International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

Optimizing Neural Networks: A Comparative Analysis of Activation Functions in Deep Learning

Mainak Mitra[1], Soumit Roy[2], Vikram Maghnani[3]

[1]Technical Manager Cognitive and Analytics, Deloitte, San Jose, United States
Email: *mainakmitra1210@email.com*

[2]Data Analytics Architect, TCS, Chicago, United States
Email: *soumit123@gmail.com*

[3]Technical Product Manager, eGain, Sunnyvale, United States
Email: *vikramce@gmail.com*

Abstract: ***This paper provides a comprehensive analysis of activation functions in multilayer neural networks. The study delves into the role of different activation functions and their impact when combined with various loss functions. It focuses on how activation functions contribute to neuron activation based on weight calculations and inclinations, thereby introducing nonlinearity in neuron outputs. The research highlights the importance of back-propagation in neural networks facilitated by activation functions. This study aims to offer clear guidance on selecting the appropriate activation function for specific problems and parameters.***

Keywords: Neural Networks, Activation Functions, Deep Learning, Back-Propagation, Loss Functions

1. Introduction

Neural Networks based on Deep Learning, is a sub-field of Machine Learning where the Algorithm is inspired by the structure of the human brain. Neural Networks take in the data and then train themselves to recognize the patterns with the help of the numerical value and predict the output for the new set of similar data.

Neural networks are made with layers of neurons which are the core processing units. First, we have the input layer which receives the input, where the pixel is the input. We have the output layer where decisions will be shown. In between them, we have the Hidden layer which performs most of the computation required by our network. In neural networks layers are connected with the **Weight**, these weights are assigned with numerical values. Also, inputs are multiplied with the corresponding weight, and their sum is sent as input into the hidden layers, In hidden layers each neuron is associated with a numerical value called **Bias** which is then added to the input sums. After that, all those values pass through a threshold function known as the **Activation Function.**

The result of the activation function determines if the particular neuron will get activated or not. Activated neurons then transmit the data to the neuron of the next layer, in this way data is propagated through the network which is called Forward Propagation. In the output layer the neuron with the highest value fires and determines the output and outputs are nothing but just a probability.

If our actual pattern is a circle but in the output layer highest probability is square, that's the output predicted by the neural network which is a wrong prediction. Our network is yet to be trained during this training process along with the input our network also has the output fed to it, the predicted output is compared against the actual output to realize the prediction error the magnitude of the error indicates how wrong we are and this information is then transferred backwards through our network, this is known as **Backward Propagation.** Now with this new information, the weights are adjusted, and this cycle of forward propagation and backward propagation is iteratively performed with multiple inputs. This process continues until our weights are assigned such that the network can predict the shapes accurately.

Purpose: The purpose of this article is to analyze the effectiveness of various activation functions in neural networks and to provide guidelines for selecting the most appropriate function based on the specific problem and loss function being addressed.

Significance: The significance of this article lies in its detailed examination of how different activation functions affect the performance of neural networks. This understanding is crucial for optimizing neural network models in various applications, ranging from pattern recognition to data prediction.

2. Components of Neural Network

2.1 Neuron (or Node)

Neurons, the core units of a neural network that receives data as input and processes with activation functions then produce certain outcomes.

2.2 Layer

In neural networks, there is only one input layer that receives the initial data and one output layer where we see the produced or predicted data and in between them, there can be one or more hidden layers. This layer performs critical computations with activation function; in those layers neurons are well organized.

2.3 Weights and Biases

Every layer connected with the channels is called Weight, these weights determine the strength of the connections. Now those weights are connected with a numerical entity which is called Bias, Bias has constant values that are added to the weighted sum of input then the activation function is applied to them.

2.4 Activation Function

Activation functions introduce non-linearity to the network, allowing it to learn complex patterns and relationships in the data and determine the output of a neuron based on its input. Common activation functions include sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU).

2.5. Feed Forward

When data moves or propagates from the input layer to the output layer in our neural network that process is called FeedForward or Forward Propagation, Each neuron gets the value as input from the previous layer, after applying the activation function, and passes the output to the next layer.

2.6. Back propagation

When we get some value after forward propagation our neural network estimates the error between the predicted output and the actual output then with those values neural network adjusts its weights and bias this process continues until the network's performance is satisfactory, this process is called Backward Propagation or Backpropagation.

2.7 Loss Function

The difference between the predicted output and the actual output is called the Loss function, when building the neural network somewhere our goal is to minimize this loss. There are some loss functions such as mean squared error (MSE) for regression tasks and classification tasks commonly used as categorical cross-entropy.

