International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Analyzing the Effect of Different Activation Functions in Deep Learning on Accuracy and Execution time

Dr. Mahesh D. Titiya*[1], Arjun V. Bala[2], Dr. Sheshang Degadwala[3]

Submitted:11/03/2024 **Revised**: 26/04/2024 **Accepted**: 04/05/2024

Abstract*:* Activation functions is critical in specifying the active node within neural networks. Choosing the most suitable activation function is crucial because to its impact on the overall output of the network. Prior to choosing an activation function, it is essential to check the characteristics of each activation function based on our specific needs. The monotonicity, derivatives and range of the activation function are important characteristics. In our review study, we examined 13 different activation functions, such as ReLU, Linear, Exponential Linear Unit, Gaussian Error Linear Unit, Sigmoid, SoftPlus, among others.

Keywords: *activation functions, neural network, deep learning, sigmoid, types of ReLU, softsign, tanh*

1. Introduction

Deep learning neural networks are applicable in various fields such as voice detection, speech recognition, pattern identification, and object detection. The initial deep learning model required for detection had a limited number of layers, such as the LeNet5 model which consisted of only five layers. Various types of neural networks have been developed in the past decade, including as ANN, CNN, and RNN. [24].

Due to advancements in computational power and input data, researchers were able to improve the network's depth, resulting in more accurate results. VGGNet has nine-teen or sixteen layers depending on the version. AlexNet has twelve levels. [6], Goog-\\leNet has twenty-two layers, ResNet architecture has 152 layers, and Stochastic Depth networks have approximately 1,200 layers. As a result, studying neural net-works in more depth will provide better findings. The activation function calculates a neural cell's output. The activation function's derivative is used in the backpropagation algorithm. Therefore, the activation function for analysis needs to be selected. This prevents the oscillatory pattern observed in the sigmoid function during back-propagation weight updates. Alom et al.[9] suggest that creating an activation function with minimal computational complexity is crucial for large neural networks including millions of nodes. The activation function is a crucial aspect for mapping parameters in artificial neural networks to handle non-linear complex functions effectively. The main role of the activation function is to specify the state of the node in Artificial Neural Network (ANN).

Multiple hidden layers in a neural network increase the complexity and difficulty of training. Challenges in neural networks can arise from issues such as the vanishing gradient problem, oscillating weight values, complex formulas, or activation function saturation. This leads to a continuous learning process that may be time-consuming [11], [12].

We have primarily concentrated on performing exploratory data analysis on various Activation Functions such as ReLU, Linear, ELU, GeLU, SeLU, Sigmoid, Hard Sigmoid, Tanh, SoftPlus, SoftMax, SoftSign, Swish, and Exponential in this review paper.

2. Dataset

For this analysis we have generated isotropic Gaussian blobs for clustering using sklearn library of the python with the below parameters,

- n_samples, to be 1000
- n_features, to be 2
- centers, to be 4
- random_state, to be 0

Generated data is visualized using Matplotlib library, here is how it looks,

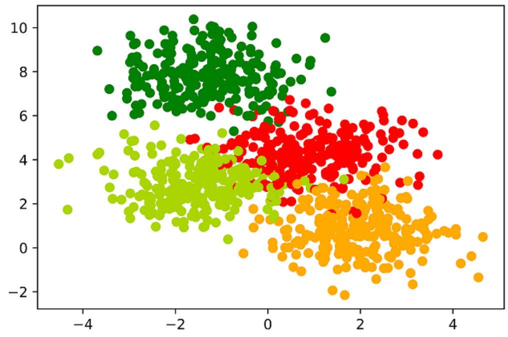

Fig. 1. Generated sample data used for comparison.

3. Activation Functions

3.1. Linear Activation

It's an easy activation function in which the input is precisely proportional to our activation function. Linear activation functions gives a wider range of activations, and a sloped line may enhance the firing rate as the input rate increases.

[1]*Assistant Professor, Computer Engineering Department, Government Engineering College, Rajkot, Gujara,India*
ORCID ID : 0000-0001-6181-6562,Email: mdtitiya@gmail.com
[2] *Research Scholar, Gujarat Technological University, Ahmedabad, Gujarat, India,*
ORCID ID : 0009-0005-9661-0149Email: er.arjunbala@gmail.com
[3] *Professor & Head Department of Computer Engineering, Sigma University, Vadodara, Gujarat, India*
ORCID ID : 0000-0002-2385-7790 Email:sheshang13@gmail.com
** Corresponding Author Email: mdtitiya@gmail.com*

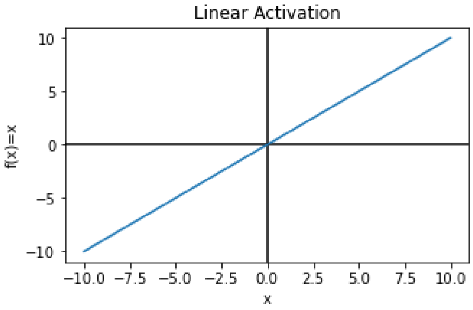

Fig. 2. Linear Activation Function Graph

Function	:	$f(x) = x$
Derivative	:	$f'(x) = 1$
Range	:	-∞ to ∞

Consider the function's derivative, which in our case is 1 because our function is linear with a slope of 1. Because the derivative of the linear activation function is constant, training the model is challenging.

