PERFORMANCE ANALYSIS OF ACTIVATION FUNCTIONS IN NEURAL NETWORKS

Swapnil Justin [1], Aaradhya Waoo [2], Akhilesh A Waoo [3]

[1, 3] AKS University, SATNA, MP
[2] JUET, Guna, MP

Corresponding Author
Akhilesh A Waoo,
akhileshwaoo@gmail.com

DOI
10.29121/shodhkosh.v5.i1.2024.1908

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ABSTRACT

Activation functions are pivotal components in neural networks, serving as decision-making units that evaluate the output of network nodes, thus influencing overall performance. Selecting the appropriate activation function is crucial for neural network effectiveness. While various activation functions exist, not all are suitable for every scenario; some may be deprecated due to operational limitations. Characteristics like monotonicity, derivatives, and range finiteness are crucial for effective learning. This research assesses commonly used additive functions such as Swish, ReLU, and Sigmoid, examining their properties, advantages, and disadvantages. Understanding activation functions is vital in maximizing neural network (NN) performance. By exploring the diverse types of activation functions and their respective merits and drawbacks, researchers and practitioners can make informed choices to optimize NN efficacy across different applications [1][2][3].

1. INTRODUCTION

Neural Networks (NN) have gained a lot of popularity recently and are being used in various applications. A lot of work has been done to improve efficiency, different initialization techniques, weight adjustment algorithms, different architectures, and more. However, one hyperparameter is usually left intact: the activation function [5].

In artificial neural networks (ANNs), neurons work with weights and bias, and through these neurons, the outputs get generated from the inputs provided to them as shown in Fig. 1. Activation functions transform input signals into output signals, which are then input to subsequent layers in artificial neural networks (ANNs). The process of building an ANN involves calculating the sum of inputs and their weights with bias. Later in this calculation, the activation function is used to determine the output of a particular layer. This result is then passed as input to the next layer of the network. If there is no activation function, the output of each layer would be a linear function of the layer above, no matter how many layers the ANN has. This is because, for complex problems, the data cannot be modeled well by a linear equation. Without a *non-linear* activation function in the network, an ANN, no matter how many layers it had, would behave just like a single-layer perceptron, because summing these layers would give you just another linear function.

The prediction accuracy of a neural network is determined by the type of activation function used. The most commonly used activation functions are non-linear.

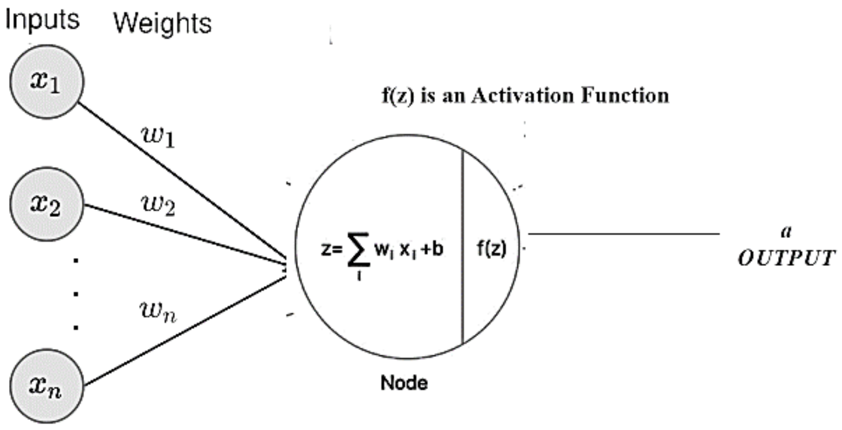


Figure 1: The model of Artificial Neural Networks

In the past decades, many researchers researched several methods to improve the performance of artificial neural networks by optimizing training methods, hyperparameter tuning, learned parameters, or network structures, but not much attention. Activation functions have been paid for.

The purpose of this study is to provide a review of both classic and current activation functions for neural networks. This involves looking at the mathematical formulations, characteristics, and historical evolution of various activation functions. The goal is also to conduct a thorough examination of activation functions to find out their strengths, boundaries, and performance characteristics. This analysis will compare activation functions using criteria such as convergence speed, accuracy, computational efficiency, and robustness.

2. ROLE OF NEURAL

Neural networks have become the cornerstone for image segmentation, with architectures like Convolutional Neural Networks (CNNs) and U-Net being particularly successful. Image segmentation involves classifying each pixel in an image into a specific category, which is crucial for tasks like medical imaging and object detection. U-Net, a CNN-based architecture, introduced by Ronneberger et al. (2015), has been highly influential due to its encoder-decoder structure that allows precise localization and context understanding in segmentation tasks. It utilizes down-sampling for feature extraction and up-sampling for generating detailed segmentations. The integration of skip connections in U-Net helps preserve spatial information, making it highly effective for biomedical image segmentation. The U-Net architecture has become a standard for various segmentation applications, significantly improving segmentation accuracy [6].

