JOURNAL OF SCIENCE
OF LAC HONG UNIVERSITY
ISSN: 2525 - 2186
Tạp chí Khoa học Lạc Hồng, **2026**, 26, 056-062

HÀM KÍCH HOẠT TUYẾN TÍNH THÍCH NGHI CHO KIẾN TRÚC MẠNG HỌC SÂU

Đỗ Trung Dũng *
Trường Đại học Công thương TP. HCM, 140 Lê Trọng Tấn, Phường Tây Thạnh, TP. Hồ Chí Minh, Việt Nam
* Tác giả liên hệ: *dungdt@huit.edu.vn*

THÔNG TIN BÀI BÁO

Ngày nhận: 09/02/2026
Ngày hoàn thiện: 26/03/2026
Ngày chấp nhận: 27/03/2026
Ngày đăng: 15/04/2026

TỪ KHÓA

Thị giác máy;
Phân loại ảnh;
Hàm kích hoạt;
Hàm kích hoạt ReLU;
Học sâu.

TÓM TẮT

Hàm kích hoạt đóng vai trò quan trọng trong các mạng nơ-ron sâu, giúp nâng cao khả năng khái quát hóa và học các quan hệ phi tuyến phức tạp giữa đầu vào và đầu ra. Trong số các hàm kích hoạt phổ biến, Rectified Linear Unit (ReLU) và các biến thể của nó được sử dụng rộng rãi nhờ khả năng tạo biểu diễn thưa và cải thiện hiệu quả huấn luyện. Tuy nhiên, ReLU vẫn tồn tại hạn chế, đặc biệt là hiện tượng nơ-ron chết khi các trọng số được khởi tạo ngẫu nhiên. Bài báo này đề xuất một phương pháp kích hoạt mới cho mạng nơ-ron sâu. Thay vì đặt các giá trị âm về 0, phương pháp đề xuất phân tích phân bố các giá trị dương và âm trong mỗi lớp trước khi áp dụng hàm kích hoạt, từ đó làm suy giảm các giá trị có tần suất xuất hiện cao hơn. Các thí nghiệm trên các tập dữ liệu công khai như MNIST, CIFAR-10 và CIFAR-100 cho thấy phương pháp này giúp quá trình huấn luyện hội tụ ổn định hơn so với ReLU và có tiềm năng được sử dụng như một hàm kích hoạt mặc định.

ADAPTIVE RECTIFIED LINEAR UNIT FOR DEEP NEURAL NETWORK ARCHITECTURE

Do Trung Dung*
Ho Chi Minh City University of Industry and Trade, 140 Le Trong Tan, Tay Thanh Ward, Ho Chi Minh City, Vietnam
*Corresponding Author: *dungdt@huit.edu.vn*

ARTICLE INFO

Received: Feb 9th, 2026
Revised: Mar 26th, 2026
Accepted: Mar 27th, 2026
Published: Apr 15th, 2026

KEYWORDS

Computer Vision;
Image Classification;
Activation Function;
Rectified Linear Unit;
Deep Learning.

ABSTRACT

Activation function is a crucial factor in many deep neural network structures as it helps to boost the generalization for learning more complex mapping functions between inputs and outputs. Many activation functions can be used to build a neural network model including Rectified Linear Unit and its extended forms. By setting negative values to zeros, ReLU can introduce sparsity representation and mimic nonlinearity during training. Even though ReLU works well in creating almost Deep Neural Network structures, it still faces the problem of dying neurons especially when weights are randomly initialized. This paper introduces a novel method to activate neurons in a Deep Neural Network layer. Instead of assigning the negative values to zero, this paper automatically calculates the distribution of positive and negative values in each layer before performing the activation function. According to the distribution, the values with a higher distribution are then suppressed. Experimental results are performed with a simple Deep Neural Network structure on some public datasets such as MNIST, CIFAR10, and CIFAR100 showing that the proposed method obtains much more stable convergence results compared to the ReLU. Thus, it can be used as the default activation function when training a Deep Neural Network model.