2.8 Gradient Descent

This is nothing but an optimization algorithm that is used during the training of our neural network, Gradient Descent adjusts the weights and biases in the direction that reduces the loss.

2.9 Epochs and Batch Size

In simple words when forward propagation and backward propagation are done then we can say that accumulated one Epoch, Epoch is the main reason behind our neural network model training. In the context of neural network training, the model makes predictions for each training example, computes the loss or error, and updates its weights to minimize this error.

3. Introduction to Activation Function

Activation functions determine whether a neuron should be activated or not by calculating the weighted sum and advance including its inclination. The reason for the activation function is to present non-linearity in the yield of a neuron.

We know that the neural network has neurons that work in correspondence with weight, bias, and their particular activation function. in a neural network, we overhaul. This process is known as back-propagation. Activation functions make the back-propagation conceivable since the slopes or gradients are provided at the side of the error to update the weights and biases.

Consider a simple neural network model without any hidden layers.

Equation for output label
It can range from $-\infty$ to $+\infty$. So it is necessary to bound the output to get the desired prediction or generalized results

So the activation function is an important part of a neural network. They determine whether a neuron should be activated and apply a non-linear transformation.

The activation function is a crucial element to decide whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. Activation function introduces **nonlinearity** into the output of a neuron.

4. Purpose of Activation Function

4.1 Avoiding the "Dying Neuron" Problem

During training, the activation function prevents the neuron from becoming inactive, when neurons stop learning then it is called a dying neuron, a dying neuron consistently outputs zero or near-zero values.

4.2 Handling the Vanishing Gradient Problem

Without non-linear activation functions, gradients can diminish rapidly when they backpropagate which hinders the learning of deep hierarchical features.

4.3 Facilitating Generalization

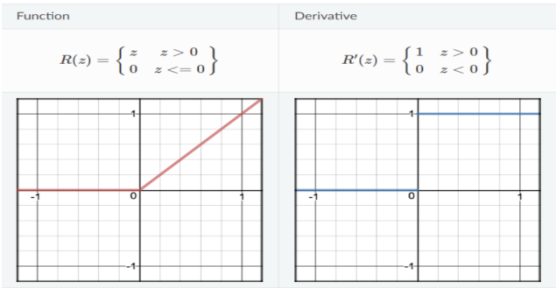
The non-linear activation function has the ability to capture the patterns which have the nature of non-linearity to make an accurate prediction, enabling a language model to understand and the context of the word in the sentence is enabled by the non-linear activation function also improves the generalization to new sentences.

5. Different Activation Functions

5.1 Rectified Linear Unit (ReLU):

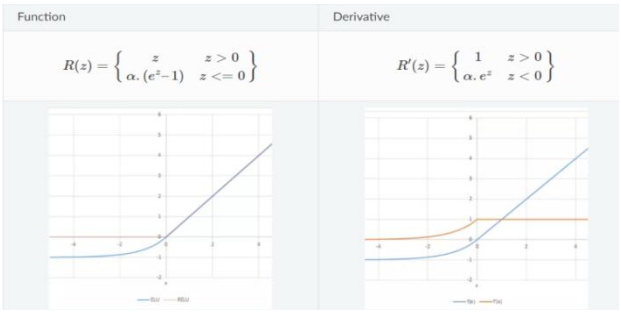
A recent invention which stands for Rectified Linear Units. The formula is deceptively simple: max(0,z). Despite its

name and appearance, it is not linear. and provides the same benefits as Sigmoid (i.e. the ability to learn nonlinear functions), but with better performance.



5.2 Exponential Linear Unit (ELU):

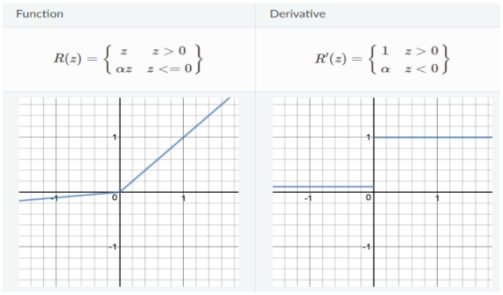
ELU tends to converge cost to zero faster and produce more accurate results. The alpha constant in ELU should be a positive number. ELU becomes smooth slowly until its output is equal to -α.