3.2. Rectified Linear Unit (ReLU) Activation

Rectified linear unit (ReLU), is the highly applied activation function today, it has a range of 0 to infinity and replacing all negative values with zero.

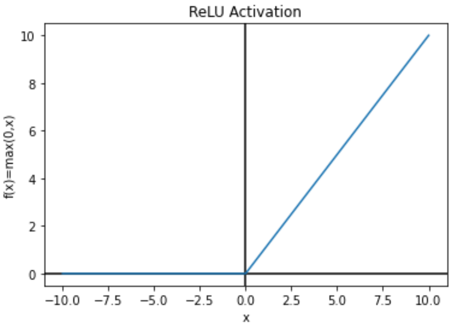

Fig. 3. Rectified Linear Unit (ReLU) Graph

Function	:	$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$
Derivative	:	$f'(x) = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}$
Range	:	0 to ∞

Consider the function's derivative: we have 0 for values less than 0 and 1 for values larger than 0, but the derivative for zero is undefined.

The problem with the ReLU is that the conversion rate is so quick that it can't map or fit into data effectively. To solve this, the Leaky ReLU was created.

3.3. Exponential Linear Unit (ELU) Activation

The Exponential Linear Unit (ELU) has negative values, which lets them bring mean-unit closer to 0 and achieve lower computing complexity, similar to batch normalization [23]. Because of the lower bias shift impact, mean shifts near 0 aid to boost learning speed by bringing the normal gradient closer to the 0 gradient [23].

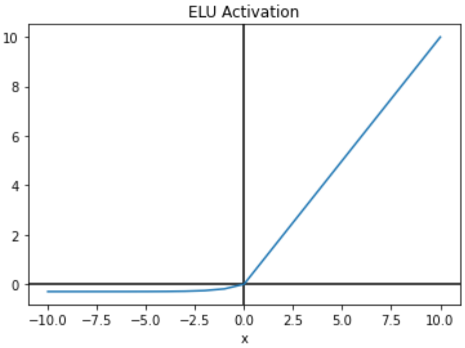

Fig. 4. Exponential Linear Unit (ELU) Graph

Function	:	$f(x) = \begin{cases} \alpha(e^{x} - 1), & x \leq 0 \\ x, & x > 0 \end{cases}$
Derivative	:	$f'(x) = \begin{cases} f(x) + \alpha, & x \leq 0 \\ 1, & x > 0 \end{cases}$

In ELU, we must choose a parameter called alpha, which has a common value in the range of 0.1 to 0.3. In the example above, we have chosen alpha value to be 0.3.

3.4. Gaussian Error Linear Unit (GeLU) Activation

NLP models such as ALBERT, ROBERTa, and BERT frequently use the GELU activation function. This function combines ReLU, dropout, and zoneout features.

GELU can be considered as a smoother ReLU.

This distribution can be useful with Batch Normalization as neuron inputs have a normal distribution.

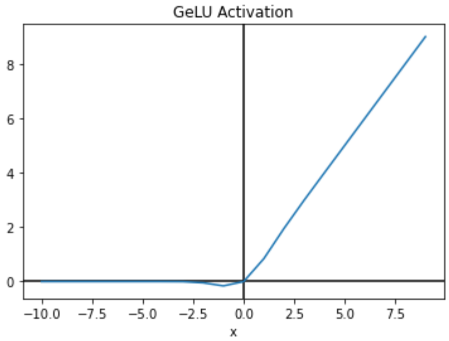

Fig. 5. Gaussian Error Linear Unit (GeLU) Function Graph

Function	:	$f(x) = \frac{1}{2}x(1 + \mathrm{erf}(\frac{x}{\sqrt{2}}))$

3.5. Sigmoid Activation

A sigmoid function is a mathematical function having a "S"-shaped or sigmoid curve. The logistic function is a fundamental example of a sigmoid function; it includes nonlinear features. This is a historical function which is one of the earliest activation function used in neural network.

A sigmoid function is a bounded, differentiable real function defined for all real input values. It has a non-negative derivative at every point and exactly one inflection point. Typically, a sigmoid function is monotonic, with a first derivative that forms a bell-shaped curve. Conversely, integrating any continuous, non-negative, bell-shaped function (having one local maximum and no local minima, unless degenerate) produces a sigmoidal function.