Doi: https://doi.org/10.61591/jslhu.26.1075
Available online at: https://lhj.vn

1. INTRODUCTION

Computer vision is a very active area and always a frontier of finding new models, and novel applications to be applied to other fields such as medical treatment [1-2] and rescue [3]. The problems in computer vision can be solved by using classical machine learning algorithms including Support Vector Machine (SVM) [4], Decision Tree [5], Random Forest [6], AdaBoost [7-8], or Neural Networks [9].

Building models using Neural Networks, two common activation functions including Sigmoid [10] and Hyperbolic Tangent were used in the early days. The Sigmoid function is traditionally a popular activation function in the 90th decade of the last century. The inputs to the Sigmoid function are transformed into values between 0.0 and 1.0. Inputs that are much larger than 1.0 are transformed to the value of 1.0 and the values that are much smaller than 0.0 are mapped to the value of 0.0. The shape of the function for all possible inputs is an S-shape from zero up through 0.5 to 1.0. The Sigmoid function was the default activation function used on Neural Networks for a long time. The Hyperbolic Tangent function or Tanh for short, is a similarly shaped nonlinear activation function that outputs values in the range of -1.0 and 1.0. In the later years of the 90th decade, the Tanh function was preferred over the Sigmoid function since it was easier to train and often had better predictive performance. At that moment, researchers all knew that a Neural Network with only a single hidden layer is very good at handwriting digit number classification.

A model with much larger parameters can approximate a more complex mapping function from inputs to outputs. To build a Neural Network model with a huge number of parameters, many layers can be stacked over each other. Unfortunately, the Neural Network model with more than 2 hidden layers makes it difficult to learn a new thing since its gradient cannot propagate back from the deeper layer to the earlier one. Nowadays we all know it as the vanishing gradient caused by using Sigmoid or Tanh activation function. With that Neural Network limitation, researchers from all over the world switched their focus to classical machine learning algorithms to solve the problems. Even though Neural Networks and back-propagation algorithms were very famous and founded much earlier than other classical machine learning algorithms, it was rarely utilized since their effectiveness could not be seen with the network having more than two hidden layers.

Many workarounds such as different weight initialization schemes [31], layer-wise training [32] [35], unsupervised pre-training [33-34], and alternative activation functions were conducted to overcome the problem of vanishing gradient when training the Deep Neural Network. Among them, Rectified Linear Unit (ReLU) [36] is the most effective activation function for training a neural network model. In 2012, a very famous work done by Alex Krizhevsky [11] put the neural network back on the right track and got a lot of researchers' attention thanks to its excellent classification results on the ImageNet photo classification dataset [12]. The network structure was very similar to the LeNet [13] which was used since the 80th decade of the last century. The novelty of Alex Krizhevsky's work was threefold: a large classification dataset, a powerful computational device, and a new activation function. Instead of using a Sigmoid activation function, Alex's paper utilized a novel activation function, named Rectified Linear Unit (ReLU) which is known as an important activation function for preventing the vanishing gradient in the backpropagation algorithm. Moreover, the ReLU retains the network properties, easily takes the derivatives, and becomes the default activation function for building multiple-layer neural networks. Using the ReLU activation function, one can add many hidden layers to the neural network without facing the vanishing gradient problem. Since then, neural networks have attracted much attention again and Deep Learning (DL) has been found accordingly.

Along with the Deep Learning (DL) concept, a lot of new models have been discovered to advance the network structure in terms of accuracy and processing time such as VGG [14], GoogleNet [15], ResNet [16], MobileNet [17] [18], DenseNet [19-20], EfficientNet [21], NasNet [22], and ConvNeXt [23] etc. Most of these network structures obtain state-of-the-art results using the ReLU activation function. On the other hand, with the help of new DL models, not only the difficult tasks in classification can be solved but also the challenging tasks in object detection [24-26] and semantic segmentation [27-28] fields can obtain better results as well.

The role of the activation function is to boost the generalization between the inputs and outputs of the network structure while retaining the simple calculations. Although the ReLU activation function has strengths compared to the Sigmoid and Tangent activation functions in terms of its ability to prevent the appearance of vanishing gradient, using ReLU also has certain disadvantages: the network model is difficult to converge due to the problem of dying neurons [39] and it needs to be specially initialized. Some methods including extended forms of ReLU [37-38] and different strategies for the weight initialization [29-30] are proposed to overcome the problem.

