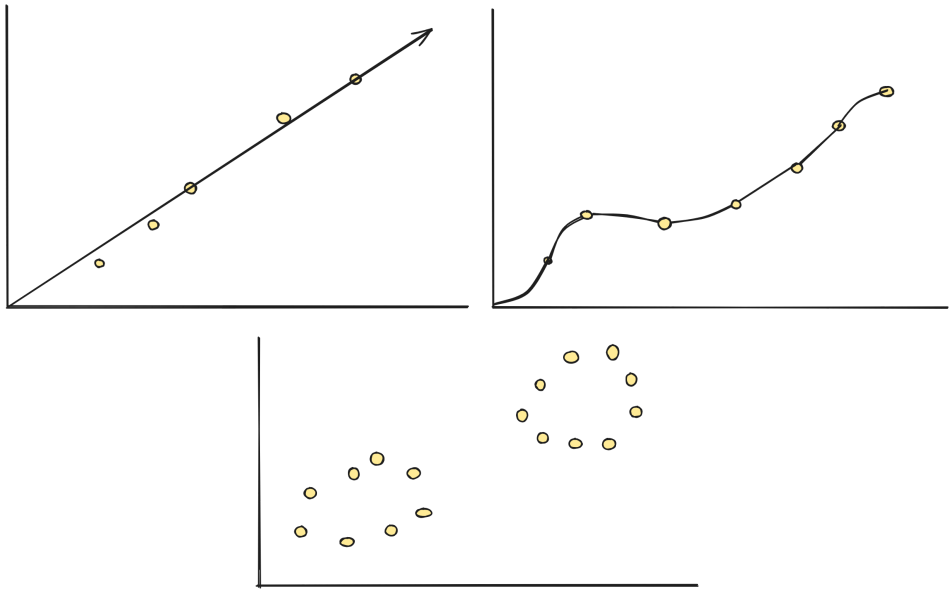
Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750

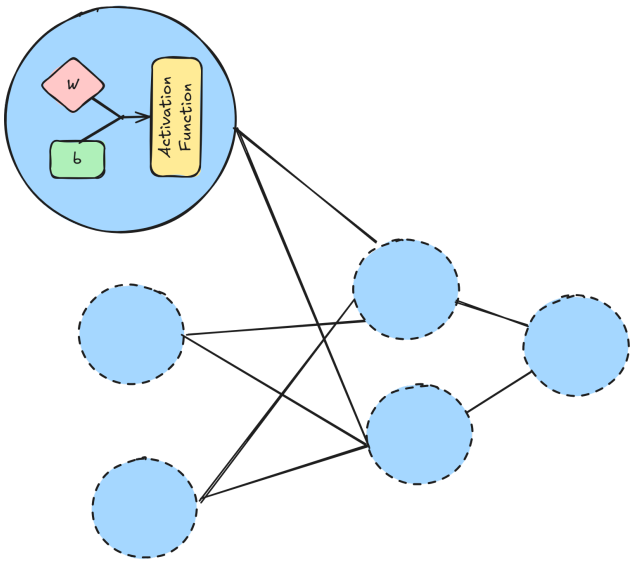
How do Activation Functions affect Neural Networks?

The activation function is the key difference between a linear model, with its weights and biases, and neural networks, which are interconnected nodes functioning as universal function approximators. An activation function is a mathematical function applied to the output of a neuron, determining whether it should be activated or not based on its input signals. This function plays a crucial role in enabling neural networks to learn complex patterns by introducing non-linearity into the model.



For the last dataset, to be able to use a regression model, it would require an appropriate transformation. Determining which transformation to apply to the dataset would require extensive domain knowledge, and even then, it would involve much trial and error and likely lead to extensive overfitting.

Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750



In a simple neural network, as illustrated, each neuron's activation function will take the products of the input signal, weight, and biases, and determine if they are "significant." If so, it will pass them on to the next layer.