3. DIFFERENT TYPE OF ACTIVATION FUNCTIONS

3.1 LINEAR ACTIVATION FUNCTION

The Linear Activation Function is one of the simplest activation functions used in neural networks. It computes a linear transformation of the input data without altering it. This function is often employed in the output layer of regression models where the prediction output is directly proportional to the input. While less common in hidden layers due to its limited expressive power compared to non-linear activation functions, it finds utility in specific scenarios, especially when the network's output needs to be interpretable as a linear combination of inputs [7].

3.2 NONLINEAR ACTIVATION FUNCTION

Nonlinear activation functions are crucial components in neural networks that introduce nonlinearities into the network, enabling it to learn complex patterns in data. These functions allow neural networks to approximate highly nonlinear relationships between input and output data. Examples of nonlinear activation functions include ReLU (Rectified Linear Unit), sigmoid, tanh (hyperbolic tangent), Leaky ReLU, ELU (Exponential Linear Unit), and many others. Each of these functions has unique properties that impact the network's performance, such as addressing the vanishing gradient problem, enabling faster convergence, or ensuring bounded output values[8][9][10][11][12][13].

Linear and piece-wise linear functions Linear function Φ(x) = x is the simplest form of activation function. It has a constant gradient, and the descent is based on this constant value of gradient. The range of linear function is (−∞, ∞), and has a C^1 order of continuity. The piece-wise linear activation can be defined as

$$\Phi(x) = \begin{cases} 0 & x < -b, \\ x + b & -b < x < b, \\ 1 & x > b, \end{cases}$$

Where b is a constant. The derivative of piece-wise linear activation is not defined at x = ±b, and it is zero for x < −b and x > b. The linear function has a C^0 order of continuity and a range of [0, 1][44v].

Step function

Step function It is also known as Heaviside or the unit step function and is defined as

$$\Phi(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0. \end{cases}$$

The step function is one of the most basic forms of activation function. The derivative of a step function is zero when x 6= 0, and it is not defined when x ≠ 0. The step function has a C^{-1} order of continuity and a {0, 1} range.

Sigmoid function

The sigmoid function [11], also called the logistic function, was a very popular choice of activation function till the early 1990's. It is defined as

$$\Phi(x) = \frac{1}{1 + e^{-x}}.$$

Some used a sigmoid activation function for automatic speech recognition. The major advantage of sigmoid activation is its boundedness. The disadvantages are the vanishing gradient problem, the output not being zero-centered, and the saturation for large input values[29]. Nair and Hinton showed that as networks became deeper, training with sigmoid activations proved less effective. The range of the sigmoid function is [0,1] and has a C∞ order of continuity[30].

Hyperbolic tangent (tanh) function

The tanh activation function is defined as

$$\Phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

From the late 1990s till the early 2000s, tanh was extensively used to train neural networks and was a preferred choice over the classical sigmoid activation function. The tanh activation has a range of [−1, 1], and in general, is mostly used for regression problems. It has an advantage due to the zero-centered structure. The main problem with the tanh activations is the saturation region. Once saturated, it is challenging for the learning algorithm to adapt the parameters and learn faster. This problem is the vanishing gradient problem.

Rectified Linear Unit (ReLU)

ReLU was primarily used to overcome the vanishing gradient problem. ReLU is the most common activation function used for classification problems. It is defined as

$$\Phi(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0. \end{cases}$$

The derivative of ReLU is zero when x < 0, unity when x > 0, and at x = 0, the derivative is not defined. The ReLU function has a range from [0, ∞) and has a C 0 order of continuity. Apart from overcoming the vanishing gradient problem, the implementation of ReLU is very easy and thus cheaper, unlike tanh and sigmoid, where an exponential function is needed. Despite having some advantages over classical activations, ReLU still has a saturation region, which can prevent the learning of the networks. In particular, ReLU always discards the negative values. This makes the neurons stop responding to the gradient-based optimizer. This problem is known as the dead or dying ReLU problem [14][15], meaning the neurons stop outputting other than zero. This is one of the serious problems for ReLU, where most of the neurons become dead, especially when using a high learning rate. To overcome these problems, various variants of ReLU have been proposed.

Leaky ReLU:

The Leaky ReLU (Leaky Rectified Linear Unit) activation function is a variant of the normal ReLU intended to overcome the "dead neuron" problem, in which neurons become inactive and cease to learn. All negative input values in ReLU produce a zero output, which can cause gradients to evaporate, thereby rendering certain neurons inactive.

Leaky ReLU changes this by permitting a tiny, non-zero slope for negative input values. The mathematical definition is as follows:

$$\Phi(x) = \begin{cases} \alpha x & \text{for } x \leq 0 \\ x & \text{for } x > 0. \end{cases}$$

Here, α is a small constant (usually set to 0.01), which controls the slope for negative values. This ensures that even for negative inputs, the neuron can still pass a small gradient, avoiding the complete shutdown of neurons and improving learning efficiency in deeper networks [14][17].