.

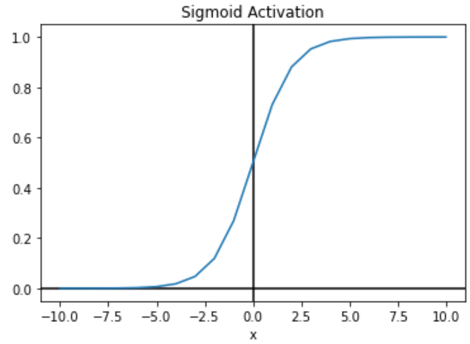

Fig. 6. Sigmoid Activation Function Graph

Function	:	$f(x) = \frac{1}{1+e^{-x}}$
Derivative	:	$f'(x) = f(x)(1 - f(x))$
Range	:	0 to 1

From the provided graph, we can observe that the Y-axis values change far more rapidly than the X-axis values inside a narrow range.

In contrast to other activation functions, which have a range between -∞ and +∞, this function's range is bounded between 0 and 1.

This activation function is commonly used in Deep Learning because of its limited range.

3.6. Hard Sigmoid

The hard sigmoid is an approximation of the logistic sigmoid function that is piecewise linear. Depending on which characteristics of the original sigmoid you wish to preserve, you can employ a unique approximation.

We have kept f(0) to be 0.5 and f'(0) to be -0.25.

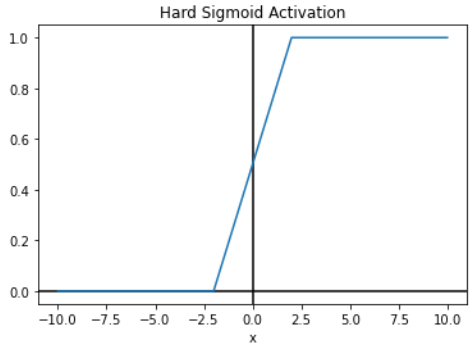

Fig. 7. Hard Sigmoid Activation Function Graph

Function	:	$f(x) = \max(0, \min\left(1, \frac{x+2}{4}\right))$
Range	:	0 to 1

3.7. Hyperbolic Tangent (tanh) Activation

When using simply sigmoid activation functions to learn deep networks, edge values become stuck; to avoid this, we must use hyperbolic functions, often known as the tanh function.

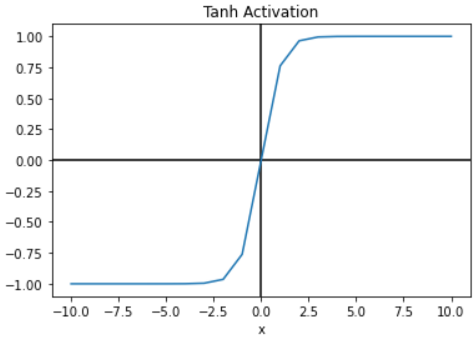

Fig. 8. Hyperbolic Tangent (tanh) Activation Function Graph

Function	:	$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
Derivative	:	$f'(x) = 1 - \tanh^{2}(x)$
Range	:	-1 to 1

3.8. SoftPlus Activation

As a better alternative to hyperbolic or tanhavtivation function we can use SoftPlus activation, which has range from 0 to ∞.

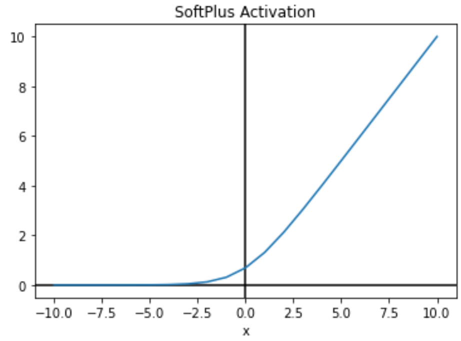

Fig. 9. SoftPlus Activation Function Graph

Function	:	$f(x) = \ln(1 + e^{x})$
Derivative	:	$f'(x) = \frac{e^{x}}{1 + e^{x}}$
Range	:	0 to ∞

3.9. SoftMax Activation

The relative probabilities are computed by the Softmax activation function. Soft-max is a type of logistic function that normalizes an input value into a vector of values that follows a probability distribution whose sum equals 1.

Function	:	$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{z_j}}$
Range	:	0 to 1

The output values are in the range [0,1], this helps us to get rid of binary classification and can have many classes or dimensions within our neural network model. Softmax is also known as multinomial logistic regression for this reason.

3.10. SoftSign Activation

Softsign is an activation function that, like a sigmoid function, rescales values be-tween -1 and 1 by applying a threshold. The benefit is that a Softsign's value is zero-centered, which aids the following neuron in propagation.

