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

Imalsha Dinuwanthi E/15/081 - e15081@eng.pdn.ac.lk
Sewwandie Nanayakkara E/15/238 - e15238@eng.pdn.ac.lk
Vidwa Sripadi E/15/345 - e15345@eng.pdn.ac.lk
Hasini Thilakarathna E/15/362 - e15362@eng.pdn.ac.lk

Background

- **Authors :** Kaiming He,

  Xiangyu Zhang,

  Shaoqing Ren,

  Jian Sun
- **Published in :** IEEE International Conference on Computer Vision (ICCV)
- **Date of conference :** 4 - 13 December 2015
- **Conference location :** Santiago, Chile

Contribution

- Parametric Rectifiers
- Initialization of Filter Weights for Rectifiers
- Experiments on ImageNet

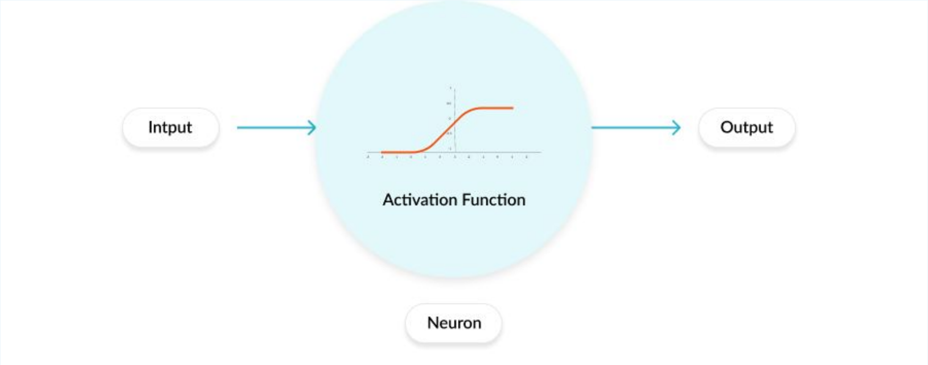
Overview

Two main stages:

1. Propose a Parametric Rectifier Linear Unit (PReLU)
2. Derive a robust initialization method that considers the rectifier nonlinearities

Activation Functions

- Determine the output of a NN
- Decides whether a neuron should be activated or not.
- Examples: sigmoid function,tanh function, ReLU



Activation Functions

Hidden layer i.e. layer 1

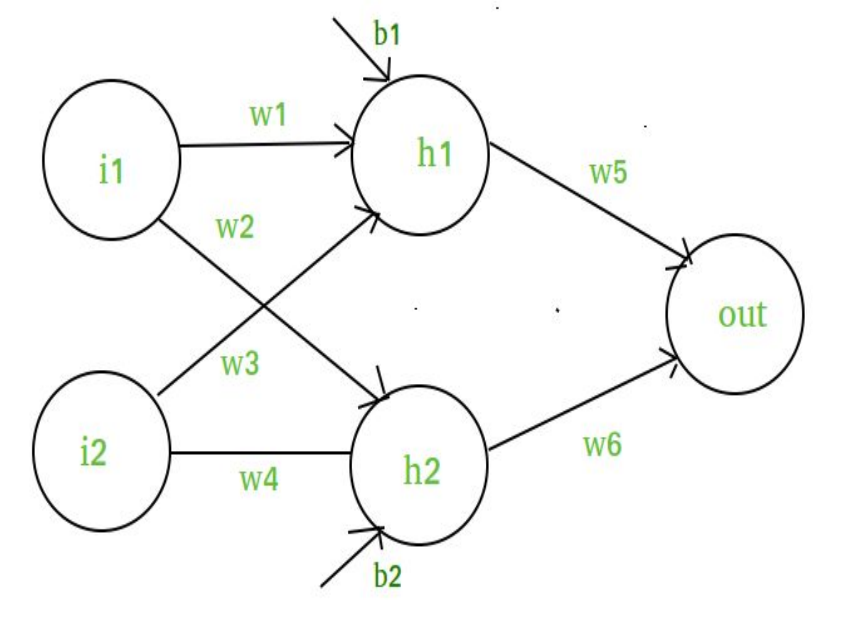
z(1) = W(1)X + b(1)

a(1) = z(1)