Types of Activation Functions

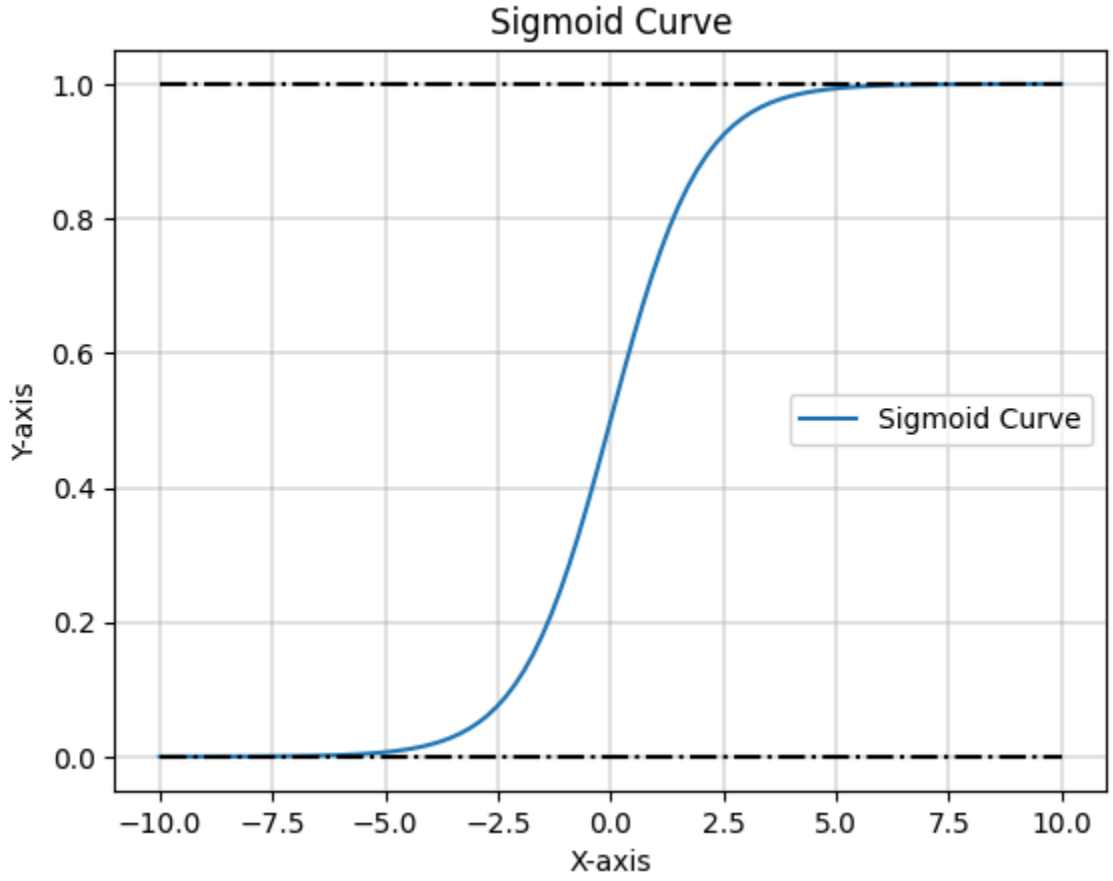
Technically, there are neurons that could be simple linear units or use a binary threshold function. However, these simple functions are limited to specific scenarios. In most deep learning applications, we rely on more generalized activation functions.

1. Sigmoid
   The sigmoid activation function (or logistic function), maps any real number to a value between 0 and 1. Larger inputs yield outputs closer to 1, while smaller inputs result in outputs closer to 0.

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

	General Uses	Limitations
⇥	Binary Classification: Sigmoid is often used in binary classification output layers, mapping outputs to probabilities between 0 and 1.	Vanishing Gradient Problem: For large positive or negative input values, the gradient approaches zero, which can slow down or halt learning during backpropagation, especially in deep networks.
⇥	Smooth Output: Sigmoid's smooth nature enables gradual output changes, useful for	Non-Zero-Centered: Positive outputs (0 to 1) can lead to inefficient weight updates and slower convergence.

	problems requiring soft decision boundaries.	
⇥	Hidden Layer in Historical Models: The sigmoid function was historically used in the hidden layers of neural networks to introduce non-linearity.	Computational Inefficiency: Exponential calculations make sigmoid less efficient than ReLU.



Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750

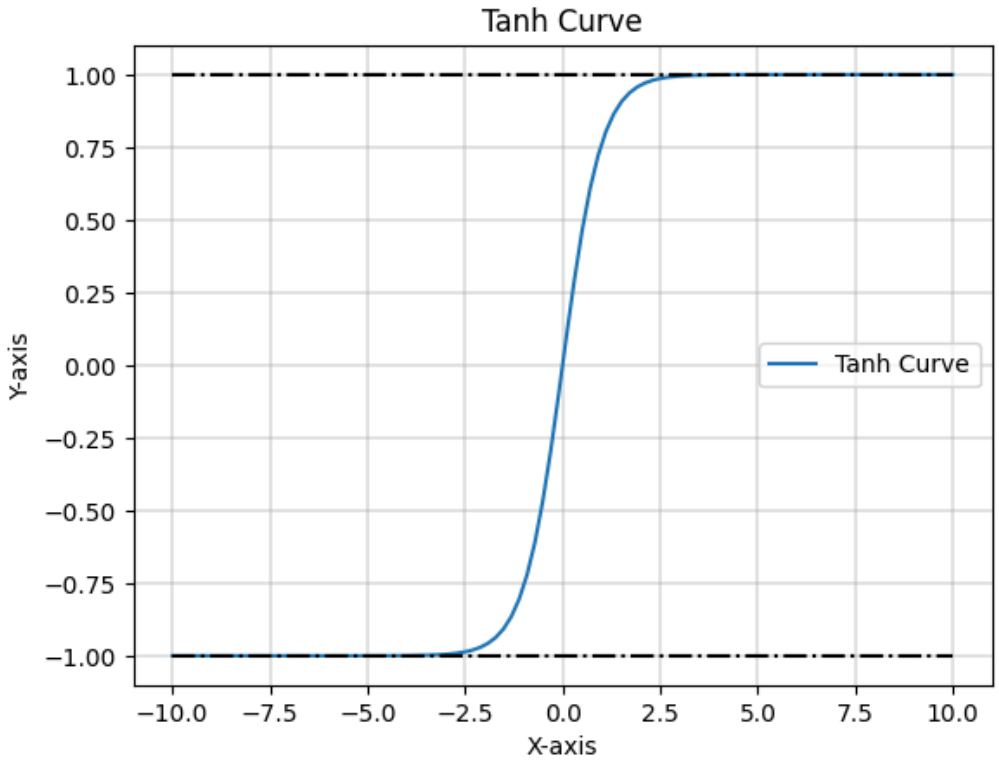
2. Tanh

Tanh (or the hyperbolic tangent activation function), similar to sigmoid, maps real inputs to the range -1 to 1. Larger inputs yield values closer to 1, while smaller inputs yield values closer to -1.

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

	General Uses	Limitations
⇥	Zero-Centered Output: The -1 to 1 range enables zero-centered gradients, aiding faster convergence.	Vanishing Gradients: Similar to sigmoid, tanh can suffer from vanishing gradients for extreme inputs, slowing deep network learning.
⇥	Hidden Layers: Tanh is used in hidden layers, especially in RNNs, for effective handling of negative inputs and outputs.	Saturation at Extremes: For extreme inputs, the function saturates, leading to near-zero gradients and slow learning.
⇥	-	Computational Complexity: Tanh's exponential calculations can be computationally expensive.

Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750



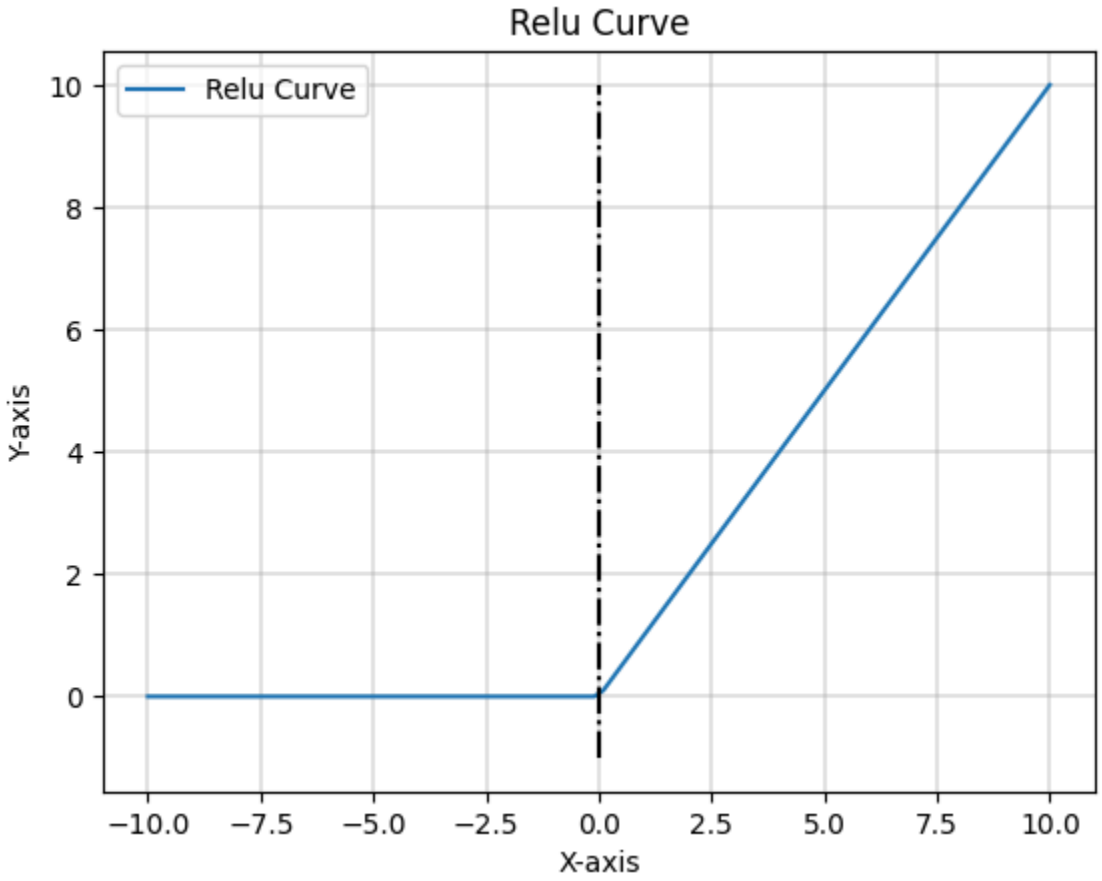
3. Rectified Linear Unity (ReLU)

ReLU, a common hidden layer activation function, is simple and effective, overcoming Sigmoid and Tanh limitations.