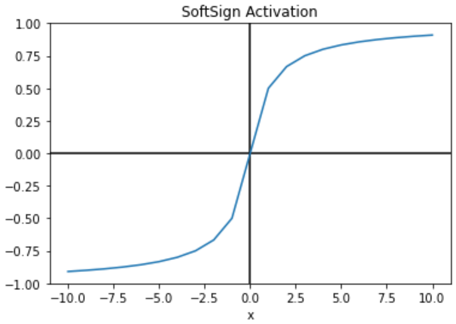

Fig. 10. SoftSign Function Graph

Function : $f(x) = \frac{x}{1 + |x|}$

Range : -1 to 1

4. Neural Network Model

We have created a simple model in which we have used two hidden and one output layer with all combination of the given activation functions.

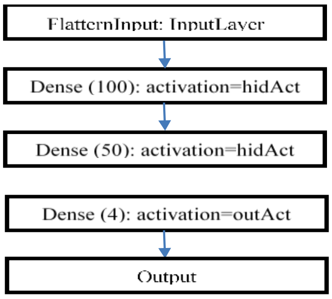
FlatternInput: InputLayer
Dense (100): activation=hidAct
Dense (50): activation=hidAct
Dense (4): activation=outAct
Output

Fig. 11. Neural Network Model used for the experiment.

Here hidAct and outAct is the different combination of below listed activation functions.

- Linear Activation
- Rectified Linear Unit Activation
- SoftPlus Activation
- SoftMax Activation
- SoftSign Activation
- Exponential Linear Unit Activation
- Gaussian Error Linear Unit Activation
- Swish Activation
- Exponential Activation
- Scaled Exponential Linear Unit Activation
- Sigmoid Activation
- Hard Sigmoid Activation
- Hyperbolic Tangent Activation

5. Results

We have done performance analysis using accuracy and execution time parameters.

5.1. Accuracy

Here are the accuracy results we have achieved with the combination of different Activation Functions.

Table 1. Raw Result Data for accuracy

		Hidden Layer											
		linear	**relu**	**elu**	**gelu**	**selu**	**sigmoid**	**Hard sigmoid**	**tanh**	**softplus**	**softmax**	**softsign**	**swish**
Output Layer	**linear**	0.441	0.671	0.559	0.321	0.471	0.269	0.263	0.383	0.399	0.220	0.510	0.451
	relu	0.309	0.432	0.433	0.408	0.320	0.269	0.269	0.350	0.269	0.269	0.408	0.416
	elu	0.504	0.389	0.601	0.268	0.387	0.220	0.220	0.641	0.460	0.220	0.435	0.323
	gelu	0.386	0.218	0.302	0.181	0.217	0.248	0.317	0.308	0.269	0.269	0.330	0.438
	selu	0.314	0.480	0.419	0.302	0.263	0.269	0.372	0.502	0.438	0.269	0.294	0.556
	sigmoid	0.930	0.930	0.933	0.933	0.925	0.928	0.933	0.933	0.925	0.897	0.925	0.928
	hard sigmoid	0.510	0.291	0.439	0.912	0.840	0.348	0.520	0.531	0.269	0.704	0.804	0.342
	tanh	0.410	0.302	0.448	0.577	0.314	0.248	0.593	0.540	0.096	0.220	0.562	0.516
	softplus	0.915	0.934	0.930	0.930	0.928	0.925	0.604	0.921	0.925	0.703	0.927	0.933
	softmax	0.924	0.924	0.931	0.928	0.930	0.927	0.933	0.931	0.928	0.918	0.933	0.925
	softsign	0.356	0.480	0.567	0.555	0.523	0.430	0.263	0.410	0.220	0.220	0.459	0.596
	swish	0.223	0.263	0.523	0.185	0.191	0.263	0.248	0.389	0.018	0.248	0.173	0.408
	exp	0.925	0.931	0.924	0.927	0.921	0.263	0.643	0.934	0.921	0.676	0.931	0.918