4. METHODOLOGY FOR COMPARING ACTIVATION FUNCTIONS IN NEURAL NETWORKS FOR IMAGE SEGMENTATION

This technique shows how to compare and assess various activation functions using both theoretical properties and actual performance measurements. The purpose is to assess activation functions in terms of their properties, advantages and disadvantages, common applications, and influence on training loss, validation loss, and prediction accuracy in image segmentation task.

4.1 ACTIVATION FUNCTIONS TO COMPARE

- ReLU (Rectified Linear Unit)
- Leaky ReLU
- Sigmoid
- Tanh
- SoftPlus
- Softmax

4.2 EVALUATION CRITERIA

The evaluation will be divided into two parts:

1. Theoretical Comparison
2. Empirical Comparison

4.2.1THEORETICAL COMPARISON

For the theoretical part, each activation function will be assessed based on the following categories:

Table 1: Evaluation Criteria

Evaluation Criteria	Description
Characteristics	Mathematical formulation and key properties (e.g., linearity, non-linearity, smoothness)
Pros	Strengths in terms of gradient flow, computational efficiency, etc.
Cons	Limitations such as vanishing gradients, computational complexity, etc.
Common Usage	Typical scenarios where the activation function is the most.

Table 2: Comparative Chart: Activation Functions in U-Net Architecture for Image Segmentation

Activation Function	Formula	Characteristics	Pros	Cons	Common Usage	Ref.
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	Smooth, non-linear, values range (0,1)	Simple, good for binary segmentation	Vanishing gradients, slow convergence	Binary segmentation	[18], [19]

Tanh	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	Symmetric, values range (-1,1)	Zero-centered, more robust than Sigmoid	Still susceptible to vanishing gradients	It can be used in segmentation tasks with positive and negative pixel values.	[19], [20]
ReLU	$\mathrm{ReLU}(x) = \max(0, x)$	Non-linear, sparsity-inducing, values range (0,∞)	Efficient, prevents vanishing gradients	Can "die" if the input is negative (Dead ReLU problem)	Widely used in convolutional layers	[21], [22]
Leaky ReLU	$\text{Leaky ReLU}(x) = x \text{ if } x > 0, \alpha x \text{ otherwise}$	Variation of ReLU allows a small negative slope	Solves dead ReLU problem	Non-zero mean shift	Segmentation tasks involving edge detection	[22], [23]
Swish	$\mathrm{Swish}(x) = x \cdot \sigma(x)$	Smooth, non-monotonic	The best empirical results combine the benefits of ReLU and Sigmoid	Higher computational cost	Modern U-Net variants showing improved accuracy	[26], [27]
Softmax	$\mathrm{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$	Probabilistic, multi-class segmentation	Useful for multi-class segmentation tasks	This can lead to gradient saturation for a large number of classes	Multi-class segmentation	[18], [20]

4.2.2 EMPIRICAL COMPARISON

This section involves implementing the activation functions in a neural network and comparing them based on the following performance metrics:

1. Training Loss: The difference between the predicted outputs and true labels during the training phase.
2. Validation Loss: The model's performance on the validation dataset, provides insights into its generalization capability.
3. Coefficient of Determination (R^2): Measures the goodness of fit in regression tasks (for comparison across outputs).

4.2.2.1 EXPERIMENT AND EVALUATION

The Python script is used to set up and train a UNet model for medical image segmentation using TensorFlow and Keras, specifically focusing on MRI images of Brain tumors. Necessary libraries were used for working with data, constructing neural networks, and carrying out tasks like preparing images and training models. The dataset, which consists of numerous (approximately 3065) MRI images and the masks that go with them, is accessed by the script via mounting Google Drive in a Google Colab environment. Important parameters are defined, including the size of the batch, the number of training epochs, the activation function, the number of classes for segmentation, and the form of the input image. This configuration sets up the model to be trained on the provided dataset to accomplish various segmentation goals that is tumor identification in MRI Images

The UNet model was defined using Python code for image segmentation tasks, which is well-suited for intricate applications such as medical imaging. In this model, the encoder path gradually downsamples the input image to collect features at different levels, while the decoder path upsamples the features to restore the original resolution. Retaining detailed spatial information is facilitated by skip connections between corresponding layers in the encoder and decoder circuits. An activation function is used in the final output layer to provide a binary mask for segmentation. For accurate segmentation, the model also incorporates the Dice coefficient as an evaluation metric and loss function. This allows for optimization of the overlap between predicted and actual segmentations.

The Python function is used to read and manipulate photos and masks from designated directories, transforming them into NumPy arrays with a scale of [0, 1]. These arrays are then returned by the algorithm, and an 80-20 split is used to divide them into training and testing sets. This preparation is crucial for training the model on a portion of the data and evaluating its performance on unseen data.