Although the usage of ReLU mitigates the vanishing gradient problem, it faces another issue named dying neuron which occurs when neurons output negative values. Those neurons are in the inactive mode that has no contribution to the learning process. To overcome the dying neuron problem, some advanced versions of ReLU have been introduced such as Leaky ReLU [37], PReLU [38], GeLU [40], and ConvReLU [41]. The Leaky ReLU [37] introduces a slope for the negative values of neurons. Instead of setting negative values to zeros, it assigns small values to the negative neuron thus the dying neuron can be avoided. However, selecting an effective coefficient of slope is another issue and needs to be carefully experimented with. On the other hand, the PreLU [38] is a combination version of ReLU and Leaky ReLU that makes coefficient slope a learnable parameter. The GeLU [40] is a relatively recent addition to the family of activation functions. It stands out due to its probabilistic foundations

and smooth approximation characteristics and gains notable popularity in transformer [42] architectures, such as those used in models like BERT and GPT. Another version of the activation function is ConvReLU [41] which introduces a hyperparameter θ. The ConvReLU is always the identical function except for the neuron whose input values are smaller than θ which are set to be θ. This method needs many parameters compared to the ReLU and cannot utilize the sparsity representation as in the ReLU.

This paper proposes a novel activation function named the Adaptive Rectified Linear Unit. Instead of setting all negative values at each layer to zero (as in ReLU), the proposed method first calculates the distribution of negative and positive numbers in a layer. According to the distribution, the Adaptive Rectified Linear Unit is applied to suppress the higher distribution. Using the proposed activation function, the positive values can be retained in one layer while preserving the negative ones in another. That means both positive and negative values are used for training a model hence avoiding the dying neuron problem. The contribution of this paper is twofold: i) propose a new activation that can stabilize the loss function during training of a deep neural network; ii) Run and compare the proposed activation function with ReLU on some public datasets such as MNIST, CIFAR-10, and CIFAR-100 to show the effectiveness of the proposed activation function.

The remainder of the paper is organized as follows: Section 2 introduces the proposed Adaptive Rectified Linear Unit. Section 3 discusses and shows the experimental results on some public datasets and intermediate results during training and testing. Finally, Section 4 provides the conclusions and future works.

2. ADAPTIVE RECTIFIED LINEAR UNIT FOR DEEP NEURAL NETWORK

Activation function is a crucial factor in building any deep neural network. Choosing the right activation function helps the model learn more complex mapping functions, converge faster during the training process, and thus obtain a higher performance in the inference. Currently, the most widespread activation function, used as the default in many deep neural network, is the Rectified Linear Unit thanks to its simplicity and effectiveness during the training process.

2.1 Rectified linear unit

The valuable properties of ReLU activation function over Sigmoid and Hyperbolic Tangent are the ability to avoid vanishing gradient and to have sparsity neurons during training. Generally, the activation function of a layer j on a neuron i is denoted as $f^j(x_i^j)$. The output of the layer j on a neuron i is given by,

$$h_i^j = f^j(x_i^j) \tag{1}$$

The ReLU activation function is mathematically given by,

$$h_i^j = \max(x_i^j, 0) = \begin{cases} x_i^j & if\ x_i^j > 0 \\ 0 & else \end{cases} \tag{2}$$

The ReLU activation function outputs the same values as the inputs if the input values x_i^j are positive, and outputs zeros if the values are negative. Using ReLU, all neurons with negative values are set to zeros introducing the desirable property, sparsity. That means, during the training of the model, not all neurons are used to learn the mapping function between inputs and outputs thus avoiding the overfitting problem.

Since only a single activation *max(...)* is applied over all hidden layers, f^j keeps the same for all hidden layers, h^j with $j = 1..n$, where n is the number of hidden layers in the deep neural network. When a neuron only receives the negative value during training, the output from that neuron through the ReLU activation function is zero all the time. It is known that the problem of dying neurons occurs when neurons are never activated during training. Figure 1 shows the graphical illustration of the Rectified Linear Unit.

