classification



He, Kaiming, et al.

# Abstract

we study rectifier neural networks for image classification from two aspects.

**1.** we propose a **Parametric Rectified Linear Unit (PReLU)** that generalizes the traditional rectified unit.

-> improves model fitting with nearly zero extra computational cost and little overfitting risk.

**2.** we derive a robust initialization method that particularly considers the rectifier nonlinearities.

-> enables us to train extremely deep rectified models directly from scratch and to investigate deeper or wider network architectures

Our result is the first to surpass the reported **human-level performance** (5.1%, [26]) on this dataset.

# Introduction

Rectified Linear Unit (ReLU), is one of several keys to the recent success of deep networks.

- > It expedites convergence of the training procedure and leads to better solutions than conventional sigmoid like units

Unlike traditional sigmoid-like units, ReLU is not a symmetric function.

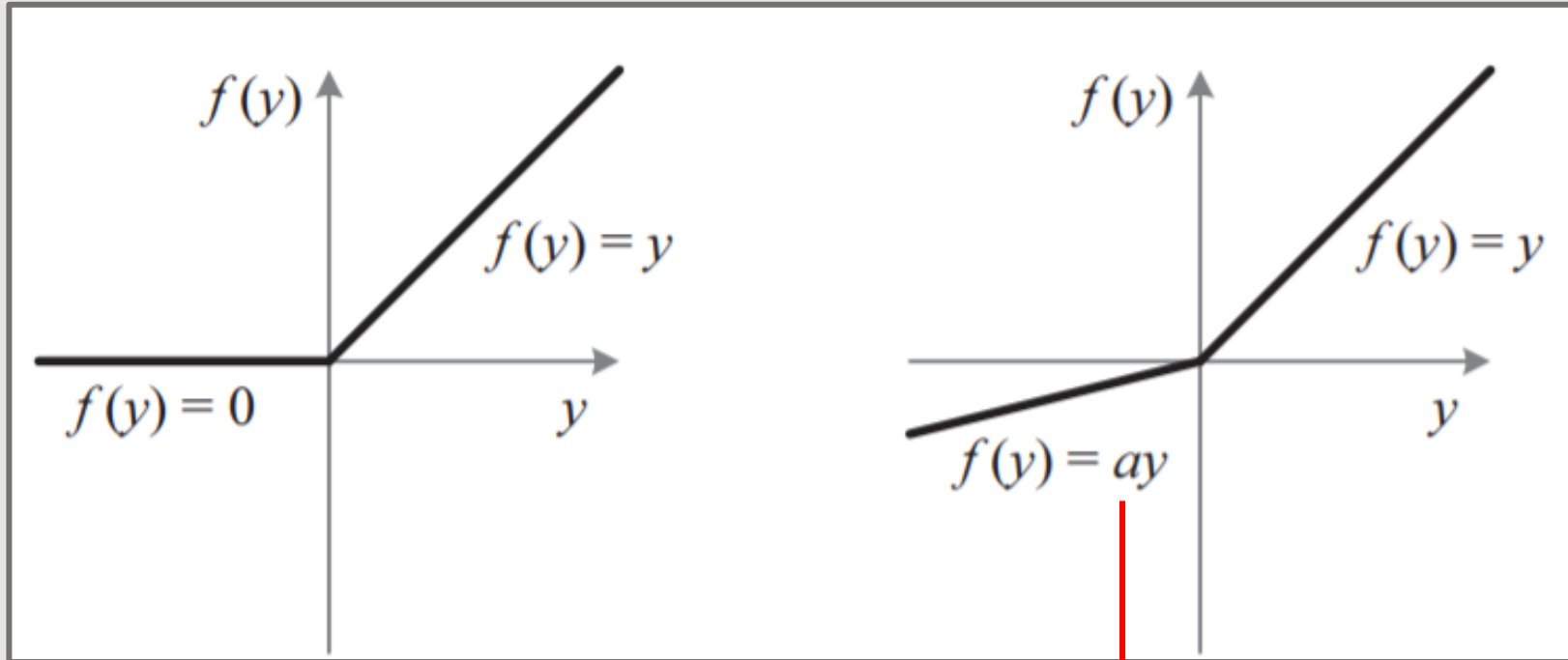
- > the mean response of ReLU is always no smaller than zero
- Even assuming the inputs/weights are subject to symmetric distributions, the distributions of responses can still be asymmetric because of the behavior of ReLU.

# Introduction

We investigate neural networks from two aspects

1. we propose a new extension of ReLU, which we call **Parametric Rectified Linear Unit (PReLU)**
2. we study the difficulty of training rectified models that are very deep. By explicitly modeling the nonlinearity of rectifiers (ReLU/PReLU), we derive a theoretically sound initialization method, which helps with convergence of very deep models

# Introduction



ReLU vs PReLU

If  $a = 0.01$ :  
Leaky ReLU

# Parametric Rectifiers

We show that replacing the parameter-free ReLU by a learned activation unit improves classification accuracy.