5.3 Leaky RELU

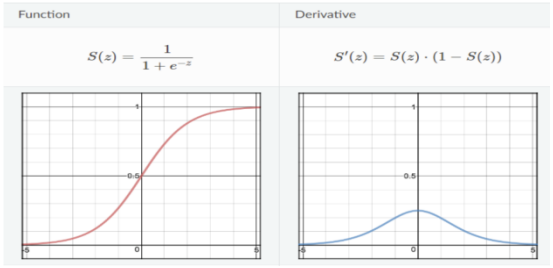
Leaky ReLU is a variant of ReLU. Instead of being 0 when z<0, a leaky ReLU allows a small, non-zero, constant gradient α (Normally, α=0.01.

Leaky ReLU(x)=max(αx,x) where α is a small positive constant.



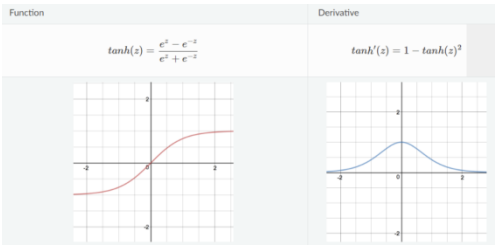
5.4 Sigmoid

The Sigmoid function takes a real value as input and outputs another value between 0 and 1. It's easy to work with and has all the nice properties of activation functions: it's non-linear, continuously differentiable, monotonic, and has a fixed output range.



5.5 Tanh

Tanh compresses a real-valued number to the range [-1, 1]. It's non-linear. But unlike Sigmoid, its output is zero-centered. Therefore, in practice, the tanh nonlinearity is always preferred to the sigmoid nonlinearity.



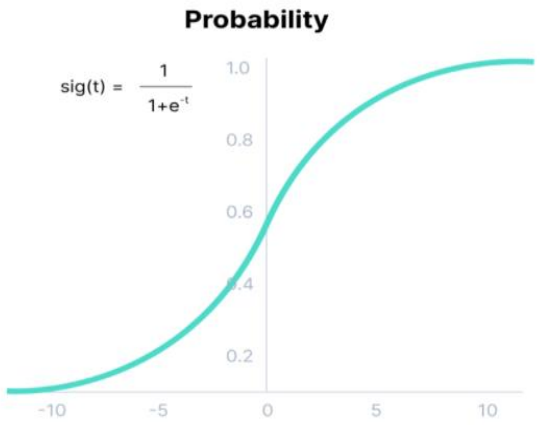
5.5 Softmax

Softmax function calculates the probability distribution of the event over 'n' different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

6. General guidance to choose the right Activation Function

It is important to ensure that the activation function for the output layer is selected based on the type of prediction problem we are solving, to be specific the type of predicted variable.

As a rule of thumb, we begin with the ReLU activation function and then move over to other activation functions if ReLU doesn't provide optimum results.



6.1 Consideration to choose activation functions for better outcomes:

1) The ReLU activation function is best suited for the hidden layers.
2) Sigmoid/Logistic and Tanh are not good for hidden layers as they may create problems due to vanishing gradients
3) Swish function is better if the depth of the network is greater than 40 layers.

Table 1: Guideline to select Activation function

Problem Type	Loss Function	Activation Function
Binary Classification	Binary Crossentropy	Sigmoid or Logistic
Multi-class Classification	Categorical Crossentropy	Softmax
Regression (Predicting a continuous value)	Mean Squared Error	Linear, ReLU, ELU
Regression with Bounded Output	Mean Squared Error or Mean Absolute Error	Sigmoid
Custom Scenarios	If you have a custom scenario or a specific goal, you may need to experiment with different activation and loss functions.	

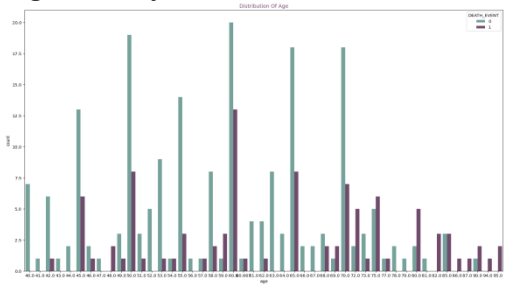
7. Test cycle with different choices of Loss function and Activation Functions

We will utilize a heart failure clinical record dataset and ANN model with several activation functions with the same loss function.

Cases of Cardiovascular diseases (CVDs) are the primary cause of death worldwide. Considering an estimated 17.9 million deaths each year, CVDs account for 31% of all deaths worldwide. This dataset contains 12 features that can be used to predict mortality by heart failure.