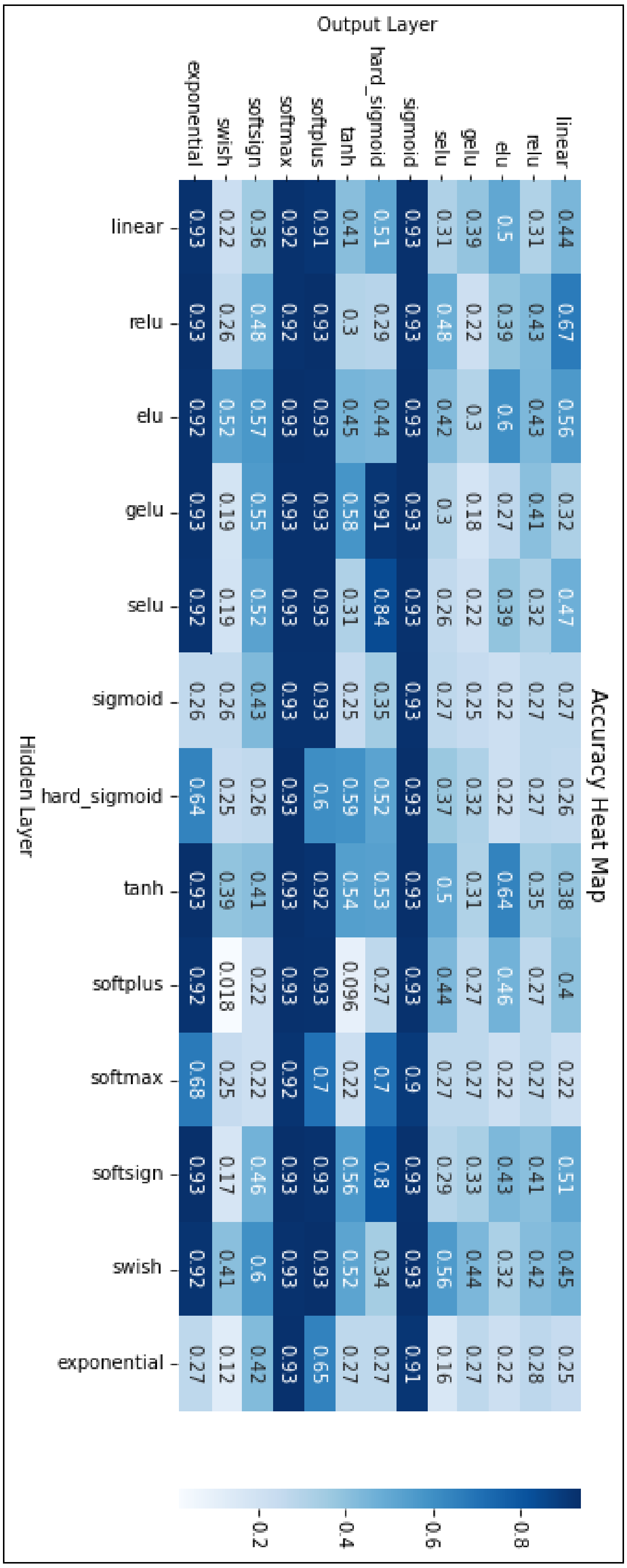

Fig. 12. Accuracy heat map of the result

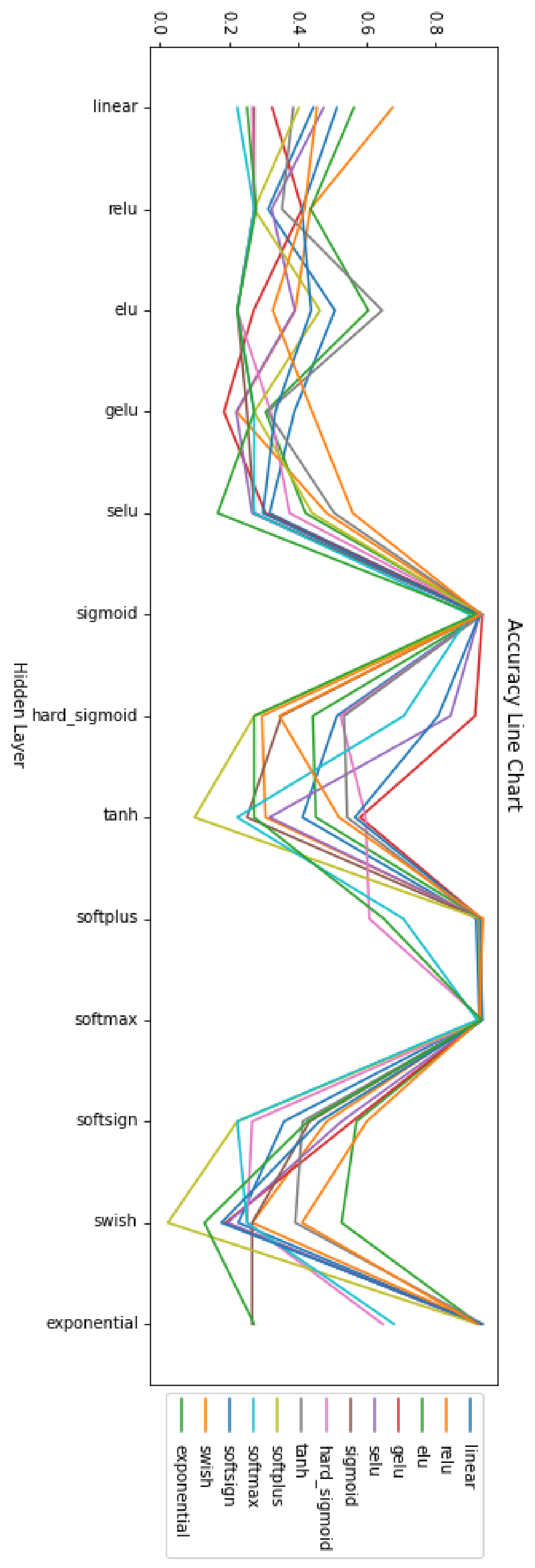

Fig. 13. Accuracy line chart of the result

5.2. Execution Time

Here are the Execution Time (in Seconds) results we have achieved with the combination of different Activation Functions.