PReLU introduces a very small number of extra parameters.

- The number of extra parameters is equal to the total number of channels, which is negligible when considering the total number of weights.
- > we expect no extra risk of overfitting.

# Parametric Rectifiers

The gradient of  $a_i$  for one layer is:

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


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

The time complexity due to PReLU is negligible for both forward and backward propagation

# Parametric Rectifiers

We adopt the momentum method when updating  $a_i$  :

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

Momentum

Learning  
rate

Initialization :  $a_i=0.25$

we do not use weight decay when updating  $a_i$ .

A weight decay tends to push  $a_i$  to zero, and thus biases PReLU toward ReLU.



# Initialization of Filter Weights for Rectifiers

Rectifier networks are easier to train,  
But a bad initialization can still hamper the learning of  
a highly non-linear system  
-> we propose a robust initialization method that removes an obstacle of  
training extremely deep rectifier networks.

## Xavier initialization

Glorot and Bengio [8] proposed to adopt a properly scaled uniform  
distribution for initialization.

-> Its derivation is based on the assumption that the activations are linear.

→ This assumption is **invalid for ReLU and PReLU**.

$$Var(W_i) = \frac{2}{n_{in} + n_{out}}$$

→ "He" initialization

# Initialization of Filter Weights for Rectifiers

## Forward Propagation Case

The central idea is to investigate the variance of the responses in each layer. For a conv layer, a response is:

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

$\mathbf{x} : k^2 c$  by 1vector (k by k pixels in c input channel)

$n = k^2 c$

$\mathbf{W} : d$  by  $n$  (d : the number of filter)

$\mathbf{b} : \text{bias}$

$y : \text{response at a pixel of the output map}$

# Initialization of Filter Weights for Rectifiers

We let the initialized elements in  $W_l$  be independent and identically distributed (i.i.d.)

- $x_l$  are also i.i.d
- $x_l$  and  $W_l$  are independent of each other

$$\text{Var}[y_l] = n_l \text{Var}[w_l x_l],$$

the variance of the product of independent variables gives us

$$\text{Var}[y_l] = n_l \text{Var}[w_l] E[x_l^2].$$

# Initialization of Filter Weights for Rectifiers

let  $W_{l-1}$  have a symmetric distribution around zero and  $b_{l-1} = 0$ ,  
then  $y_{l-1}$  has zero mean and has a symmetric distribution around zero

$$\longrightarrow E[x_l^2] = \frac{1}{2} \text{Var}[y_{l-1}] \quad \text{when } f \text{ is ReLU}$$

$$\text{Var}[y_l] = \frac{1}{2} n_l \text{Var}[w_l] \text{Var}[y_{l-1}].$$

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

Key to the initialization design

Sufficient condition

zero-mean Gaussian distribution whose standard deviation (std) is  $\sqrt{2/n_l}$

# Initialization of Filter Weights for Rectifiers

## Backward Propagation Case

The gradient of a conv layer is computed by:

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

Note that  $\hat{n} \neq n = k^2 c$ .

$W_l$  and  $\Delta y_l$  are independent of each other, then  $\Delta x_l$  has zero mean for all  $l$ , when  $W_l$  is initialized by a symmetric distribution around zero

# Initialization of Filter Weights for Rectifiers

$$\Delta y_l = f'(y_l) \Delta x_{l+1}$$

We assume that  $f'(y_l)$  and  $\Delta x_{l+1}$  are independent of each other

$$E[\Delta y_l] = E[\Delta x_{l+1}]/2 = 0,$$

$$E[(\Delta y_l)^2] = \text{Var}[\Delta y_l] = \frac{1}{2} \text{Var}[\Delta x_{l+1}].$$