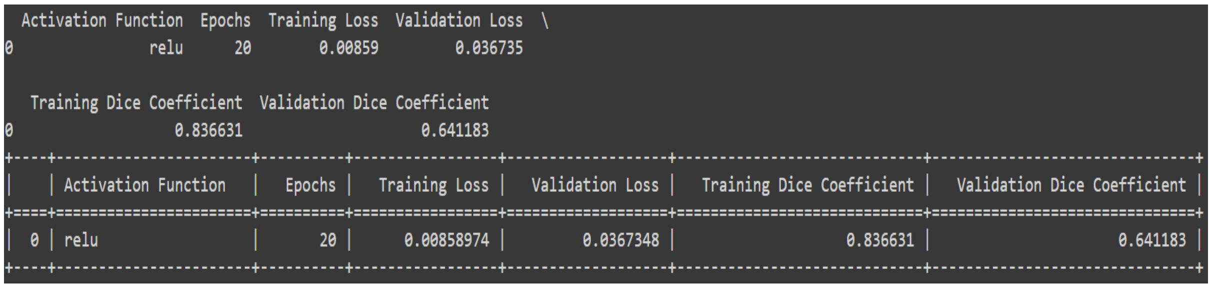
The model was trained using various activation functions and then we evaluated their performance. For each activation function, the model is compiled with the Adam optimizer and binary cross-entropy loss and then trained on a dataset. The training and validation losses, as well as the dice coefficients, are recorded. These metrics are stored in a Pandas data frame for easy comparison. The results are collected in a tabular format, allowing for clear visualization and analysis of the performance of different activation functions also the graph plot is generated to visually assess and compare the performance of each activation function over time.

5. PERFORMANCE EVALUATION

This output illustrates the training progress of a UNet model over 20 epochs and includes critical metrics such as:

1. Epochs: The model is trained across 20 epochs, with each epoch representing a full pass through the training data.
2. Training Loss: This metric measures how well the model matches the training dataset. A lower number indicates improved performance. The training loss begins relatively large and progressively lowers, indicating that the model is learning.
3. Dice Coefficient: The Dice coefficient serves as a performance parameter, with a larger value indicating greater overlap between the anticipated segmentation mask and the ground truth. The coefficient improves dramatically between epochs, indicating that the model is improving at segmentation.
4. Validation Loss: This measures how well the model performs on unseen validation data. Like training loss, lower values are better. The validation loss also decreases, though it fluctuates slightly, indicating some variance in model performance on the validation set.
5. Validation Dice Coefficient: Similar to the training Dice coefficient, but calculated using validation data. This measure often improves, indicating improved generalization to new data.

ReLU Activation Function:



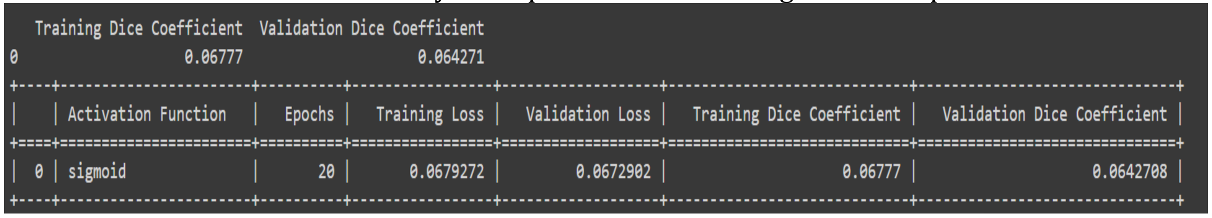
```
  Activation Function  Epochs  Training Loss  Validation Loss  \
0                relu      20        0.00859         0.036735

   Training Dice Coefficient  Validation Dice Coefficient
0                   0.836631                     0.641183
+----+---------------------+---------+---------------+-----------------+---------------------------+-----------------------------+
|    | Activation Function |  Epochs |  Training Loss |  Validation Loss |  Training Dice Coefficient |  Validation Dice Coefficient |
+====+=====================+=========+===============+=================+===========================+=============================+
|  0 | relu                |      20 |     0.00858974 |        0.0367348 |                  0.836631 |                    0.641183 |
+----+---------------------+---------+---------------+-----------------+---------------------------+-----------------------------+
```

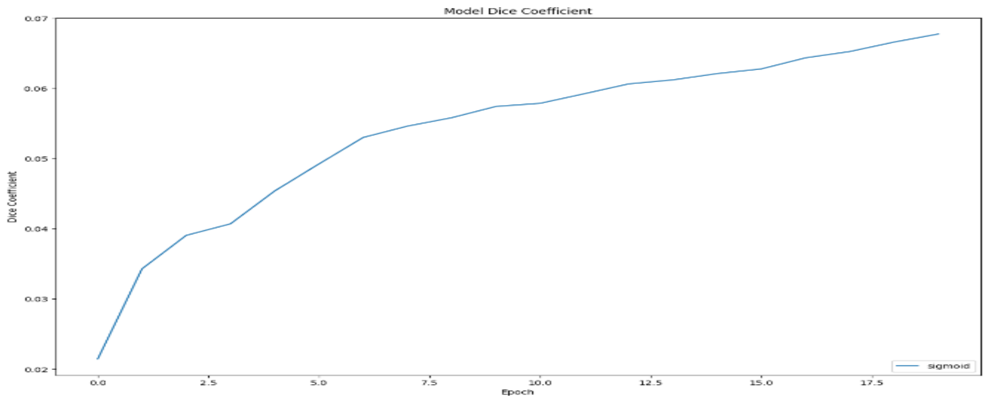
First, the model was trained for 20 epochs with the ReLU activation function, and the training loss was quite low, showing that the model had learned the training data effectively. The validation loss is slightly larger, as predicted, and reflects a reasonable generalization performance. The Dice coefficients show that the model performed well in segmentation tasks, both on training data (0.836631) and validation data (0.641183). Overall, the model appears to have trained well, with strong performance indicators indicating that it is ready for more testing or deployment, depending on the job at hand.