Table 2. Raw Result Data for execution time

		Hidden Layer					
		linear	**relu**	**elu**	**gelu**	**selu**	**sigmoid**
Output Layer	**linear**	5.87	5.83	5.85	6.32	5.93	5.86
	relu	5.89	5.90	5.82	6.43	5.96	5.91
	elu	5.85	5.93	5.93	6.31	6.49	5.89
	gelu	6.08	5.97	6.14	6.44	6.49	6.06
	selu	5.82	5.85	5.97	6.36	5.92	5.88
	sigmoid	5.53	5.54	5.66	6.16	5.70	5.74
	hard sigmoid	5.97	5.91	6.02	6.40	6.06	5.98
	tanh	5.84	5.89	6.76	6.36	6.14	5.93
	softplus	5.78	5.78	6.14	6.29	6.03	5.95
	softmax	5.39	5.49	5.72	6.24	5.86	5.78
	softsign	5.93	5.89	5.90	6.38	5.88	5.97
	swish	5.88	5.96	5.96	6.41	5.98	6.00
	exp	5.79	5.88	5.96	6.32	5.93	5.86
		Hard sigmoid	**tanh**	**softplus**	**softmax**	**softsign**	**swish**
Output Layer	**linear**	6.20	5.77	6.10	6.11	5.98	6.07
	relu	6.25	5.91	6.12	6.09	5.93	6.05
	elu	6.26	6.10	6.13	6.22	6.10	6.05
	gelu	6.42	6.23	6.28	6.30	6.36	6.22
	selu	6.18	5.89	6.08	6.18	5.90	6.05
	sigmoid	6.19	5.66	6.26	5.94	5.64	5.76
	hard sigmoid	6.27	6.02	6.55	6.21	6.01	6.26
	tanh	6.25	5.85	6.16	6.19	5.96	6.06
	softplus	6.20	5.98	6.31	6.12	5.97	6.09
	softmax	6.10	5.53	6.15	5.93	5.63	5.90
	softsign	6.24	6.07	6.24	6.10	5.89	6.57
	swish	6.30	6.66	6.26	6.27	5.94	6.24