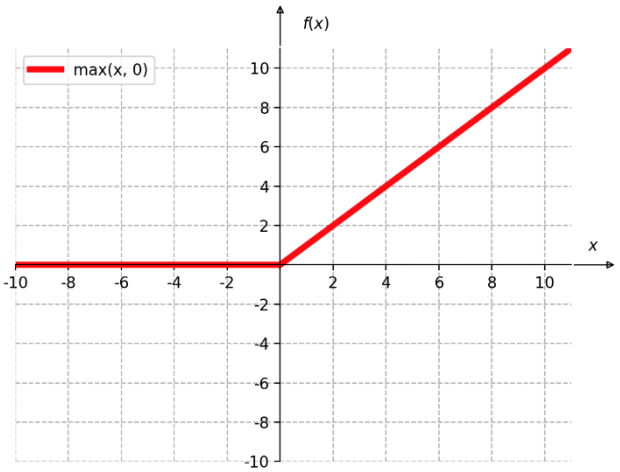

Figure 1. *Graphical illustration of Rectified Linear Unit, ReLU.*

2.2 Adaptive rectified linear unit

Instead of using a single activation function, ReLU, that activates positive values in all layers of a deep neural network model, this paper proposes an Adaptive Rectified Linear Unit (AdaReLU) to activate not only positive values but also negative values in layers during training. The AdaReLU owns several interesting properties including sparsity like the original ReLU, no more hyperparameters are needed as the extended versions Leak ReLU, PReLU, and ConvReLU. The basic equation of the adaptive activation function for the output of a layer is described in equation (3). Unlike the ReLU which uses the same activation function *max(...)* over all layers, the AdaReLU automatically generates various activations for neurons in each layer depending on the distribution of positive and negative values. The equation of AdaReLU is defined as,

$$h_i^j = \begin{cases} \min(x_i^j, 0) & if\ |H^j|_{pos} > |H^j|_{neg} \\ \max(x_i^j, 0) & else \end{cases} \tag{3}$$

Where h_i^j denotes the pre-activation output of the i-th neuron in the layer j. Let $H^j = \{h_1^j, h_2^j, \dots, h_N^j\}$ denote the set of all neuron outputs in layer j for a given mini-batch, where N is the total number of activations in that layer. $|H^j|_{pos}$ and $|H^j|_{neg}$ denote the number of positive and negative values in layer j, respectively.

The *min(...)* activation function is applied to neurons if, in a layer *j*, the number of positive values is greater than the number of negative values. Otherwise, the *max(...)* activation function is being utilized. The reason for setting zeros on all neurons in a layer with a bigger distribution is to maximize the sparsity for the overfitting problem. Unlike the ReLU activation function which retains only positive values, the AdaReLU function can retain positive or negative values depending on the distribution of layer values thus avoiding the problem of dying neurons as using the original activation function ReLU. Figure 2 shows the graphical illustration of the Adaptive Rectified Linear Unit.

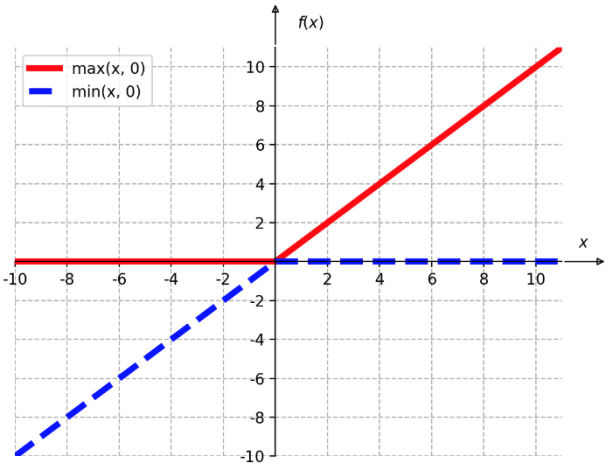

Figure 2. *Graphical illustration of Adaptive Rectified Linear Unit, AdaReLU.*

In deep neural networks, the distribution of neuron activations within a layer often becomes highly imbalanced, particularly when using rectified activation functions. When one side dominates (e.g., $|H^j|_{poss} \gg |H^j|_{neg}$), the learning dynamics tend to be driven primarily by the majority activation pattern. This imbalance can reduce the diversity of learned feature representations and may lead to biased gradients during optimization, as the majority activation region contributes disproportionately to the gradient updates. Consequently, the network may overemphasize specific activation patterns, which can negatively affect generalization performance.