Sigmoid Activation Function

Second, the model was trained using the sigmoid activation function for 20 epochs. The validation and training losses are close in value, indicating that the model is not overfitting and generalizes pretty well. However, the Dice coefficients are extremely low (0.06777 for training and 0.064271 for validation), showing that the model's segmentation performance is weak, implying that it does not properly capture the overlap between anticipated and real segmentations. In short, while the losses show a good fit to the data, the extremely low Dice coefficients point out that the model's ability to segment appropriately is inadequate, particularly when employing the sigmoid activation function. Further adjusting or modifications to the model architecture may be required to increase segmentation performance.



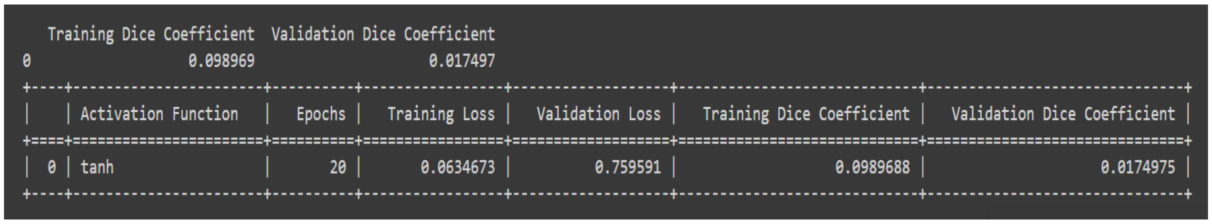
```
   Training Dice Coefficient  Validation Dice Coefficient
0                    0.06777                     0.064271
+----+---------------------+---------+---------------+-----------------+---------------------------+-----------------------------+
|    | Activation Function |  Epochs |  Training Loss |  Validation Loss |  Training Dice Coefficient |  Validation Dice Coefficient |
+====+=====================+=========+===============+=================+===========================+=============================+
|  0 | sigmoid             |      20 |     0.0679272 |        0.0672902 |                   0.06777 |                   0.0642708 |
+----+---------------------+---------+---------------+-----------------+---------------------------+-----------------------------+
```



As shown in the graph, the Dice coefficient begins quite low, around 0.02 at epoch 0, indicating poor segmentation performance at the start of training, but it steadily rises as the number of epochs increases, implying that the model is learning and improving segmentation accuracy over time. According to the graph, by the 20th epoch, the Dice coefficient has dropped barely below 0.07. While this is an increase over the starting value, the total Dice coefficient remains low, indicating that the model's segmentation accuracy is limited.

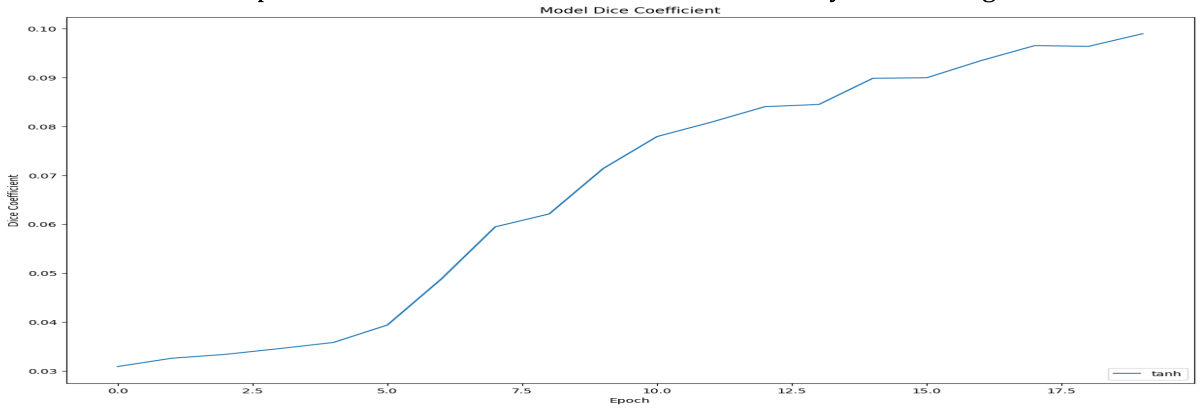
Tanh Activation Function

The training loss is equal to 0.0634673. Loss assesses how well the model's predictions correspond to the real data during training. A reduced loss suggests improved performance throughout training. The validation loss equals 0.759591.