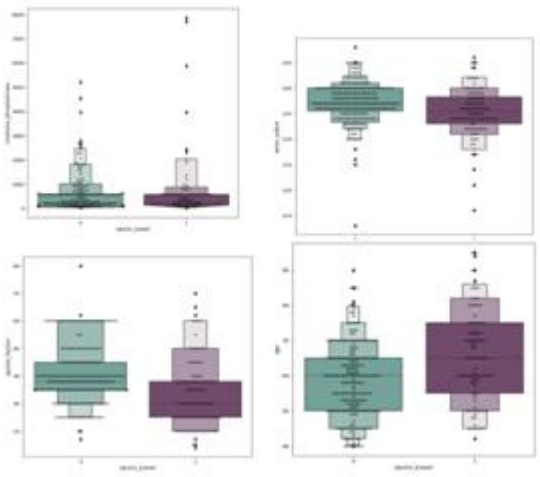
8. Exploratory analysis of the data

8.1 Histogram Analysis:

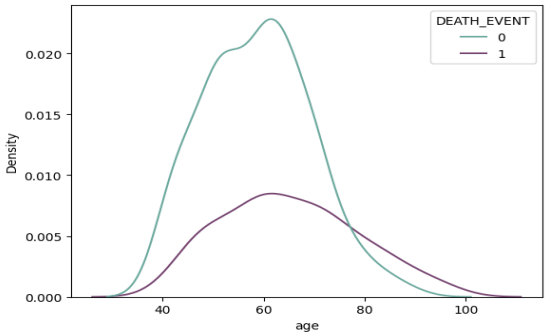


We understand that mostly 50 to 70 year old people are prone to heart disease.

8.2 Outlier Analysis:



8.3 Probability Distribution using KDE analysis:

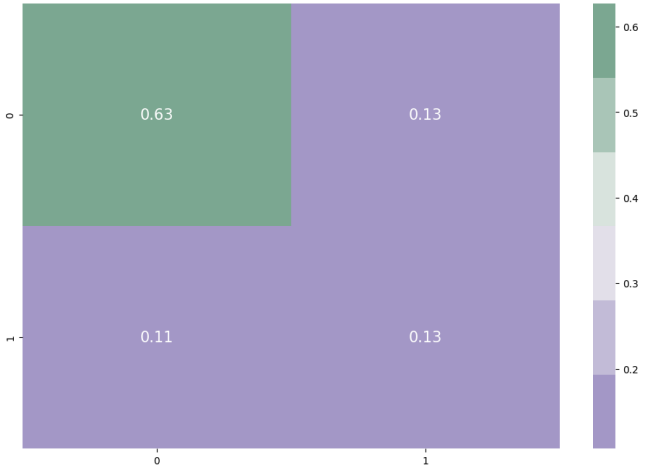


9. ANN Model for the binary classification

- Four hidden layers were used, with Uniform or Gaussian distribution as kernel initializers.
- First three layers we have used activation function is ReLU
- In the last hidden layer, we used sigmoid with the **adam** optimization and **mean_squared_error** loss function

9.1 Confusion Matrix

```
cmap1 = sns.diverging_palette(275,150,  s=40, l=65, n=6)
plt.subplots(figsize=(12,8))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), cmap = cmap1,
annot = True, annot_kws = {'size':15})
```



9.2Classification Report

	precision	recall	f1-score	support
0	0.86	0.89	0.88	57
1	0.62	0.56	0.59	18
accuracy			0.81	75
macro avg	0.74	0.73	0.73	75
weighted avg	0.81	0.81	0.81	75

- The Model looks good with avg 81% validation accuracy.
- Multiple rounds were performed with different combinations of activation function and loss functions for Classification and Regression problems but have not attached the code here.
- However it should be very straight forward to leverage the same model but play around with the activation functions and loss functions.
- We have conducted a similar analysis. With different loss function and activation function to get the essence of which Activation function is best suited for a given scenario.

9.4 Test Results Analysis:

Following charts will show the intuitive analysis to visualize the performance of various activation functions under different scenarios

Individuals with cardiovascular disease or at high cardiovascular risk need early detection and management wherein a ML or DL model can be of great help.

We will predict the survival of patients with heart failure from serum creatinine and ejection fraction alone.

This test will be performed with various combinations to establish a scientific approach of selecting best Activation function for the specific type of problem

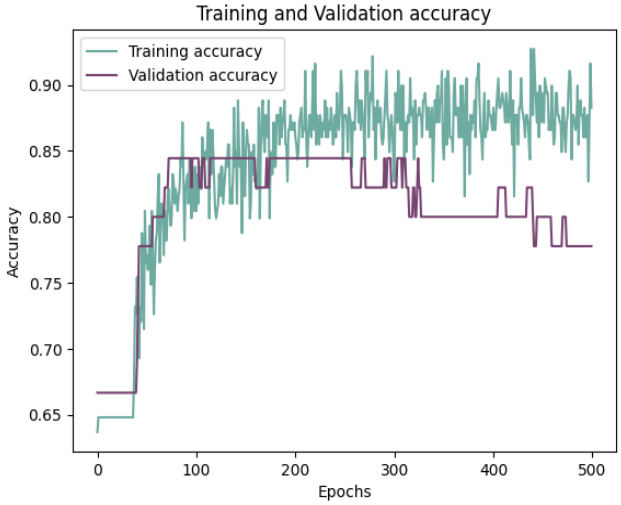
We will perform this test with different combinations.

