University of Peradeniya
Department of Computer Engineering

CO542 Neural Networks and Fuzzy Systems

Neural networks Reading Group 2021

---

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

Group 01:
E/16/049 Harshana Bandara
E/16/087 Sachini Dissanayaka
E/16/276 Buddhi Perera
E/16/388 Thushara Weerasundara
September 24,2021

Contents

List of Figures

List of Tables

Paper title	Delving Dive into Rectifiers:Surpassing Human-Level Performance on ImageNet Classification
Paper year	2015
Paper authors	Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

Table 1: Paper information

1 Introduction

Given in Table 1 are some information about this paper [1]. In this paper, authors discuss their study on rectifier neural networks for image classification under two areas. First, they propose a modified version of the rectified linear unit named Parametric Rectified Linear Unit (PReLU) and then they propose a new initialization scheme for deep neural networks with non-linear activation functions. The authors state that PReLU improves model fitting with nearly zero extra computational cost and little over fitting risk and with the suggested initialization method, they were able to train deep neural networks in a stable manner. They have used the ImageNet 2012 classification dataset and they were able to achieve 4.94% top-5 test error on this dataset. Authors were able to win the 2015 Imagenet competition with the improvements proposed in this paper.

Answered by: Buddhi Perera (E/16/276)

2 Questions and answers

2.1 How many parameters will be introduced in PReLU?

This will introduce single additional parameter per layer. If we have n layers, we have to calculate additional n parameters in the network. The paper suggests the time complexity due to PReLU is negligible for both forward and backward propagation.

Answered by: Sachini Dissanayake (E/16/087)

2.2 What is the value of a_i in LReLU? Is it fixed?

For Leaky ReLU, the slope is a small, fixed value. Usually it is taken as 0.1, but it can be changed. We can specify this ai as a hyper parameter since in LReLU, it does not change when training the network.

Answered by: Buddhi Perera (E/16/276)

2.3 Compare the improvement of PReLU over ReLU for the model used in the paper?

The improvement of PReLU over ReLU has been observed on the 14 layer model selected by the authors of this paper. They have considered a convolutional neural network approach. This is a model that has been studied in [1]. This model can be trained faster and more accurately than other Neural Network models. Also this model does not require multiple high end GPUs to work with as well. The model was trained 1st with ReLU and then PReLU was applied using $a_i = 0.25$ as the initialization. The results of both procedures were observed. They considered top-1 and top-5 errors as evaluation metrics. Top-N error means, for top N highest probability classes, the expected vs predicted answers were checked and the accuracies were considered to each class individually. This is a better metric than considering overall accuracy because it might give a wrong sense of better performance.

Additionally they have considered the performance of LReLU as well. For PReLU top-1 error is reduced to 32.64% from 33.82% with this approach. This is a 1.2% gain over the ReLU as displayed in Fig. 1.

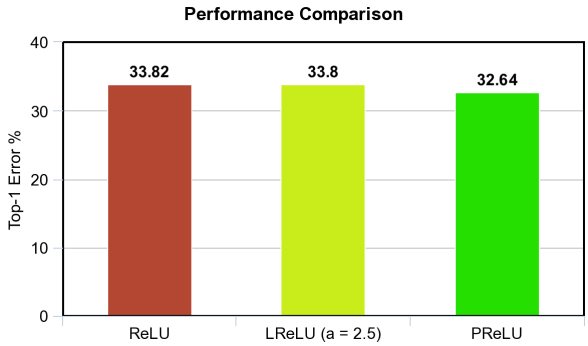


Figure 1: Top - 1 Error of 3 Activation Functions

When considering top-5 error, it was reduced to 12.75% from 13.34% in PReLU over ReLU as displayed in Fig. 2.