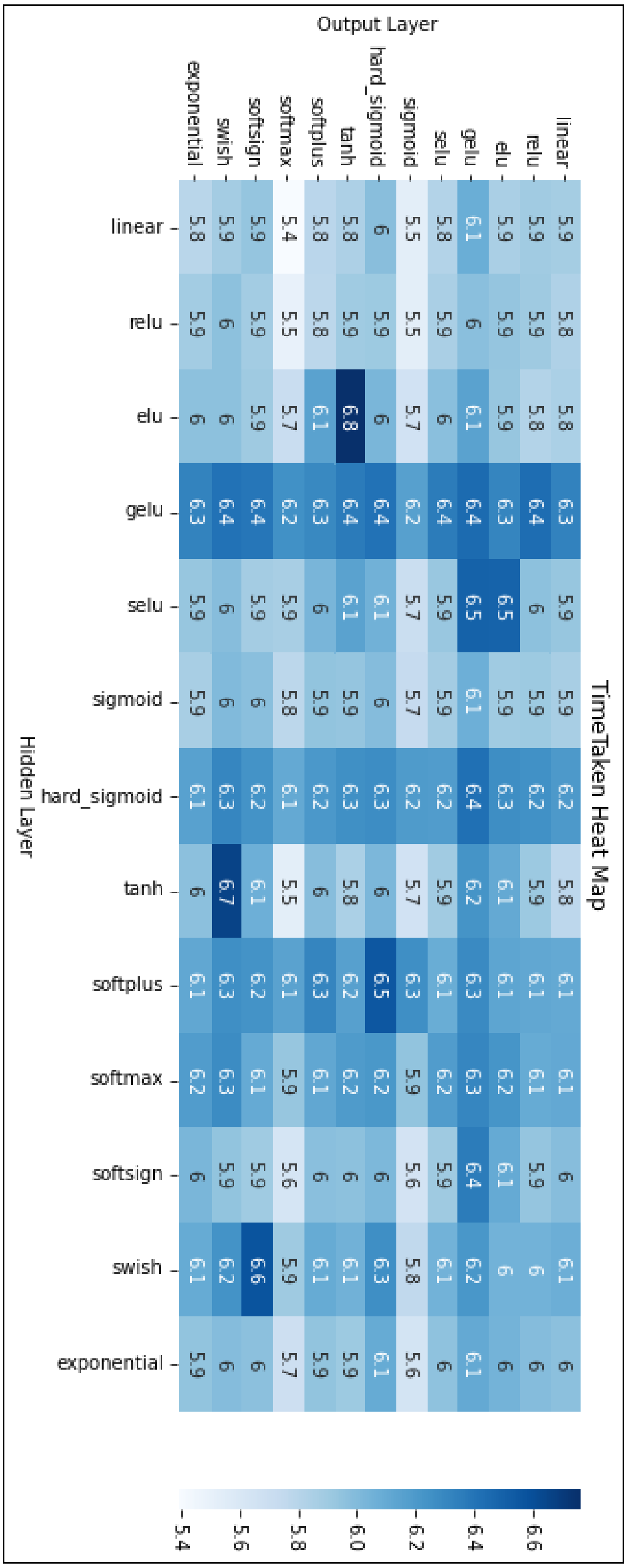

Fig. 14. Execution time heat map of the result

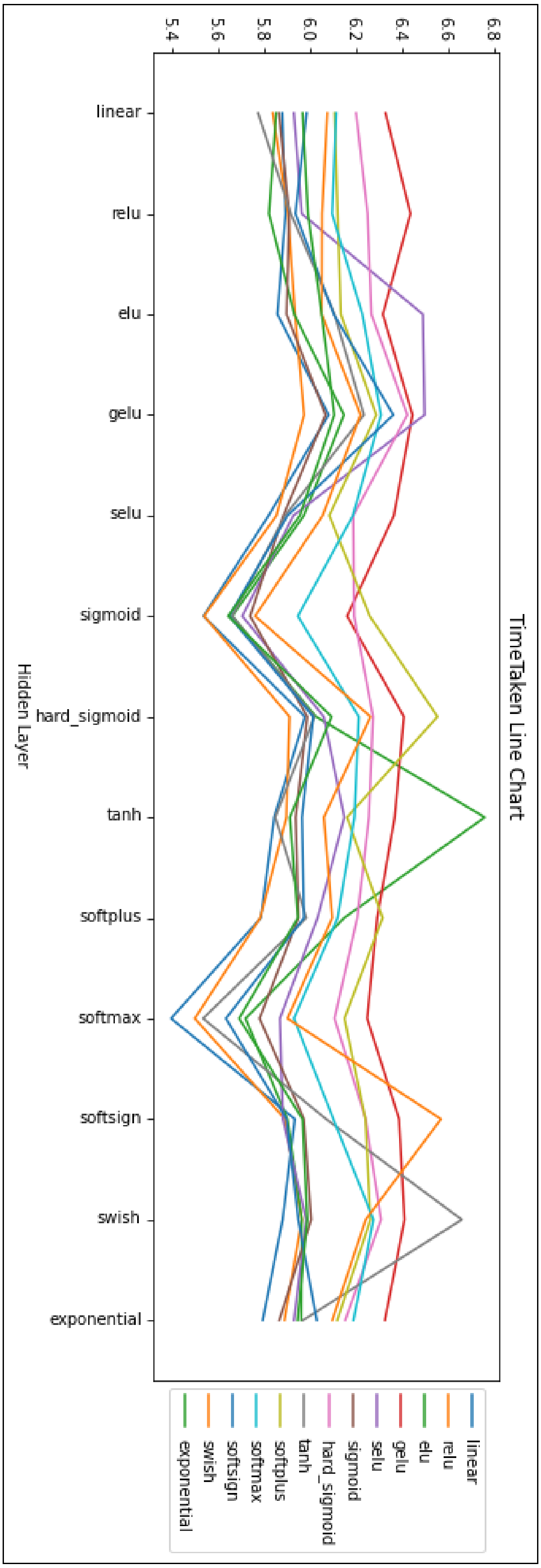

Fig. 15. Execution time line chart of the result

6. Conclusion

In this study, we examined the impact of various activation functions on the performance of deep learning models in terms of both accuracy and execution time. Our experiments included a comprehensive evaluation of popular activation functions such as Sigmoid, Tanh, ReLU, Leaky ReLU, and Swish.

The results indicate that the choice of activation function significantly influences the model's accuracy and computational efficiency. Specifically, ReLU and its variants (Leaky ReLU and Swish) consistently outperformed traditional activation functions like Sigmoid and Tanh in achieving higher accuracy and faster convergence. This can be attributed to their ability to mitigate the vanishing gradient problem, thus facilitating more effective training of deeper networks.