AdaReLU addresses this issue by adaptively suppressing the activation distribution with higher frequency. By reducing the magnitude of the dominant activation group, the method encourages a more balanced activation distribution within each layer. This mechanism can be interpreted as a form of implicit regularization, as it prevents the dominance of a single activation pattern and promotes more diverse feature representations. As a result, the learned representations tend to be more robust, which contributes to improved generalization performance.

3. EXPERIMENTAL RESULTS

3.1 General setups

In this experiment, a PC with two GeForce GTX 3080 graphics cards and 64 GB of RAM running Ubuntu 24.04 LTS with the 2.10 version of Pytorch is used. For a fair comparison with other activation functions including ReLU, Leaky ReLU, and PReLU, the same and relatively small deep neural network structure is applied with the identical weight initialization, and no data augmentation is performed.

The goal of this paper is to show the effectiveness of the proposed activation function AdaReLU, no new network structure is introduced. Therefore, the VGG-16-like [14] and ResNet-18-like [16] architectures are utilized for comparison. The mini-batch size is set to 32 at a fixed momentum of 0.9. To train a model, the total number of epochs is set to 100 and the initial value of the learning rate is set to 0.001, which keeps reducing 10% after every 20 epochs. For the stabilization results, 20 models are initialized, trained, and tested. The accuracies presented in this paper are the average accuracies taken from 20 models.

The MNIST is the dataset of handwritten digits. It has a training set of 60,000 examples and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image. The CIFAR-10 dataset consists of 60000 32×32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The CIFAR-100 dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs). The statistical information of those datasets is shown in Table 1.

Table 1. *Statistical information of MNIST, CIFAR-10, and CIFAR-100*

Dataset	# training	# testing	size	# classes
MNIST	60,000	10,000	28×28	10
CIFAR-10	50,000	10,000	32×32	10
CIFAR100	50,000	10,000	32×32	100

3.2 Results on mnist dataset

The proposed method is first evaluated on the MNIST dataset. For a fair comparison with other activation functions, the same setup is applied in the training and testing phases as discussed in the general setups section. Table 2 shows the accuracy of the models using different activation functions. As shown in Table 2, since the recognition of handwritten digits in MNIST dataset is relatively easy, both VGG-16 and ResNet-18 models provide good performances. The activation function ReLU is used as a baseline for other activation functions. The Leaky ReLU performs slightly better than the ReLU from 0.04% to 0.08% while the AdaReLU is better than the ReLU up from 0.14% to 0.33%.

Table 2. *Accuracy comparison between AdaReLU and other activation functions on MNIST dataset*

Network	ReLU	Leaky ReLU	AdaReLU
VGG-16	94.63%	94.71%	94.77%
ResNet-18	94.84%	94.88%	95.17%

3.3 Results on cifar-10 dataset

The proposed method is then evaluated on the CIFAR-10 dataset which is more challenging than the MNIST. The procedures performed with the CIFAR-10 dataset are identical to the MNIST dataset. Table 3 shows the accuracy of the models using different activation functions. As shown in Table 3, the Leaky ReLU performs slightly better than the ReLU from 0.29% to 1.08% while the AdaReLU is better than the ReLU from to 0.82% to 1.28% on 2 network structures VGG-16 and ResNet-18

***Table 3**. Accuracy comparison between AdaReLU and other activation functions on CIFAR-10 dataset*

Network	ReLU	Leaky ReLU	AdaReLU
VGG-16	92.63%	93.71%	93.91%
ResNet-18	94.75%	95.04%	95.57%

3.4 Results on cifar-100 dataset

The proposed method is finally evaluated on the CIFAR-100 dataset which is the most challenging dataset in this paper. The procedures performed with the CIFAR-100 dataset are identical to the other previous datasets. Table 4 shows the accuracy of the models using different activation functions. As shown in Table 4, the Leaky ReLU performs better than the ReLU from 0.66% to 0.87% while the AdaReLU is better than the ReLU from 0.87% to 1.10%.