Layer 2 i.e. output layer

z(2) = W(2)a(1) + b(2)

a(2) = z(2)



Calculation at Output layer:

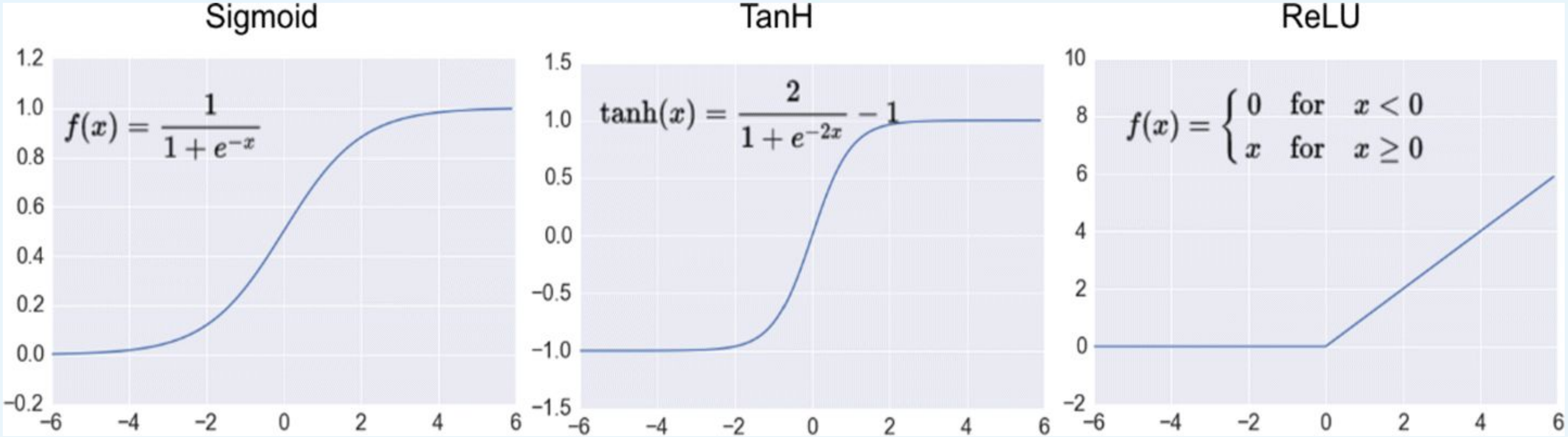
z(2) = (W(2) * [W(1)X + b(1)]) + b(2)

z(2) = [W(2) * W(1)] * X + [W(2)*b(1) + b(2)]

[W(2) * W(1)] = W   & [W(2)*b(1) + b(2)] = b

Final output : z(2) = W*X + b

Activation Functions



Definition

Generic form of Rectifier Linear Function

$$f(x_i) = \begin{cases} x_i, & \text{if } x_i > 0 \\ a_i x_i, & \text{if } x_i \leq 0 \end{cases}$$