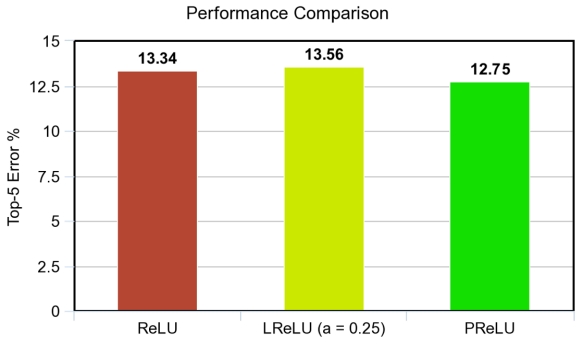


Figure 2: Top - 5 Error of 3 Activation Functions

		learned coefficients	
layer		channel-shared	channel-wise
convl	7x7, 64, *12*	0.681	0.596
pooll	3x3, /3		
$conv2_1$	2x2, 128	0.103	0.321
$conv2_2$	2x2, 128	0.099	0.204
$conv2_3$	2x2,128	0.228	0.294
$conv2_4$	2x2, 128	0.561	0.464
pool2	2x2, /2		
$conv3_1$	2x2, 256	0.126	0.196
$conv3_2$	2x2, 256	0.089	0.152
$conv3_3$	2x2, 256	0.124	0.145
$conv3_4$	2x2, 256	0.062	0.124
$conv_5$	2x2, 256	0.008	0.134
$conv3_6$	2x2, 256	0.210	0.198
spp	{6,3,2,1}		
fc_1	4096	0.063	0.074
fc_2	4096	0.031	0.075

Table 2: Channel-wise vs Channel-shared approaches for PReLU

Additionally, performance of PReLU was compared between channel-wise vs channel-shared PReLU approaches.In channel-shared approach, PReLU introduces 13 extra free parameters with respect to the ReLU approach.

Each layer introduce a parameter and it is a learnable parameter. 1.1% performance gain is a result of this parameters. We can observe the learned coefficients in Table 2.

This implies the importance of adaptive learning the shapes of activation functions and the improvement over ReLU in PReLU. Since for negative inputs, ReLU is consistently 0, therefore it has a big negative bias. We can avoid this using PReLU. Since the parameters in each layer are adapting in PReLU and enable back propagation for negative values, we can observe improved results for metrics considered than in ReLU.

Answered by: Weerasundara W.M.T.M.P.B. (E/16/388)

2.4 What is the difference between LReLU and PReLU?

Leaky ReLU is a modification added to ReLU which replaces the zero part of the domain in [-∞,0] by a small gradient, as we can see in the equation 3. The gradient a is fixed for LReLU. This avoids 0 gradient rate in ReLU therefore, the neurons that are not activated in ReLU because they are in negative domain are activated in LReLU.
In PReLU a_i is a learnable parameter and i represents layer. If is a small fixed value, then PReLU is reduced to RELU. Gradient parameters are iteratevely updated using the chain rule as the weights in the neural network with back propagation. Momentum method was used for update gradients. Since the a_i is updating with each iteration, errors can be reduced and we can expect improved performance for ImageNet 2012 classification data set. Activation function for PReLU is equation 4.

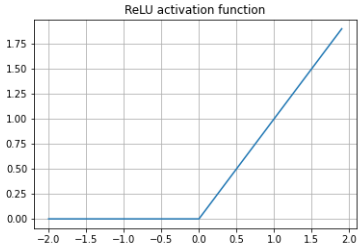
$$\nabla a_i = \mu \nabla a_i + \epsilon \frac{\partial \varepsilon}{\partial a_i} \tag{1}$$

$\mu = momentum, \epsilon = learningrate$

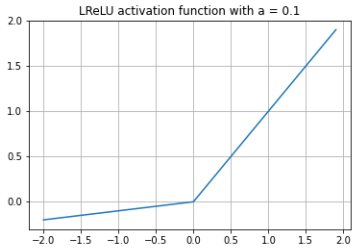
Answered by: Weerasundara W.M.T.M.P.B. (E/16/388)