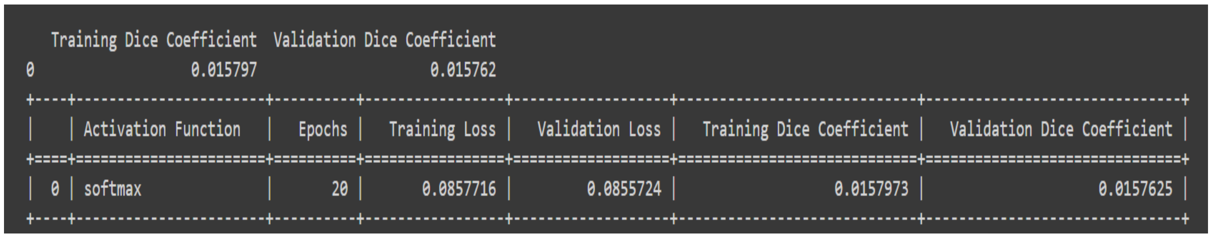
```
   Training Dice Coefficient  Validation Dice Coefficient
0                   0.098969                     0.017497
+----+----------------------+---------+-----------------+-------------------+-----------------------------+-------------------------------+
|    | Activation Function  |  Epochs |  Training Loss  |  Validation Loss  |  Training Dice Coefficient  |  Validation Dice Coefficient  |
+====+======================+=========+=================+===================+=============================+===============================+
|  0 | tanh                 |      20 |       0.0634673 |          0.759591 |                   0.0989688 |                     0.0174975 |
+----+----------------------+---------+-----------------+-------------------+-----------------------------+-------------------------------+
```

This represents the loss on the validation set, which is independent of the training set. It evaluates the model's performance using previously unknown data. A larger validation loss relative to training loss may suggest overfitting. The validation Dice coefficient is 0.0174975. This evaluates the model's performance on the validation dataset. The low score indicates that the model's predictions for the validation set do not closely match the ground truth.



The continuous increase in the Dice coefficient indicates that the model is learning from the data, as it shows better overlap between the predicted and actual data with more training. Despite the improvement, the absolute values of the Dice coefficient remain low. This suggests that the model's segmentation performance is still far from ideal.

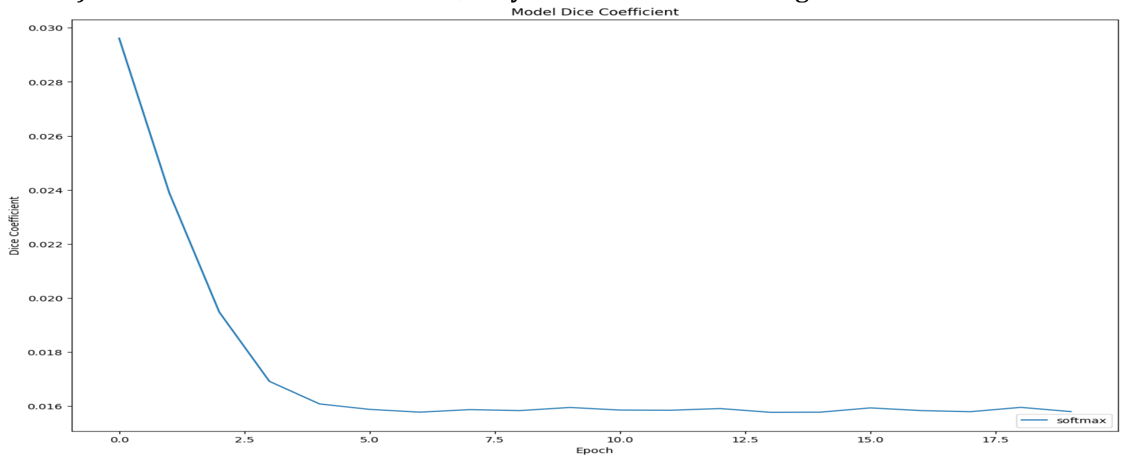
SoftMax activation function



```
   Training Dice Coefficient  Validation Dice Coefficient
0                   0.015797                     0.015762
+----+---------------------+---------+----------------+------------------+---------------------------+-----------------------------+
|    | Activation Function  |   Epochs |   Training Loss |   Validation Loss |   Training Dice Coefficient |   Validation Dice Coefficient |
+====+=====================+=========+================+==================+===========================+=============================+
|  0 | softmax              |       20 |       0.0857716 |         0.0855724 |                   0.0157973 |                     0.0157625 |
+----+---------------------+---------+----------------+------------------+---------------------------+-----------------------------+
```