***Table 4**. Accuracy comparison between AdaReLU and other activation functions on CIFAR-100 dataset*

Network	ReLU	Leaky ReLU	AdaReLU
VGG-16	75.67%	76.54%	76.77%
ResNet-18	76.28%	76.94%	77.15%

3.5 Intermediate results

Figure 3 shows the intermediate results during training and testing of VGG-16 network structure using both ReLU (the classical rectified linear unit) and AdaReLU (the proposed rectified linear unit). Given the same network structure, the proposed rectified linear unit outperforms the commonly used rectified linear unit which is a strong evidence that the AdaReLU is a good choice to train any network for a better performance.

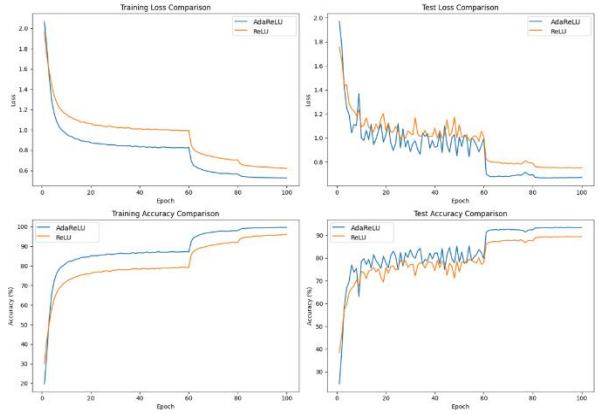

Figure 3. *Loss and accuracy comparisons of Adaptive Rectified Linear Unit, AdaReLU on CIFAR-10 dataset and VGG-16 network structure*

4. CONCLUSIONS AND FUTURE WORK

The activation function plays an important role in almost deep neural network structure. By using the AdaReLU, not only neurons with positive values but also the ones with negative values are activated in the training process. The results of several published small datasets prove the effectiveness of the proposed activation function over the original one. In the future, it will be interesting to train the larger and deeper neural network structures with the AdaReLU as well as test the performance of the trained models on more challenging datasets for different purposes such as classification, detection, and segmentation.

5. REFERENCES

[1] Esteva, A., Chou, K., Yeung, S. et al. ”Deep learning-enabled medical computer vision”. npj Digital Medicine 4, 2021.

[2] Yunzhu Li, Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Sebastian Thrun, ”Skin Cancer Detection and Tracking Using Data Synthesis and Deep Learning”. AAAI Workshops 2017.

[3] Rahnemoonfar, M., Chowdhury, T. Murphy, R., ”RescueNet: A High Resolution UAV Semantic Segmentation Dataset for Natural Disaster Damage Assessment”. Sci Data 10, 913, 2023.

[4] Marti A. Hearst, ”Trends Controversies: Support Vector Machines”. IEEE Intell. Syst. 13(4): p. 18-28, 1998.

[5] Kotsiantis, S.B., ”Decision trees: a recent overview”. Artif Intell Rev. vol. 39, p. 261–283, 2013.

[6] Breiman, L., ”Random Forests”. Machine Learning vol. 45, p. 5–32, 2001.

[7] Robert E. Schapire, ”Advances in Boosting”. UAI p. 446-452, 2002.

[8] Robert E. Schapire, Marie Rochery, Mazin G. Rahim, Narendra K. Gupta, ”Incorporating Prior Knowledge into Boosting”. International Conference on Machine Learning p. 538-545, 2002.

[9] Alberto Prieto, Beatriz Prieto, Eva M. Ortigosa, Eduardo Ros, Francisco J. Pelayo, Julio Ortega, Ignacio Rojas, ”Neural networks: An overview of early research, current frameworks and new challenges”. Neurocomputing 214: 242-268, 2016.

[10] Sridhar Narayan, ”The Generalized Sigmoid Activation Function: Competetive Supervised Learning”. Inf. Sci. 99(1-2): 69-82, 1997.

[11] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ”ImageNet Classification with Deep Convolutional Neural Networks”. NIPS, 1106-1114, 2012.

[12] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, ”ImageNet: A large-scale hierarchical image database,” IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, pp. 248- 255, 2009.

[13] Yann LeCun, L´eon Bottou, Yoshua Bengio, Patrick Haffner, ”Gradient-based learning applied to

document recognition". Proc. IEEE 86(11), 2278-2324, 1998.

[14] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition". ICLR 2015.

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, "Going deeper with convolutions". CVPR, 1-9, 2015.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition". IEEE Conference on Computer Vision and Pattern Recognition, p. 770-778, 2016.