ReLU: when $a_i = 0$

$f(x_i) = max(0, x_i)$

PReLU: when a_i is a learnable parameter

$f(x_i) = max(0, x_i) + a_i min(0, x_i)$

LReLU: Leaky ReLU, when $a_i = 0$

$f(x_i) = max(0, x_i) + 0.01 min(0, x_i)$

ReLU Vs PReLU

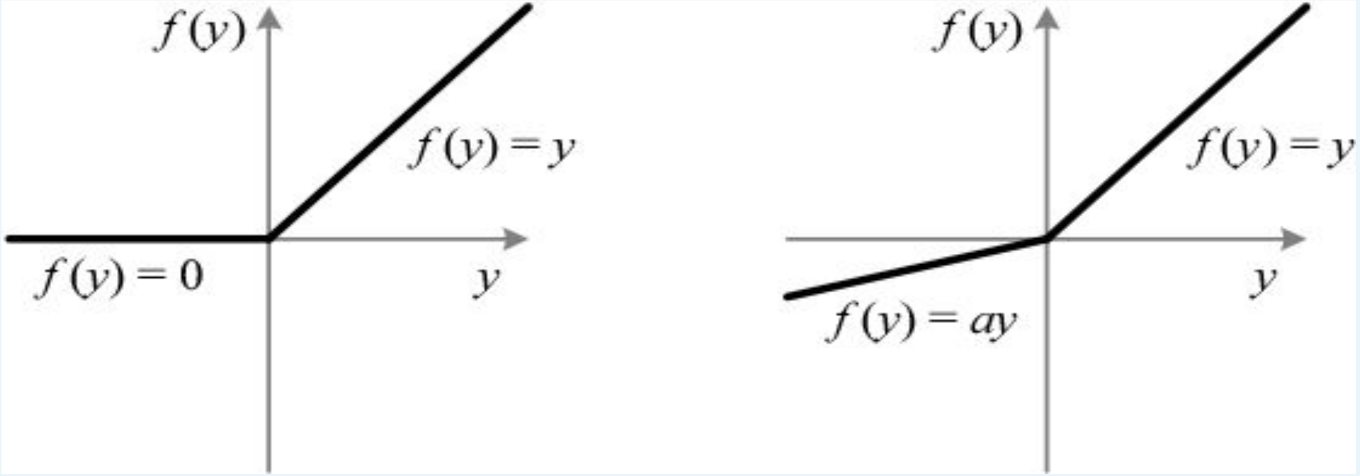


Figure 1. ReLU vs. PReLU. For PReLU, the coefficient of the negative part is not constant and is adaptively learned.

Parametric Rectifier Linear Unit (PReLU)

- As a replacement for Rectifier Linear Unit (ReLU)
- Significance:
  - **Enables to train extremely deep rectified models directly from scratch**
  - **Negligible additional computational cost and overfitting risk**
  - **A 26% relative improvement over the ILSVRC 2014 winner GoogLeNet**
  - **The first report to surpass the reported human-level performance on this particular dataset.**

What is Optimization?

- The term optimize is “to make perfect”
- It is the process of choosing the best inputs which gives the best possible output
- An act, process, or methodology of making something (such as a design, system, or decision) as fully perfect, functional, or effective as possible
  - **Ex: Minimal cost, maximal profit, minimal error**

What is Optimization in NNs?

- Non-convex optimization
- Optimizers are algorithms for changing attributes of NNs.
  - **Examples: Gradient Descent, Stochastic Gradient Descent(SGD),SGD with momentum**

Optimization

- Trained using backpropagation
- Optimized simultaneously with other layers

The gradient of a_i for one layer:

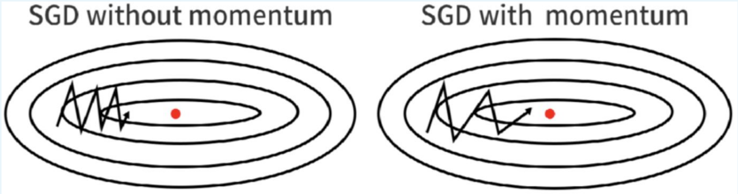
$$\frac{\partial \varepsilon}{\partial a_i} = \sum_{y_i} \frac{\partial \varepsilon}{\partial f(y_i)} \frac{\partial f(y_i)}{\partial a_i}$$

$\partial \varepsilon$: Objective function

$\frac{\partial \varepsilon}{\partial f(y_i)}$: Gradient propagated from the deeper layer

$\frac{\partial f(y_i)}{\partial a_i}$: Gradient of the activation $= \begin{cases} 0, & if \;\; y_i > 0 \\ y_i, & if \;\; y_i \leq 0 \end{cases}$