9.5 Code Snippet

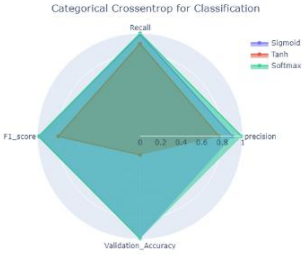
```
# Initializing the NN
model = Sequential()
# layers
model.add(Dense(units = 16, kernel_initializer = 'uniform',
activation = 'relu', input_dim = 12))
model.add(Dense(units = 8, kernel_initializer = 'uniform',
activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(units = 4, kernel_initializer = 'uniform',
activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 1, kernel_initializer = 'uniform',
activation = 'sigmoid'))
from keras.optimizers import SGD
# Compiling the ANN
model.compile(optimizer = 'adam', loss =
'mean_squared_error', metrics = ['accuracy'])

# Train the ANN
history = model.fit(X_train, y_train, batch_size = 32, epochs
= 500, validation_split=0.2)
```

Validation Accuracy:

```
history_df = pd.DataFrame(history.history)
plt.plot(history_df.loc[:, ['accuracy']], "#6daa9f",
label='Training accuracy')
plt.plot(history_df.loc[:, ['val_accuracy']], "#774571",
label='Validation accuracy')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

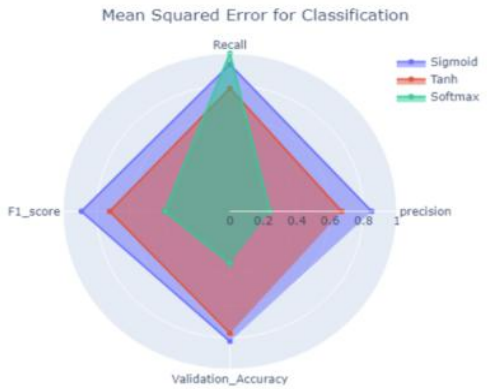


9.5.1 Classification: Categorical_Crossentropy(Table2)



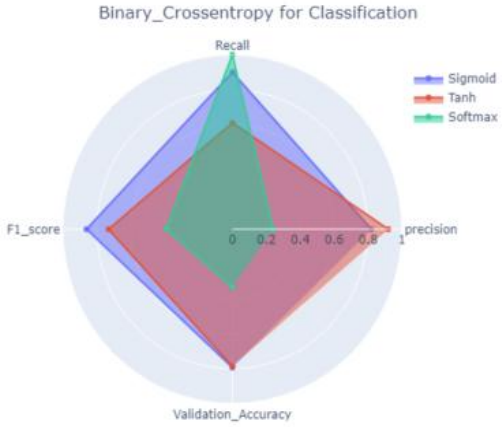
Recommended activation function is **Softmax**

9.5.2 Classification: Mean_squared_error(Table3)



Recommended Activation Function is **Sigmoid**.

9.5.3 Classification: Binary_Crossentropy(Table4)

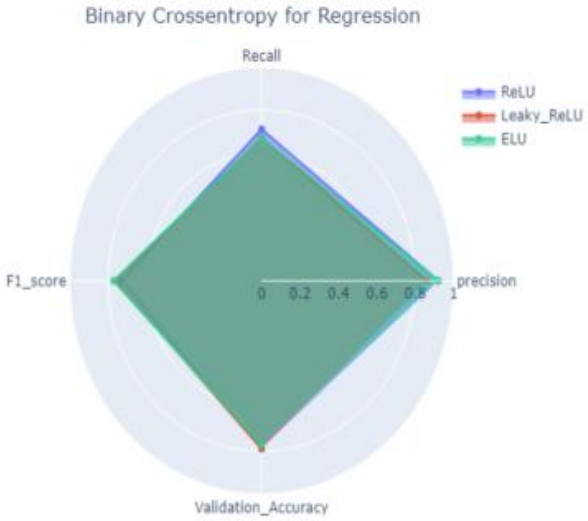


Recommended Activation Function is **Sigmoid**.

9.5.4 Regresion: Mean_squared_error(Table5)

Recommended Activation Function is **ELU**.

9.5.5 Regression: Binary_Crossentropy(Table6)



Recommended Activation Function is **RELU** or **ELU**.

Table 2: Classification Problems: Loss Function Categorical_Crossentropy

Loss Function: Categorical_crossentropy					
Activation Function: Sigmoid					
	Precision	**Recall**	**F1_score**	**Support**	**Validation Accuracy**
0	0.99	0.99	0.99	980	
1	0.99	1	1	1135	
Accuracy			0.99	10000	99.13%
Macro Avg	0.99	0.99	0.99	10000	
Weighted Avg.	0.99	0.99	0.99	10000	
Activation Function: Tanh					
	Precision	**Recall**	**F1_score**	**Support**	**Validation Accuracy**
0	0.21	0.91	0.34	980	
1	0.16	0.9	0.27	1135	
Accuracy			0.18	10000	18.14%
Macro Avg,	0.04	0.18	0.06	10000	
Weighted Avg.	0.04	0.18	0.06	10000	
Activation Function: Softmax					
	Precision	**Recall**	**F1_score**	**Support**	**Validation Accuracy**
0	0.99	1	0.99	980	
1	0.99	1	0.99	1135	
Accuracy			0.99	10000	99.13%
Macro Avg,	0.99	0.99	0.99	10000	
Weighted Avg.	0.99	0.99	0.99	10000	

Table 3: Classification Problems: Mean_squared_error

Loss Function: Mean_squared_error					
Activation Function: Sigmoid					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0.85	0.93	0.89	57	
1	0.69	0.5	0.58	18	
Accuracy			0.83	75	82.36%
Macro Avg,	0.77	0.71	0.74	75	
Weighted Avg.	0.82	0.83	0.82	75	
Activation Function: Tanh					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0.93	0.88	0.9	57	
1	0.67	0.78	0.72	18	
Accuracy			0.85	75	77.20%
Macro Avg,	0.8	0.83	0.81	75	
Weighted Avg.	0.86	0.85	0.86	75	
Activation Function: Softmax					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0	0	0	57	
1	0.24	1	0.39	18	