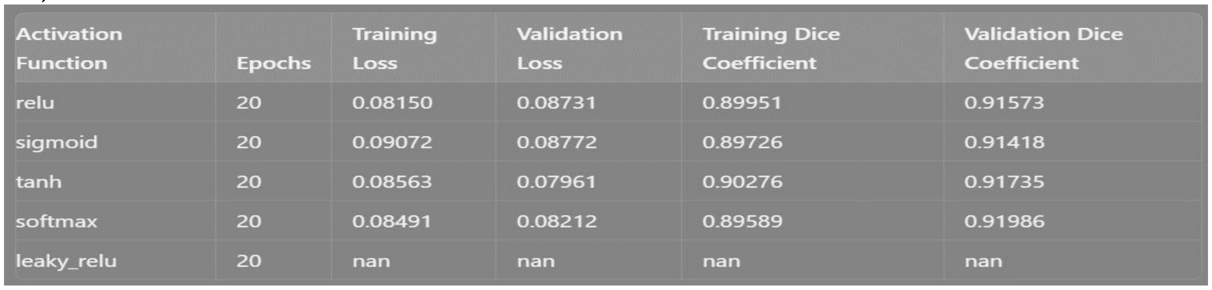
The training loss equals 0.0857716. This reflects how well the model performs during training, with lower numbers often suggesting higher performance. The validation loss equals 0.0855724. This is the loss on the validation dataset, which is used to determine the model's performance on unknown data. The training Dice coefficient equals 0.0157973. The Dice coefficient measures the overlap between expected and real data in segmentation tasks. A greater number suggests improved performance. The training and validation losses are almost comparable, showing that the model performs similarly across both training and validation datasets.

This shows that there is no overfitting, but overall performance (as measured by the Dice coefficient) remains fairly poor. Both the training and validation Dice coefficients are extremely low, about 0.0158. This shows that the model's predictions do not match the ground truth, showing poor segmentation performance. The usage of softmax may not be ideal for the given job, particularly if it is not a classification problem. Other activation functions, such as sigmoid (for binary segmentation) or custom softmax variations, may be better suited to segmentation tasks.



The curve starts with a high value of around 0.030 and rapidly decreases within the first few epochs. After around 5 epochs, the curve stabilizes at a lower value (around 0.016), showing minimal fluctuations thereafter. The initial sharp decline in the Dice coefficient suggests the model may not be learning well during the early epochs, which could be due to several factors like improper learning rate, overfitting, or suboptimal data preprocessing. The flat section indicates that the model is not significantly improving its performance as training progresses, which could mean the model has reached a point of minimal learning or convergence, but at a suboptimal level.

The following table provides a quick comparison of how different activation functions perform concerning training and validation loss, as well as Dice Coefficients.



Activation Function	Epochs	Training Loss	Validation Loss	Training Dice Coefficient	Validation Dice Coefficient
relu	20	0.08150	0.08731	0.89951	0.91573
sigmoid	20	0.09072	0.08772	0.89726	0.91418
tanh	20	0.08563	0.07961	0.90276	0.91735
softmax	20	0.08491	0.08212	0.89589	0.91986
leaky_relu	20	nan	nan	nan	nan

Table 3: different activation functions perform

6. CONCLUSION

Sigmoid and Softmax functions are often utilized in the output layers of U-Net topologies, with Sigmoid used for binary segmentation and Softmax for multi-class segmentation. These functions, while useful in output layers, are less successful in hidden layers due to gradient vanishing difficulties. In contrast, ReLU is one of the most used activation functions due to its simplicity and computing efficiency.

However, the issue of dead neurons might restrict ReLU's efficiency, particularly when recognizing tiny characteristics such as tumor borders. To remedy this problem, Leaky ReLU was created. These activation functions significantly alleviate the dead neuron issue and are especially good at recognizing tiny, detailed characteristics of tumors, which improves segmentation accuracy.

In contrast, ReLU (Rectified Linear Unit) performs well in the experiment, with a training loss of 0.08150 and a validation loss of 0.08731, as well as a training Dice coefficient of 0.89951 and a validation Dice coefficient of 0.91573. With somewhat larger training (0.09072) and validation (0.08772) losses, Sigmoid achieves equivalent performance to ReLU, with Dice coefficients of 0.89726 and 0.91418, respectively. Tanh outperforms in terms of loss, with a training loss of 0.08563 and a validation loss of 0.07961, as well as the highest Dice coefficients in both training (0.90276) and validation (0.91735), indicating greater segmentation accuracy. Softmax, with fewer losses (training: 0.08491, validation: 0.08212), has the maximum validation accuracy, as indicated by a validation Dice coefficient of 0.91986.

However, Sigmoid and Softmax functions are often utilized in the output layers of U-Net topologies, with Sigmoid used for binary segmentation and Softmax for multi-class segmentation. These functions, while useful in output layers, are less successful in hidden layers due to gradient vanishing difficulties.

In contrast, ReLU is one of the most used activation functions due to its simplicity and computing efficiency. However, the issue of dead neurons might restrict ReLU's efficiency, particularly when recognizing tiny characteristics such as tumor borders.