Momentum



Adopt The Momentum Method When Updating a_i

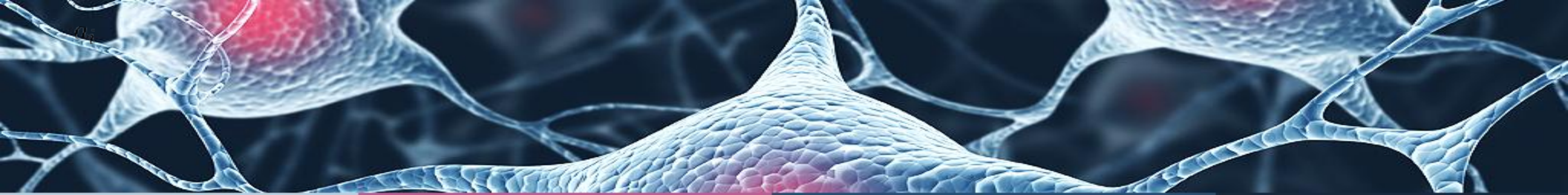
$$\Delta a_i := \mu \Delta a_i + \epsilon \frac{\partial \varepsilon}{\partial a_i}$$

μ : Momentum

ϵ : Learning Rate

Initial a_i : 0.25

- No weight decay!
- Momentum method is used to get a better / faster convergence



- For the channel-shared variant, the gradient of a_i ,

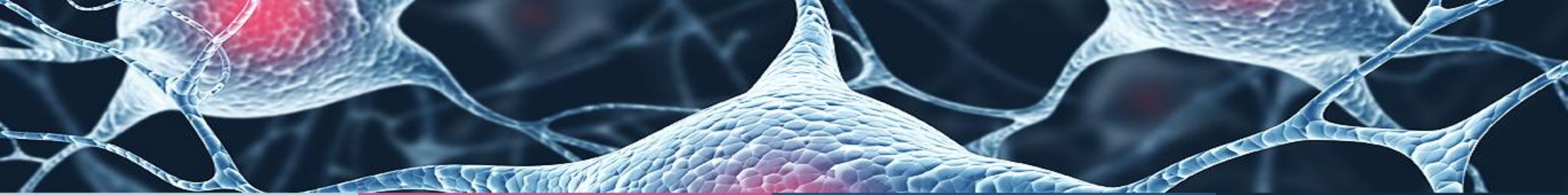
$$\frac{\partial \varepsilon}{\partial a} = \sum_i \sum_{y_i} \frac{\partial \varepsilon}{\partial f(y_i)} \frac{\partial f(y_i)}{\partial a}$$

$\sum_i$: sums over all channels of the layer

Comparison Experiments

layer		learned coefficients	
		channel-shared	channel-wise
conv1	7×7, 64, /2	0.681	0.596
pool1	3×3, /3		
$conv2_1$	2×2, 128	0.103	0.321
$conv2_2$	2×2, 128	0.099	0.204
$conv2_3$	2×2, 128	0.228	0.294
$conv2_4$	2×2, 128	0.561	0.464
pool2	2×2, /2		
$conv3_1$	2×2, 256	0.126	0.196
$conv3_2$	2×2, 256	0.089	0.152
$conv3_3$	2×2, 256	0.124	0.145
$conv3_4$	2×2, 256	0.062	0.124
$conv3_5$	2×2, 256	0.008	0.134
$conv3_6$	2×2, 256	0.210	0.198
spp	{6, 3, 2, 1}		
fc_1	4096	0.063	0.074
fc_2	4096	0.031	0.075

- Comparisons on a deep but efficient model with 14 weight layers



- Comparisons between ReLU, LReLU, and PReLU on the small model for ImageNet 2012
  - **10- view testing**
  - **Each view is 224×224**
  - **All models are trained using 75 epochs**

	top-1	top-5
ReLU	33.82	13.34
LReLU ($a = 0.25$)	33.80	13.56
PReLU, channel-shared	32.71	12.87
PReLU, channel-wise	**32.64**	**12.75**