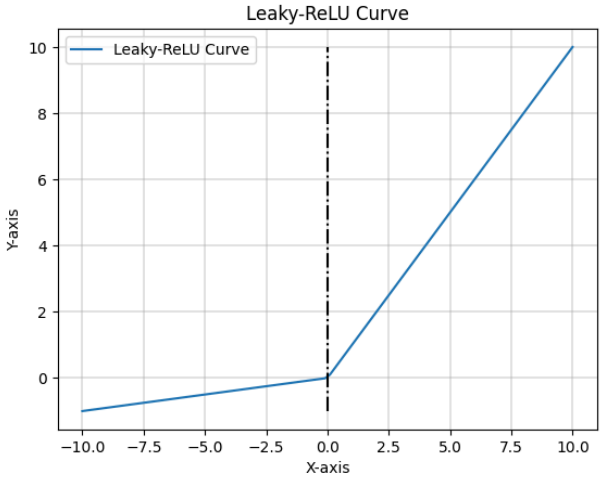
$$f(x) = max(0, x)$$

	General Uses	Limitations
⇥	Computational Efficiency: ReLU's simple thresholding operation leads to faster training.	Dead Neurons: Neurons can become inactive (outputting zero) for all inputs, reducing model capacity.
⇥	Vanishing Gradient Mitigation: ReLU's non-saturating nature allows for better gradient flow in deep networks.	Unbounded Output: ReLU's unbounded output can lead to exploding gradients if not managed.
⇥	Sparse Activation: ReLU's zeroing out of negative values introduces sparsity, potentially leading to more efficient representations and reduced overfitting.	Non-Differentiability at Zero: ReLU's non-differentiability at zero can pose challenges in optimization, though this is often mitigated.

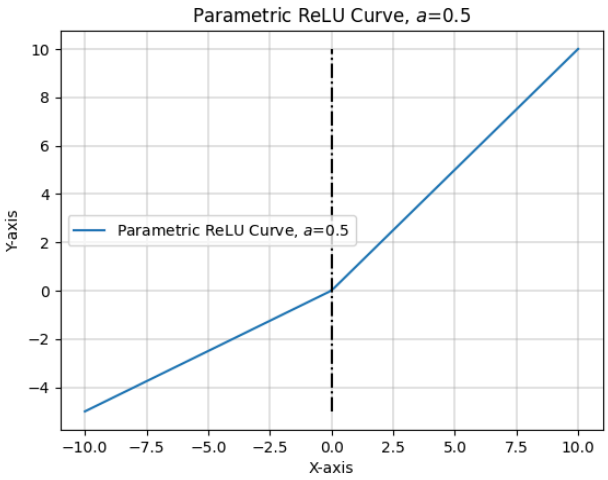
Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750



Variants of ReLU exist to address the dead-neurons problem:

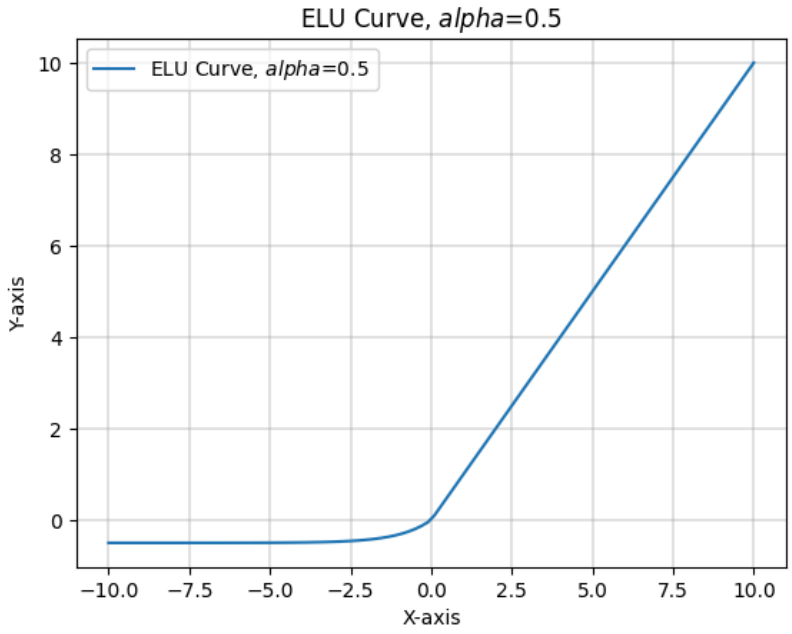


a. Leaky ReLU



b.Parametric ReLU

Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750



c. Exponential Linear Units

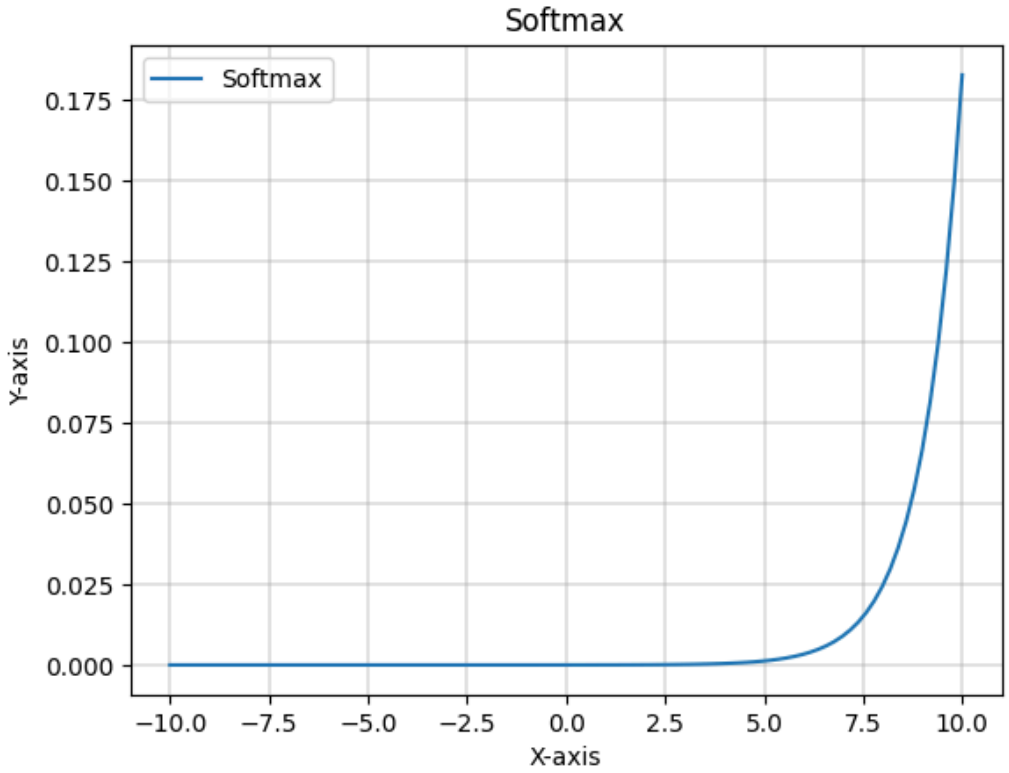
4. Softmax

The softmax function gives a set of probabilities for different classes that all add up to 1. The class with the highest probability is the predicted class for that particular input.

$$f(x) = \frac{e^x}{\Sigma\, e^x}$$

	General Uses	Limitations
⇥	Multi-Class Classification: Softmax, used in output layers, converts logits to probabilities for multi-class classification.	Sensitivity to Outliers: Softmax can be heavily influenced by large disparities in logits, leading to overconfidence in predictions and making the model sensitive to outliers and noisy data.
⇥	Probability Distribution: It transforms outputs into a probability distribution, essential for tasks like image classification and NLP.	Computational Complexity: Softmax's exponentiation and normalization can be computationally expensive.
⇥	Differentiability: Softmax is differentiable, allowing for gradient-based optimization and effective weight updates.	Not Suitable for Multi-Label Tasks: Softmax assumes mutually exclusive classes, making it unsuitable for multi-label tasks.

Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750



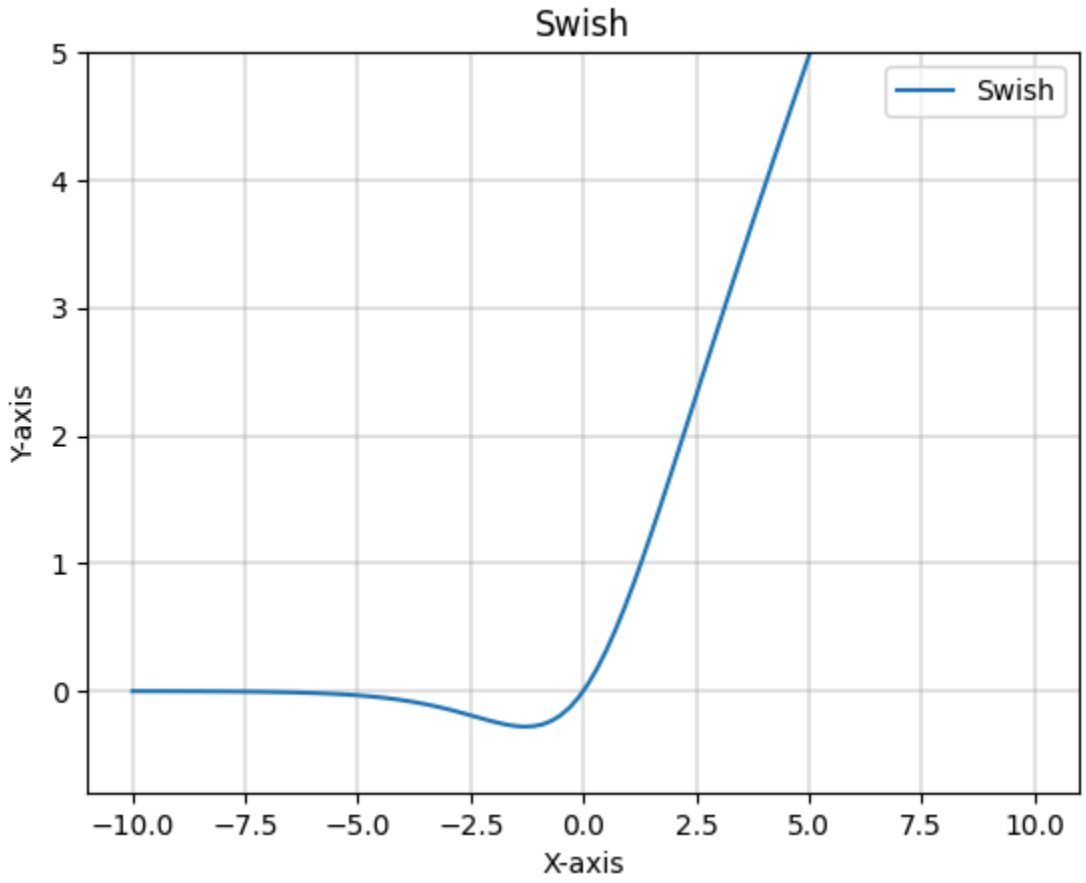
5. Swish

Swish, a recent innovation, is a ReLU replacement for large, deep networks on tasks like image classification and NLP.

$$f(x) = x * sigmoid(x)$$

	General Uses	Limitations
⇥	Improved Performance: Swish often outperforms ReLU in deep networks, especially in image classification and language modeling.	Computational Intensity: Swish's multiplication operation can be computationally expensive.
⇥	Better Handling of Negative Inputs: Unlike ReLU, Swish allows for small negative values, potentially capturing more information and improving learning.	Saturation for Large Inputs: Swish can saturate for very large inputs, leading to reduced gradient flow and slower learning.

Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750

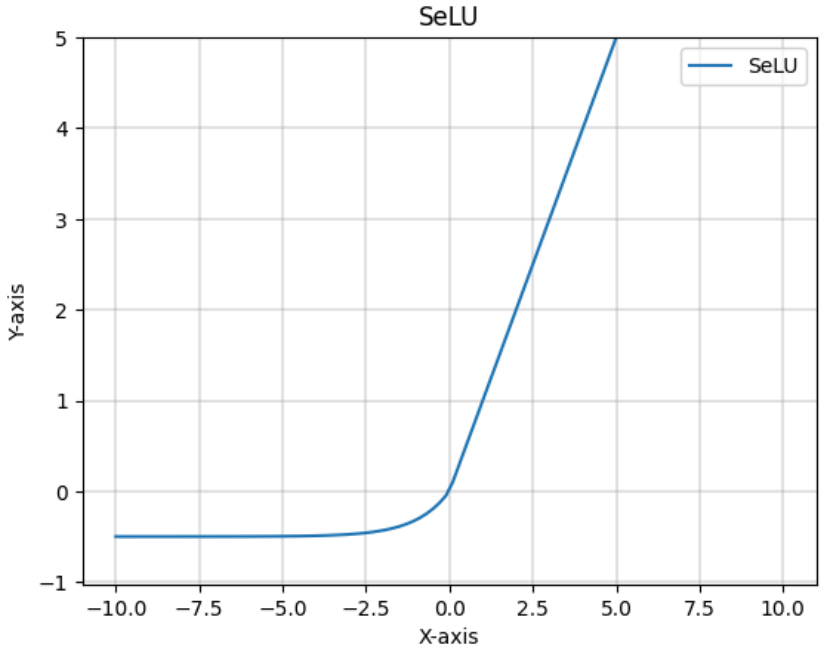


6. Scaled Exponential Linear Unit (SELU)

SELU is a self-normalizing activation function, ensuring stable mean and variance. This can lead to faster training and improved performance in deep networks.

$$f(x) = \lambda\alpha(e^{x} - 1) \text{ if } x < 0$$
$$f(x) = \lambda x \quad \text{if } x \geq 0$$

Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750



Our Experiment

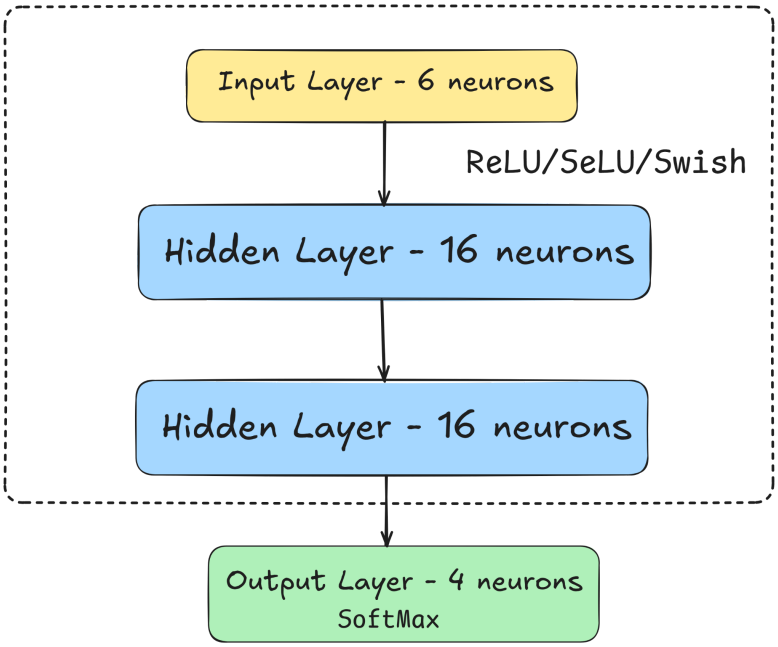
Choosing the right activation function is crucial. It affects training speed, accuracy, and the model's ability to avoid local minima. Some functions, like ReLU, are computationally efficient and help gradient flow, while others, like Sigmoid and Tanh, can suffer from vanishing gradients. The choice also depends on the specific task. For instance, Softmax is ideal for multi-class classification, while ReLU and its variants are well-suited for deep networks. We'll set up a simple experiment to illustrate these ideas.

We will attempt to train models using the different activation functions on the same dataset, and benchmark their performances.

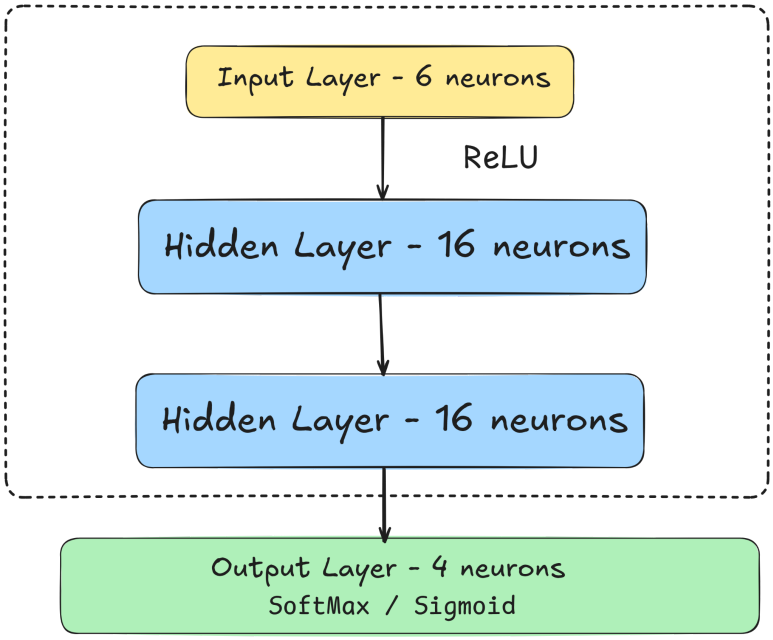
Our dataset: The UCI Car Evaluation Dataset is derived from a hierarchical decision model used to evaluate cars based on various attributes such as buying price, maintenance cost, number of doors, passenger capacity, luggage boot size, and safety. This dataset consists of 1,728 instances and is primarily utilized for classification tasks, where the goal is to predict car acceptability categorized into four classes: unacceptable, acceptable, good, and very good. All features are categorical, and we'll need to apply relevant encoding. The categories are ordinal, so label encoding would be the better choice over one hot encoding.