Accuracy			0.24	75	33.33%
Macro Avg,	0.38	0.5	0.1	75	
Weighted Avg.	0.58	0.24	0.09	75	

Table 4: Classification Problems: Binary_cross_entropy

Loss Function: Binary_crossentropy					
Activation Function: Sigmoid					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0.85	0.93	0.89	57	
1	0.69	0.5	0.58	18	
Accuracy			0.83	75	79.21%
Macro Avg,	0.77	0.71	0.74	75	
Weighted Avg.	0.82	0.83	0.82	75	
Activation Function: Tanh					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0.89	0.98	0.93	57	
1	0.92	0.61	0.73	18	
Accuracy			0.89	75	78.61%
Macro Avg,	0.9	0.8	0.83	75	
Weighted Avg.	0.9	0.89	0.89	75	
Activation Function: Softmax					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0	0	0	57	
1	0.24	1	0.39	18	
Accuracy			0.24	75	33.33%
Macro Avg,	0.12	0.5	0.18	75	
Weighted Avg.	0.06	0.24	0.09	75	

Table5:Regression Problems: Mean_squared_error

Loss Function: Categorical_crossentropy					
Activation Function: ReLU					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0.85	0.88	0.86	57	
1	0.56	0.5	0.53	18	78.90%
Accuracy			0.79	75	
Macro Avg,	0.7	0.69	0.7	75	
Weighted Avg.	0.78	0.79	0.78	75	
Activation Function: Leaky ReLU					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0.92	0.95	0.93	57	
1	0.82	0.72	0.76	18	75.83%
Accuracy			0.89	75	
Macro Avg,	0.86	0.83	0.85	75	
Weighted Avg.	0.89	0.89	0.89	75	
Activation Function: ELU					
	Precision	Recall	F1_score	Support	Validation Accuracy

0	0.9	0.9	0.94	57	
1	0.92	0.67	0.77	18	
Accuracy			0.91	75	81.79%
Macro Avg,	0.91	0.82	0.86	75	
Weighted Avg.	0.91	0.91	0.9	75	

Table 6: Regression Problems: Binary_cross_entropy

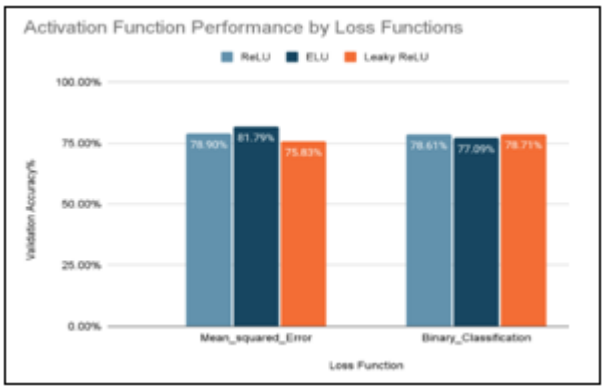
Loss Function: Categorical_crossentropy					
Activation Function: ReLU					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0.89	0.98	0.93	57	
1	0.92	0.61	0.73	18	
Accuracy			0.89	75	78.61%
Macro Avg,	0.9	0.8	0.83	75	
Weighted Avg.	0.9	0.89	0.89	75	
Activation Function: Leaky ReLU					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0.9	0.96	0.93	57	
1	0.86	0.67	0.75	18	
Accuracy			0.89	75	78.71%
Macro Avg,	0.88	0.82	0.84	75	
Weighted Avg.	0.89	0.89	0.89	75	
Activation Function: ELU					
	Precision	Recall	F1_score	Support	Validation Accuracy
0	0.9	0.9	0.94	57	
1	0.92	0.67	0.77	18	
Accuracy			0.91	75	77.09%
Macro Avg,	0.91	0.82	0.86	75	
Weighted Avg.	0.91	0.91	0.9	75	

10. Conclusion

The study conclusively demonstrates that the choice of activation function significantly impacts the performance of neural networks. It emphasizes the necessity of selecting the right activation function based on the specific problem type and loss function, leading to improved accuracy and efficiency in deep learning models.

10.1 Regression

Through extensive experimentation with various activation and loss functions in regression tasks, we have observed that the choice profoundly impacts model accuracy. Certain activation functions, such as ReLU, are well-suited for capturing complex patterns in continuous data, while specific loss functions, like Mean Squared Error, prove effective in minimizing prediction errors. This nuanced interplay emphasizes the critical role of selecting appropriate functions tailored to the nature of the regression problem for optimal model performance.