Initialization Of Filter Weights For Rectifiers

- A robust initialization method
- Removes an obstacle of training extremely deep rectifier networks
- Allows for extremely deep models
- *Xavier* initialization
  - **Based on the assumption that the activations are linear**
  - **This assumption is invalid for ReLU and PReLU**

Forward & Backward Propagation Case

Forward Propagation

Investigate the variance of the responses in each layer

For a conv layer, a response is:

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

Backward Propagation

The gradient of a conv layer is computed by:

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

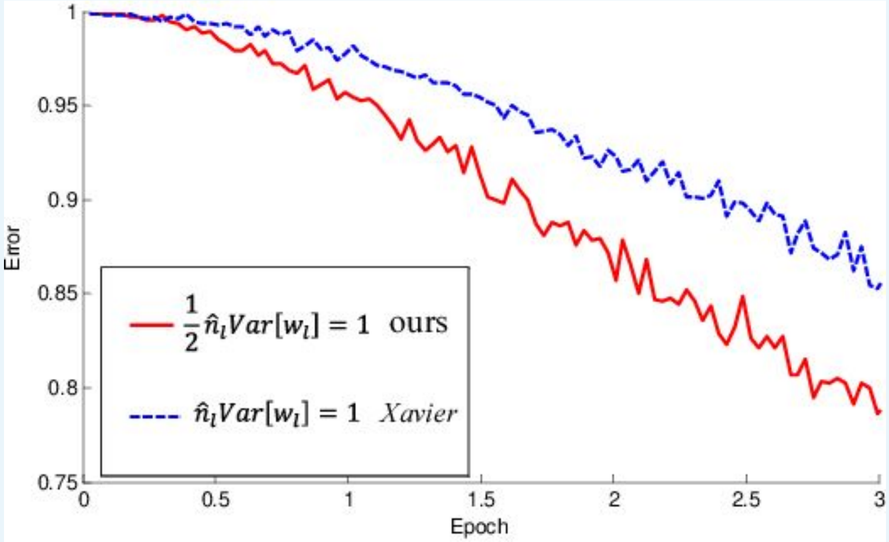
Variance of the product of independent variables

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

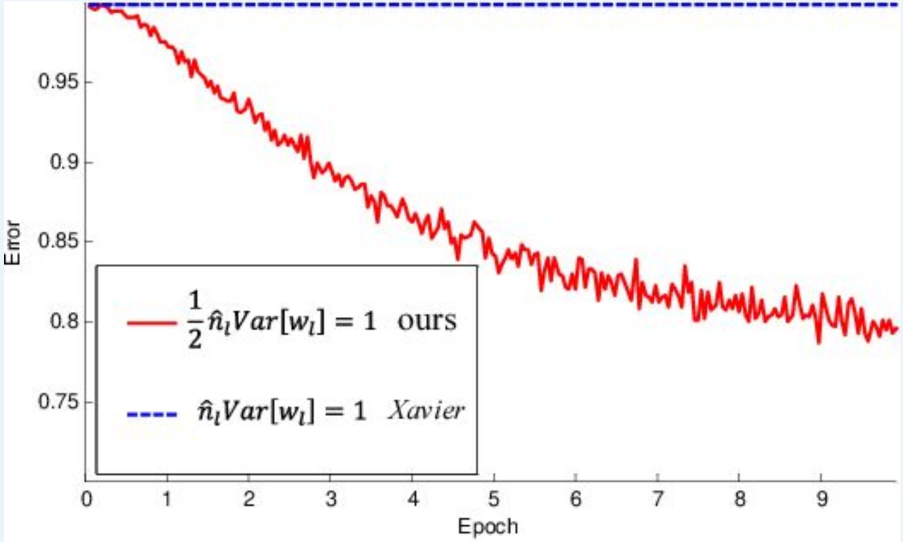
$$Var[\Delta x_2] = Var[\Delta x_{L+1}] \left(\prod_{l=2}^{L} \frac{1}{2} \hat{n}_l Var[w_l] \right)$$