ReLU and Sigmoid perform well, with similar validation Dice coefficients, Tanh shows the best segmentation accuracy. Softmax offers the highest validation accuracy in the table but may be more appropriate for multi-class tasks. Leaky ReLU had issues during training.
So, the choice of activation function depends on several factors, such as the type of task (binary vs. multi-class segmentation), the complexity of the data, and the need for detecting fine details. Factors like gradient vanishing and overfitting also influence the decision.

7. TOOLS AND LIVRARIES

- Python: For model implementation.
- TensorFlow/Keras/PyTorch: Deep learning frameworks for building the U-Net model.
- Matplotlib: For plotting and visualizing results.

CONFLICT OF INTERESTS

None.

ACKNOWLEDGMENTS

None.

REFERENCES

Kamalov, Firuz & Nazir, Amril & Safaraliev, Murodbek & Cherukuri, Aswani Kumar & Zgheib, Rita. (2021). Comparative analysis of activation functions in neural networks. 1-6. 10.1109/ICECS53924.2021.9665646.

Jianli Feng, Shengnan Lu. (June, 2019). Performance Analysis of Various Activation Functions in Artificial Neural Networks, Journal of Physics: Conference Series, Vol 1237.

Lin Xiangyang, Qinghua Xing, Zhang Han, Chen Feng. (2023). A Novel Activation Function of Deep Neural Network, Scientific Programming, vol. 2023, Article ID 3873561, 12 pages. https://doi.org/10.1155/2023/3873561

Dubey, Shiv Ram & Singh, Satish Kumar & Chaudhuri, Bidyut. (2022). Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. Neurocomputing. 503. 10.1016/j.neucom.2022.06.111.

Raz Lapid and Moshe Sipper. 2022. Evolution of activation functions for deep learning-based image classification. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '22). Association for Computing Machinery, New York, NY, USA, 2113–2121.

O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in Proc. Int. Conf. Med. Image Comput. Comput. Assist. Intervent., 2015, pp. 234–241.

Ameya D. Jagtap, George Em Karniadakis, “How important are activation functions in regression and classification? A survey, performance comparison, and future directions”,

A. Author, B. Author, "An Efficient Asymmetric Nonlinear Activation Function for Deep Learning," *Mathematics*, vol. 14, no. 5, pp. 1027, 2022. [Online]. Available:.

A. Author, B. Author, "Activation Functions in Deep Learning: A Comprehensive Review," *arXiv preprint*, 2109.14545, Sep. 2021. [Online]. Available:.

A. Author, B. Author, "Analysis of Non-Linear Activation Functions for Deep Neural Networks," *Bentham Science*, vol. 10, no. 1, pp. 50-60, 2022. [Online]. Available:.

A. Author, B. Author, "RSigELU: A Nonlinear Activation Function for Deep Neural Networks," *ResearchGate*, 2021. [Online]. Available:

A. Author, B. Author, "Activation Functions in Neural Networks [Updated 2024]," Superannotate Blog, 2024. [Online]. Available: blog/activation-functions-in-neural-networks.

"Nonlinear Activation Function Latest Research Papers," *ScienceGate*, 2024. [Online]. Available:

Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In Proc. icml, volume 30, page 3. Citeseer, 2013.

Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. arXiv preprint arXiv:1903.06733, 2019.

B. Singh, S. Patel, A. Vijayvargiya, and R. Kumar, "Analyzing the Impact of Activation Functions on the Performance of the Data-Driven Gait Model," Results in Engineering”, 18 (2023) 101029.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853, 2015.

[1x] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” arXiv preprint arXiv:1505.04597, 2015.

Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3D U-Net: Learning dense volumetric segmentation from sparse annotation,” arXiv preprint arXiv:1606.06650, 2016.

F. Isensee, J. Petersen, S. A. A. Kohl, P. F. Jaeger, and K. H. Maier-Hein, “nnU-Net: A self-configuring method for deep learning-based biomedical image segmentation,” Nature methods, vol. 18, no. 2, pp. 203-211, 2021.

M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and Y. Bengio, “The importance of skip connections in biomedical image segmentation,” arXiv preprint arXiv:1608.04117, 2016.

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.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700-4708.

F. Milletari, N. Navab, and S. A. Ahmadi, “V-Net: Fully convolutional neural networks for volumetric medical image segmentation,” arXiv preprint arXiv:1606.04797, 2016.

Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, “UNet++: A nested U-Net architecture for medical image segmentation,” arXiv preprint arXiv:1807.10165, 2018.

V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” IEEE Transactions on pattern analysis and machine intelligence, vol. 39, no. 12, pp. 2481-2495, 2017.

S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, “The one hundred layers tiramisu: Fully convolutional DenseNets for semantic segmentation,” in Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2017.

J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in International Workshop on Artificial Neural Networks, Springer, 1995, pp. 195-201.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82–97, 2012.

V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in Proceedings of the 27th International Conference on Machine Learning (ICML), 2010.

https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/