10.2. Classification

Our thorough exploration of activation and loss functions highlights their pivotal role in shaping the models outcomes. Activation functions like Softmax facilitate effective probability distribution among classes, while cross-entropy loss functions prove valuable in minimizing classification errors. The interplay of these functions reveals the intricate relationship between neural network architecture and task-specific objectives, underscoring the significance of thoughtful function selection for achieving accurate and reliable classification models.

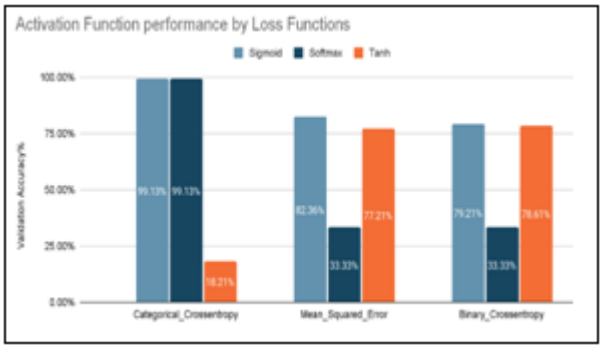


Table 8: Final Recommendation

Regression		
Activation Function	**Loss Functions**	
	Mean Squared Error	Binary Crossentropy
ReLU	78.90%	78.61%
ELU	81.79%	77.09%
Leaky ReLU	75.83%	78.71%

Classification			
Activation Function	**Loss Functions**		
	Categorical Cross Entropy (Multi Class)(to be filled up for Multi Class)	Mean Squared Error(Regression Bounded Output)	Binary Cross Entropy(Binary Classification)
Sigmoid	99.13%	82.36%	79.21%
Softmax	99.13%	33.33%	33.33%
Tanh	18.21%	77.20%	78.61%

References

[1] Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions." arXiv preprint arXiv:1710.05941 (2017).
[2] Nie, Xiaobing, and Wei Xing Zheng. "Complete stability of neural networks with nonmonotonic piecewise linear activation functions." IEEE Transactions on Circuits and Systems II: Express Briefs 62.10 (2015): 1002-1006.
[3] Chandra, Pravin, Udayan Ghose, and Apoorvi Sood. "A non-sigmoidal activation function for feedforward artificial neural networks." 2015 International Joint Conference on Neural Networks (IJCNN). IEEE, 2015.
[4] Montufar, Guido F., et al. "On the number of linear regions of deep neural networks." Advances in neural information processing systems 27 (2014).
[5] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2010.
[6] Srivastava, Rupesh K., Klaus Greff, and Jürgen Schmidhuber. "Training very deep networks." Advances in neural information processing systems 28 (2015).
[7] Gordon, Siamon, and Fernando O. Martinez. "Alternative activation of macrophages: mechanism and functions." Immunity 32.5 (2010): 593-604.
[8] Sharma, Sagar, Simone Sharma, and Anidhya Athaiya. "Activation functions in neural networks." Towards Data Sci 6.12 (2017): 310-316.
[9] Willmott, Cort J., and Kenji Matsuura. "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance." Climate research 30.1 (2005): 79-82.
[10] Wang, Zhou, and Alan C. Bovik. "Mean squared error: Love it or leave it? A new look at signal fidelity measures." IEEE signal processing magazine 26.1 (2009): 98-117.
[11] Guo, Dongning, Shlomo Shamai, and Sergio Verdú. "Mutual information and minimum mean-square error in Gaussian channels." IEEE transactions on information theory 51.4 (2005): 1261-1282.
[12] Gulcehre, Caglar, et al. "Noisy activation functions." International conference on machine learning. PMLR, 2016.
[13] Dushkoff, Michael, and Raymond Ptucha. "Adaptive activation functions for deep networks." Electronic Imaging 2016.19 (2016): 1-5.