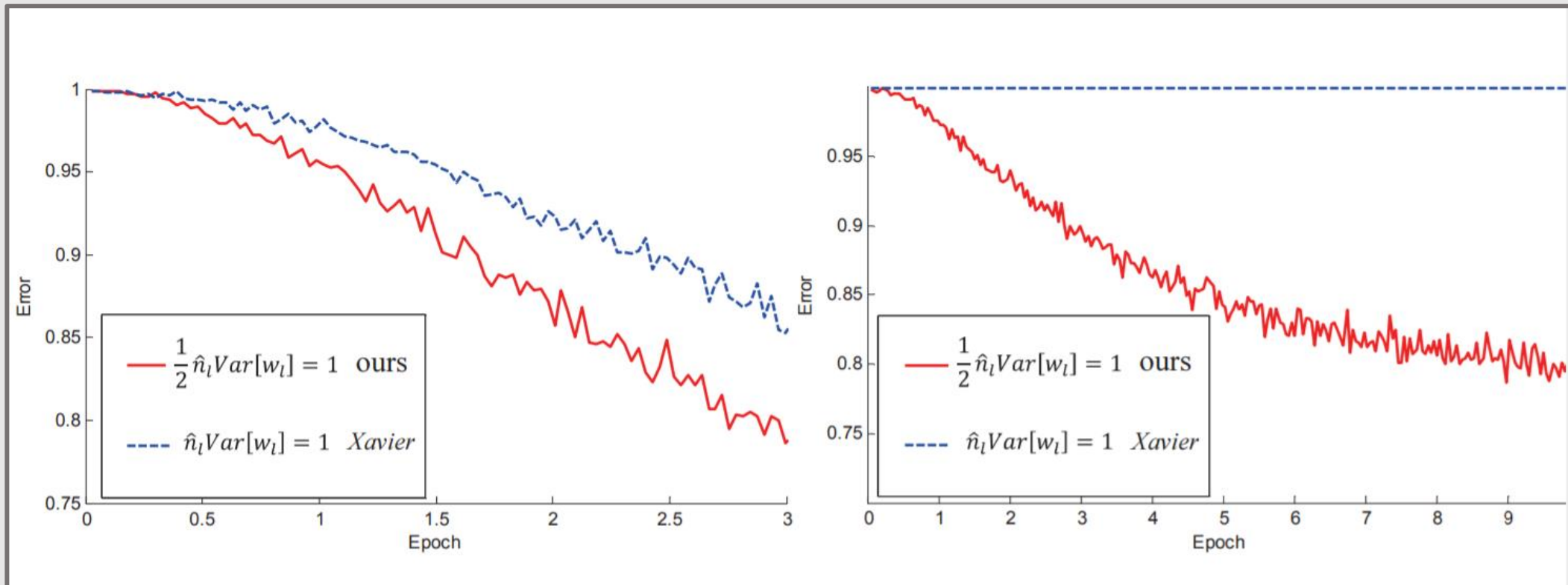
$$\begin{aligned} \text{Var}[\Delta x_l] &= \hat{n}_l \text{Var}[w_l] \text{Var}[\Delta y_l] \\ &= \frac{1}{2} \hat{n}_l \text{Var}[w_l] \text{Var}[\Delta x_{l+1}]. \end{aligned}$$

$$\frac{1}{2} \hat{n}_l \text{Var}[w_l] = 1, \quad \forall l.$$

Sufficient condition

$$\text{Var}[\Delta x_2] = \text{Var}[\Delta x_{L+1}] \left( \prod_{l=2}^L \frac{1}{2} \hat{n}_l \text{Var}[w_l] \right)$$

# Initialization of Filter Weights for Rectifiers



## Convergence of a 22-layer / 30-layer (both ReLU)

**He initialization** is able to make the extremely deep model converge.

On the contrary, the **Xavier method** completely stalls the learning, and the gradients are diminishing as monitored in the experiments.

# Initialization of Filter Weights for Rectifiers

We found that this degradation is because of the increase of training error when the model is deeper. Such a degradation is still an open problem

Though our attempts of extremely deep models have **not** shown benefits on accuracy, our initialization paves a foundation for further study on increasing depth. We hope this will be helpful in understanding deep networks.