Comparison with *Xavier* initialization

Convergence of a 22-layer model



Convergence of a 30-layer model



Architectures and Implementation

input size	VGG-19 [29]	model A	model B	model C
224	3×3, 64 3×3, 64 2×2 pool, /2	7×7, 96, /2	7×7, 96, /2	7×7, 96, /2
112	3×3, 128 3×3, 128 2×2 pool, /2	2×2 pool, /2	2×2 pool, /2	2×2 pool, /2
56	3×3, 256 3×3, 256 3×3, 256 3×3, 256 2×2 pool, /2	3×3, 256 3×3, 256 3×3, 256 3×3, 256 3×3, 256 2×2 pool, /2	3×3, 256 3×3, 256 3×3, 256 3×3, 256 3×3, 256 3×3, 256 2×2 pool, /2	3×3, 384 3×3, 384 3×3, 384 3×3, 384 3×3, 384 3×3, 384 2×2 pool, /2
28	3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 pool, /2	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 pool, /2	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 pool, /2	3×3, 768 3×3, 768 3×3, 768 3×3, 768 3×3, 768 3×3, 768 2×2 pool, /2
14	3×3, 512 3×3, 512 3×3, 512 3×3, 512 2×2 pool, /2	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 spp	3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 3×3, 512 spp	3×3, 896 3×3, 896 3×3, 896 3×3, 896 3×3, 896 3×3, 896 spp
fc_1, fc_2, fc_3	4096, 4096, 1000			
depth	19	19	22	22
comp.	1.96	1.90	2.32	5.30

Architectures of large models

- 19-layer model (A)
- Model A - faster running speed
- Model B is a deeper version of A (3 extra layers)
- Model C is a wider (with more filters) version of B

ImageNet

- A database with hundreds and thousands of images
- Images are organized in a hierarchy
- Useful for computer vision applications

Accuracy with Top-n Approach

When you have a lot of different classes, the standard accuracy metric might be misleading. Top N accuracies might help in overcoming this issue.

Top-1 accuracy is the conventional accuracy, which means that the model answer (the one with the highest probability) must be exactly the expected answer.

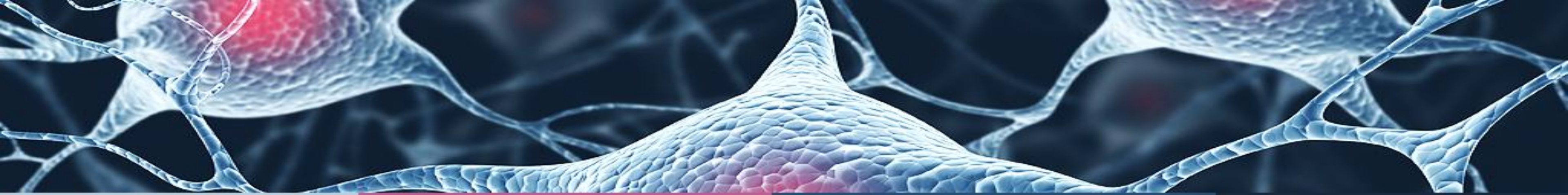
Top-5 accuracy means that *any* of your model that gives 5 highest probability answers that must match the expected answer.