[14] Lau, Mian Mian, and King Hann Lim. "Review of adaptive activation function in deep neural network." 2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES). IEEE, 2018.
[15] Tang, Yichuan. "Deep learning using linear support vector machines." arXiv preprint arXiv:1306.0239 (2013).
[16] Wanto, Anjar, et al. "Use of binary sigmoid function and linear identity in artificial neural networks for forecasting population density." IJISTECH (International Journal of Information System and Technology) 1.1 (2017): 43-54.
[17] Ide, Hidenori, and Takio Kurita. "Improvement of learning for CNN with ReLU activation by sparse regularization." 2017 international joint conference on neural networks (IJCNN). IEEE, 2017.
[18] Zhang, Chao, and Philip C. Woodland. "Parameterised sigmoid and ReLU hidden activation functions for DNN acoustic modelling." Sixteenth annual

conference of the international speech communication association. 2015.
[19] Abdelouahab, Kamel, Maxime Pelcat, and François Berry. "Why TanH is a hardware friendly activation function for CNNs." Proceedings of the 11th international conference on distributed smart cameras. 2017.
[20] Yarotsky, Dmitry. "Error bounds for approximations with deep ReLU networks." Neural Networks 94 (2017): 103-114.
[21] Rajaguru, Harikumar, and Sunil Kumar Prabhakar. "An approach to classification of oral cancer using Softmax Discriminant Classifier." 2017 2nd International Conference on Communication and Electronics Systems (ICCES). IEEE, 2017.
[22] Xin, Bin, Tianzhen Wang, and Tianhao Tang. "A deep learning and softmax regression fault diagnosis method for multi-level converter." 2017 IEEE 11th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED). IEEE, 2017.
[23] Mao, Xudong, et al. "Multi-class generative adversarial networks with the L2 loss function." arXiv preprint arXiv:1611.04076 5 (2016): 1057-7149.
[24] Wanto, Anjar, et al. "Use of binary sigmoid function and linear identity in artificial neural networks for forecasting population density." IJISTECH (International Journal of Information System and Technology) 1.1 (2017): 43-54.
[25] Hipkins, Rosemary, and Bronwen Cowie. "The sigmoid curve as a metaphor for growth and change." Teachers and Curriculum 16.2 (2016).
[26] Si, Jiong, Sarah L. Harris, and Evangelos Yfantis. "A dynamic ReLU on neural network." 2018 IEEE 13th Dallas Circuits and Systems Conference (DCAS). IEEE, 2018.
[27] Laurent, Thomas, and James Brecht. "The multilinear structure of ReLU networks." International conference on machine learning. PMLR, 2018.
[28] Qian, Sheng, et al. "Adaptive activation functions in convolutional neural networks." *Neurocomputing* 272 (2018): 204-212.
[29] https://www.geeksforgeeks.org/activation-functions-neural-networks/
[30] https://www.slideshare.net/khaledalhalabee/artificial-neural-network-247800756
[31] https://github.com/MoinDalvs/Neural_Networks_From_Scratch
[32] https://link.springer.com/chapter/10.1007/978-3-030-33110-8_1?
[33] https://pdffox.com/ml-cheatsheet-documentation-pdf-free.html
[34] https://pl.icourban.com/crypto-https-ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html