ReLU, due to its simplicity and computational efficiency, emerged as the most effective activation function in most scenarios. However, it was observed that Leaky ReLU and Swish provided marginally better performance in certain cases, particularly in deeper networks where maintaining a non-zero gradient is crucial. The Swish activation function, in particular, demonstrated superior accuracy on complex datasets, suggesting its potential for more advanced applications despite its slightly higher computational cost.

The execution time analysis revealed that activation functions with non-linear characteristics, such as Sigmoid and Tanh, incurred higher computational costs, leading to longer training times. In contrast, ReLU and its variants significantly reduced training time due to their simpler mathematical operations and sparse activation.

In conclusion, while ReLU remains a robust choice for most applications due to its balance of accuracy and efficiency, exploring variants like Leaky ReLU and Swish can provide incremental benefits in specific contexts. Future research could further explore adaptive activation functions and their integration into evolving neural network architectures to enhance model performance. This study underscores the importance of selecting appropriate activation functions tailored to the specific requirements of the task and dataset to optimize both accuracy and execution time.

Author contributions

Conceptualization: Arjun Bala; **Methodology:** Dr. Mahesh Titiya; **Formal analysis and investigation:** Arjun Bala, Maulik Trivedi; **Writing - original draft preparation:** Arjun Bala; **Writing - review and editing:** Dr. Mahesh Titiya; **Supervision:** Dr. Mahesh Titiya

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] J. A. Hertz, Introduction to the theory of neural computation . CRC Press, 2018.

[2] L. Deng, “A tutorial survey of architectures, algorithms, and applications for deep learning,” APSIPA Transactions on Signal and Information Processing , vol. 3, p. e2, 2014.

[3] 1989K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” Computer Vision and Pattern Recognition (CVPR) , vol. 7, Dec. 2015.

[4] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” Neural Computation , vol. 1, no. 4, pp. 541–551, Dec..

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Computer vision and pattern recognition (cvpr) , 2015, pp. 1–17.

[6] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” CoRR , vol. abs/1409.1556, 2014.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proceedings of the 25th international conference on neural information processing systems - volume 1 , 2012, pp. 1097–1105.

[8] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep networks with stochastic depth.” in ECCV (4) , 2016, vol. 9908, pp. 646–661.

[9] C. Y. M. Z. Alom T. M. Taha and V. K. Asari, “The history began from alexnet: A comprehensive survey on deep learning approaches,” arXiV , Dec. 2018.

[10] K. J. Piczak, “Recognizing bird species in audio recordings using deep convolutional neural networks.” in CLEF (working notes) , 2016, pp. 534–543.

[11] M. A. Nielsen, Neural networks and deep learning . Determination Press, 2015.

[12] H. Robbins and S. Monro, “A stochastic approximation method,” Ann. Math. Statist. , vol. 22, no. 3, pp. 400–407, Sep. 1951.

[13] A. Banerjee, A. Dubey, A. Menon, S. Nanda, and G. C. Nandi, “Speaker recognition using deep belief networks,” arXiv preprint arXiv:1805.08865 , 2018.

[14] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, “A stochastic quasi-newton method for large-scale optimization,” S IAM Journal on Optimization , vol. 26, no. 2, pp. 1008–1031, 2016.

[15] 93Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, “Efficient backprop,” in Neural networks: Tricks of the trade , Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 9–50.

[16] R. Hecht-Nielsen, “Theory of the backpropagation neural network,” in Neural networks for perception , H. Wechsler, Ed. Academic Press, 1992, pp. 65–.

[17] K. Hara, H. Kataoka, and Y. Satoh, “Learning spatio-temporal features with 3D residual networks for action recognition,” in Proceedings of the ieee international conference on computer vision , 2017, pp. 3154–3160.

[18] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in Proceedings of the ieee international conference on computer vision , 2015, pp. 1026–1034.

[19] R. M. Neal, “Connectionist learning of belief networks,” Artif. Intell. , vol. 56, no. 1, pp. 71–113, Jul. 1992.

[20] L. B. Godfrey and M. S. Gashler, “A continuum among logarithmic, linear, and exponential functions, and its potential to improve generalization in neural networks,” in 7th international conference on knowledge discovery and information retrieval , 2015, pp. 481–486.

[21] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” Neurocomputing , vol. 234, pp. 11–26, 2017.

[22] A. Karpathy, “Yes you should understand backprop.” https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b, 2016.

[23] Zihan Ding, Hao Dong. "Chapter 13 Learning to Run" , Springer Science and Business Media LLC, 2020.

[24] Szandała, Tomasz. "Review and comparison of commonly used activation functions for deep neural networks." Bio-inspired neurocomputing. Springer, Singapore, 2021. 203-224.