2.5 What is the dying ReLU problem in neural networks?

The ReLU activation function (2), is a widely used activation function in neural networks. Its high performance over other activation functions like



(a) ReLU activation function



(b) LReLU activation function with $a = 0.1$

Figure 3: ReLU vs. Leaky ReLU

sigmoid, tanh is based on its computational efficiency. Because in ReLU, if the input is negative, the output is zero, Fig. 3a. Although this property of ReLU gives the computational advantage over other linear functions, this can cause the neurons to always give an output of zero regardless of input values [2]. Once a neural become inactive, it loses the capability to train the parameters and mostly cannot recover from that state. If this "dying" continued, a significant part of the network can become inactive. This "dying" happens mainly because of large negative bias, or higher learning rate. To give the neuron a chance to recover Leaky ReLU (3), and Parametric ReLU (4) functions have derived from ReLU function. These functions have non-zero outputs for negative input values, Fig. 3b.

$$y = max\{x, 0\} \tag{2}$$

$$y = \begin{cases} x; & x \geq 0 \\ 0.1x; & otherwise \end{cases} \tag{3}$$

$$y = \begin{cases} x; & x \geq 0 \\ ax; & otherwise \end{cases} \tag{4}$$

Answered by: Buddhi Perera (E/16/276)

2.6 Momentum method in gradient descent

Neural networks are optimized by minimizing the cost function. To do this, different methods are used. A popular and fundamental method is gradient

descent. In gradient descent, The cost function is mapped against the parameters of the neural networks. Cost function is the penalty formula which gives a penalty according to the bias of the neural network. To optimize the neural network, we need to find which combination of parameters reduce the magnitude of cost function.
Now that we know what is the cost function, we need to find a minimum cost function value and its parameters. We use gradient descent to find the minimum value of cost function. Since the cost function has many parameters, we have an n-dimensional surface as the cost function (assuming there are n number of parameters in neural network). We can resemblance the gradient descent method with a blind man cautiously trying to find a valley [3]. The man starts at somewhere on the surface and take one step to the downward slope direction. This step-size represents the learning rate in the gradient descent. The man will proceeds to step in the downward direction until he finds a valley. Sometimes man stops at a local valley which is in a higher level than the global valley. This can happen in gradient descent. When the learning rate is too low, this can converge to a local minimum. If we increase the learning rate, this might take a long time to converge or even can diverge.
To overcome this problem, the concept of momentum has introduced to the system. Now we can resemblance the new gradient descent with a heavy ball rolling downhill [3]. The ball will gain a momentum and can overcome local minimum points and reach the global minimum point. In mathematical terms, momentum method adds This momentum method also converges faster than traditional gradient descent and can be used with a higher learning rate than traditional gradient descent.
Answered by: Bandara S. D. M. V. G. H. N. (E/16/049)

2.7 Does the paper give a constraint to a_i?, does a_i has a specific range ?

The paper [4] suggests that a_i should not tend to zero since it can biased towards ReLU. To avoid this from happening, they did not use weight-decay (l2 regularization) while learning a parameters. Also the paper [4] states that the magnitude of a is usually in $(0, 1)$ range.
Answered by: Bandara S. D. M. V. G. H. N. (E/16/049)

2.8 Basic idea of $l2$ regularization (weight-decay)

l2 regularization used to avoid model over-fitting in neural networks. in neural networks, model over-fitting often happen when weights become larger. To avoid large weights while learning, we use a method called weight-decay. This technique added a fraction of a sum of squared weights to error term. If the weights are high, the error will increase rapidly. This makes the weights to become smaller. However, in this paper [4] they have avoided using l2 regularization while learning a_i. Because it tends to push a to 0. If a becomes smaller, there won't be a much different between PReLU and ReLU.
Answered by: Bandara S. D. M. V. G. H. N. (E/16/049)

References

[1] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[2] Lu Lu. Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28(5):1671–1706, Jun 2020.

[3] Gabriel Goh. Why momentum really works. *Distill*, 2(4):e6, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.