Top-5 error is the percentage of the time that the classifier did not include the correct class among its top 5 guesses

Experiments on ImageNet

- Comparisons between ReLU and PReLU

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



- Comparisons of Single-model Results

	method	top-1	top-5
in ILSVRC 14	SPP [12]	27.86	9.08†
	VGG [29]	-	8.43†
	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)	**21.59**	**5.71**

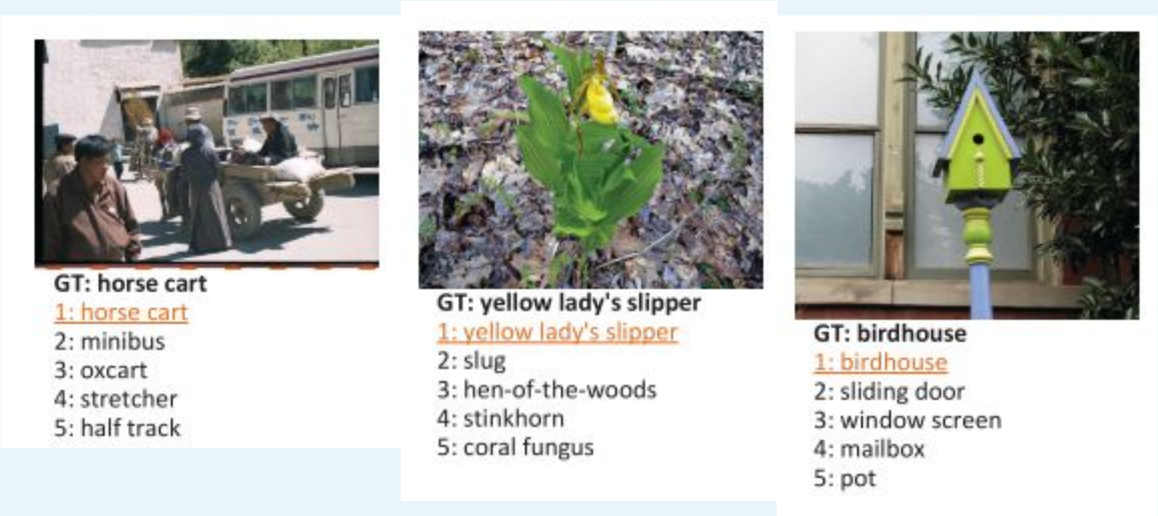
- Comparisons of Multi-model Results

	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
	ours	**4.94**

Comparison with Human Performance

- Russakovsky et al. - human performance yields a 5.1% top-5 error on the ImageNet dataset.

- Our result - 4.94% exceeds the human level performance.



Examples for successfully classified images

Summary

Investigated neural networks driven by the rectifiers:

- First, proposed **PReLU** which adaptively learns the parameters.
- Second, derived a theoretically sound **initialization method**.

On ImageNet 2012 dataset achieves **4.94%** top-5 error

- 26% relative improvement over 2014 Winner GoogLeNet(6.66%)
- **Surpassess for the first time human-level performance** (5.1%)

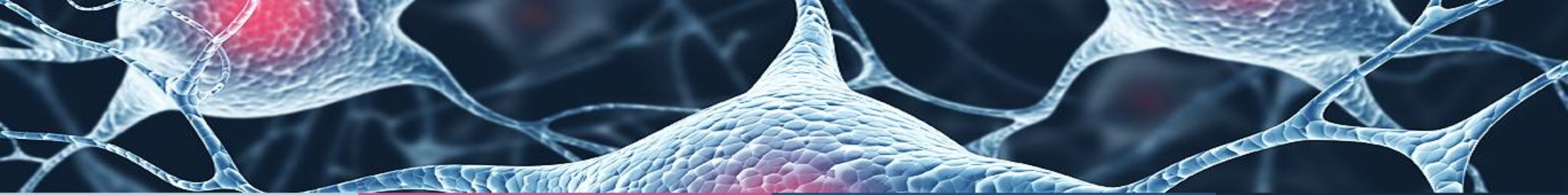
Reference

[1] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. IEEE International Conference on Computer Vision (ICCV 2015). 1502. 10.1109/ICCV.2015.123.

[2] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In International Conference on Artificial Intelligence and Statistics, 2010.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed,D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich.Going deeper with convolutions.arXiv:1409.4842, 2014.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh,S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein,et al. Imagenet large scale visual recognition challenge.arXiv:1409.0575, 2014.



Thank You!!!