# Experiments on ImageNet

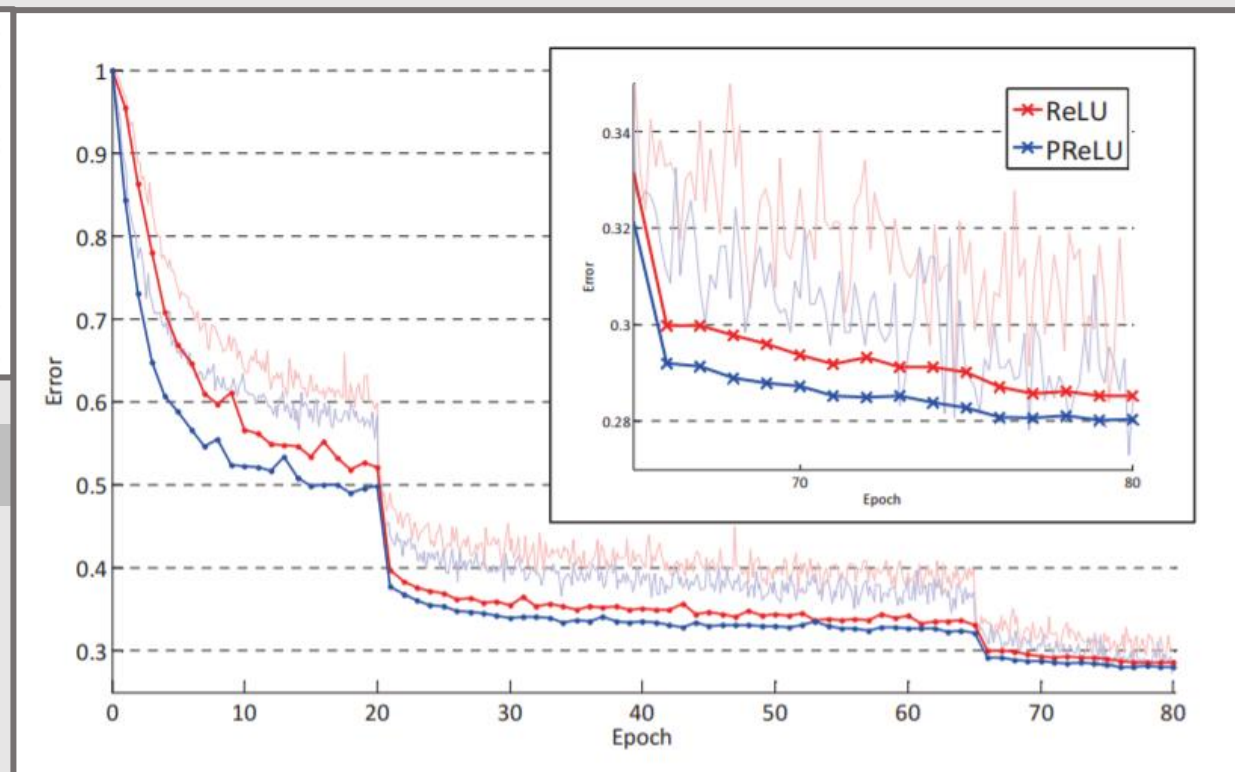
We perform the experiments on the 1000-class **ImageNet 2012 dataset**

- 1.2 million training images
- > 50,000 validation images, and 100,000 test images
- top1/top-5 error rates

# Experiments on ImageNet

model A	ReLU		PReLU	
	top-1	top-5	top-1	top-5
256	26.25	8.25	<b>25.81</b>	<b>8.08</b>
384	24.77	7.26	<b>24.20</b>	<b>7.03</b>
480	25.46	7.63	<b>24.83</b>	<b>7.39</b>
multi-scale	24.02	6.51	<b>22.97</b>	<b>6.28</b>

## Comparisons between ReLU and PReLU



PReLU converges faster than ReLU.

Moreover, PReLU has lower train error and val error than ReLU throughout the training procedure.

# Experiments on ImageNet

	method	top-1	top-5
in ILSVRC 14	SPP [12]	27.86	9.08 <sup>†</sup>
	VGG [29]	-	8.43 <sup>†</sup>
	GoogLeNet [33]	-	7.89
post ILSVRC 14	VGG [29] (arXiv v2)	24.8	7.5
	VGG [29] (arXiv v5)	24.4	7.1
	ours (A, ReLU)	24.02	6.51
	ours (A, PReLU)	22.97	6.28
	ours (B, PReLU)	22.85	6.27
	ours (C, PReLU)	<b>21.59</b>	<b>5.71</b>

Single-model results evaluated from test set

	method	top-5 (test)
in ILSVRC 14	SPP [12]	8.06
	VGG [29]	7.32
	GoogLeNet [33]	6.66
post ILSVRC 14	VGG [29] (arXiv v5)	6.8
	<b>ours</b>	<b>4.94</b>

Multi-model results for ImageNet test set

Combine six models

# Experiments on ImageNet



GT: horse cart

1: horse cart

- 2: minibus
- 3: oxcart
- 4: stretcher
- 5: half track



GT: birdhouse

1: birdhouse

- 2: sliding door
- 3: window screen
- 4: mailbox
- 5: pot



GT: forklift

1: forklift

- 2: garbage truck
- 3: tow truck
- 4: trailer truck
- 5: go-kart



GT: coucal

1: coucal

- 2: indigo bunting
- 3: lorikeet
- 4: walking stick
- 5: custard apple



GT: komondor

1: komondor

- 2: patio
- 3: llama
- 4: mobile home
- 5: Old English sheepdog



GT: yellow lady's slipper

1: yellow lady's slipper

- 2: slug
- 3: hen-of-the-woods
- 4: stinkhorn
- 5: coral fungus

Human performance yields a **5.1%** top-5 error on the ImageNet dataset.

-> Our result (**4.94%**) exceeds the reported human-level performance

Comparisons with Human Performance