[17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". CoRR abs/1704.04861, 2017.

[18] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks". IEEE Conference on Computer Vision and Pattern Recognition, p. 4510-4520, 2018.

[19] Gao Huang, Shichen Liu, Laurens van der Maaten, Kilian Q. Weinberger, "CondenseNet: An Efficient DenseNet Using Learned Group Convolutions". IEEE Conference on Computer Vision and Pattern Recognition, 2752-2761, 2018.

[20] Le Yang, Haojun Jiang, Ruojin Cai, Yulin Wang, Shiji Song, Gao Huang, Qi Tian, "CondenseNet V2: Sparse Feature Reactivation for Deep Networks". IEEE Conference on Computer Vision and Pattern Recognition, 3569-3578, 2021.

[21] Mingxing Tan, Quoc V. Le, "EfficientNetV2: Smaller Models and Faster Training". ICML, 10096-10106, 2021.

[22] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le, "Learning Transferable Architectures for Scalable Image Recognition". IEEE Conference on Computer Vision and Pattern Recognition, 8697-8710, 2018.

[23] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, Saining Xie, "A ConvNet for the 2020s". IEEE Conference on Computer Vision and Pattern Recognition, 11966-11976, 2022.

[24] Ross B. Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". IEEE Conference on Computer Vision and Pattern Recognition, 580-587, 2013.

[25] Yanghao Li, Hanzi Mao, Ross B. Girshick, Kaiming He, "Exploring Plain Vision Transformer Backbones for Object Detection". ECCV (9), 280-296, 2022.

[26] Juan R. Terven, Diana Margarita Córdova Esparza, Julio-Alejandro Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS". Mach. Learn. Knowl. Extr. 5(4): 1680-1716, 2023.

[27] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, Rohit Girdhar, "Masked-attention Mask Transformer for Universal Image Segmentation". IEEE Conference on Computer Vision and Pattern Recognition, 1280-1289, 2022.

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross B. Girshick, "Mask R-CNN". IEEE Trans. Pattern Anal. Mach. Intell. 42(2): 386-97, 2020.

[29] Leonid Datta, "A Survey on Activation Functions and their relation with Xavier and He Normal Initialization". CoRR abs/2004.06632, 2020.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". ICCV, 1026-1034, 2015.

[31] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks". In Aistats, volume 9, pages 249–256, 2010.

[32] Bernhard Schölkopf, John Platt; Thomas Hofmann, "Greedy Layer-Wise Training of Deep Networks," in Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference, MIT Press, pp.153-160, 2007.

[33] Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, Pierre-Antoine Manzagol, Pascal Vincent, Samy Bengio, "Why Does Unsupervised Pretraining Help Deep Learning?" J. Mach. Learn. Res. 11: 625-660, 2010.

[34] Dumitru Erhan, Aaron C. Courville, Yoshua Bengio, Pascal Vincent, "Why Does Unsupervised Pretraining Help Deep Learning?" AISTATS, p.201-208, 2010.

[35] David Sussillo, "Random Walks: Training Very Deep Nonlinear Feed-Forward Networks with Smart Initialization". CoRR abs/1412.6558, 2014.

[36] Vinod Nair and Geoffrey E Hinton "Rectified linear units improve restricted boltzmann machines". In Proceedings of the 27th International Conference on Machine Learning, pp. 807–814, 2010.

[37] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng, "Rectifier nonlinearities improve neural network acoustic models". In Proc. International Conference on Machine Learning, volume 30, pp. 3, 2013.

[38] 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 Conference on Computer Vision and Pattern Recognition, pp. 1026–1034, 2015.

[39] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga, "Outlier detection with autoencoder ensembles". In Proceedings of the 2017 SIAM

International Conference on Data Mining, pp. 90–98. SIAM, 2017.

[40] Dan Hendrycks, Kevin Gimpel, "Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units". CoRR abs/1606.08415, 2016.

[41] Hongyang Gao, Lei Cai, Shuiwang Ji, "Adaptive Convolutional ReLUs". AAAI 3914-3921, 2020.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "Attention is All you Need". Advances in Neural Information Processing Systems, p. 5998-6008, 2017.