For comparing the hidden layer functions, we have the architecture:

Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750



And to compare the two output layer functions, Sigmoid and Softmax,

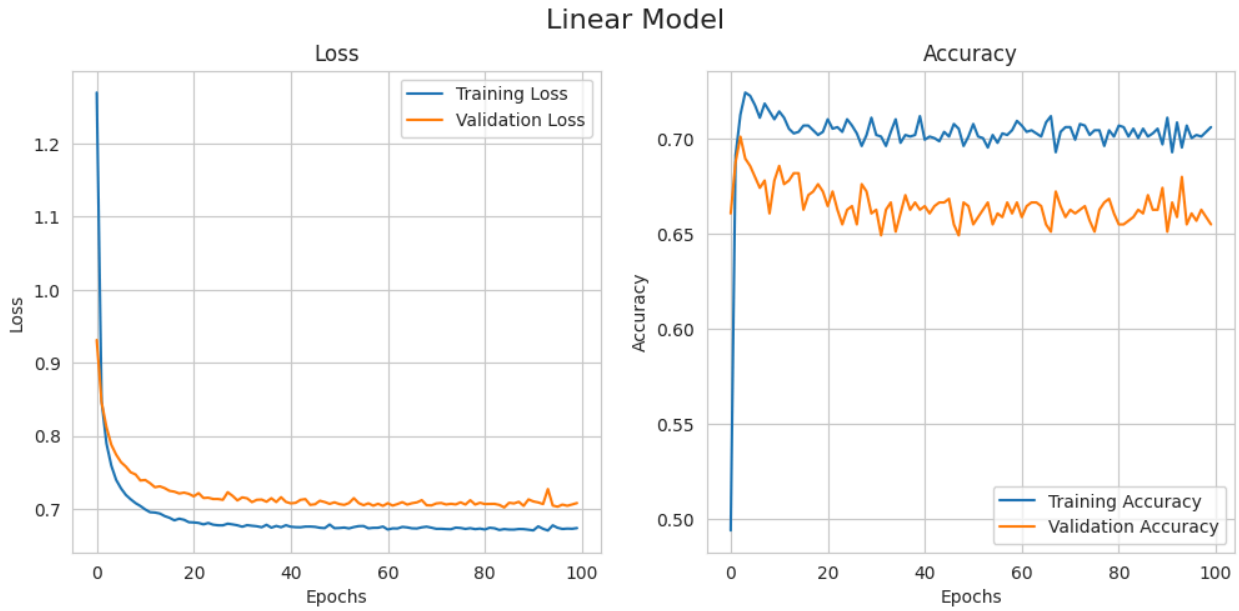


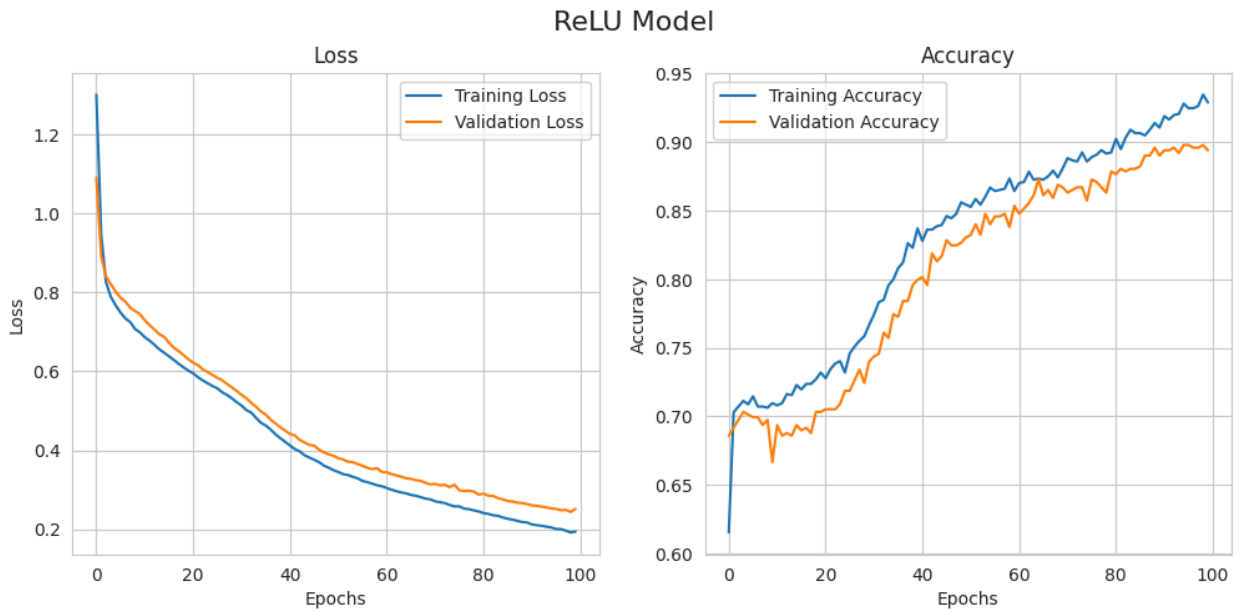
This is a straightforward machine learning task. While some activation functions might excel with larger datasets or specialized domains like image classification, this experiment will help us visualize their key characteristics. Before training and testing, we'll preprocess the data by label encoding and splitting it into training and test sets. Each model will be trained for 100 epochs using the Adam optimizer and categorical cross-entropy loss, with a 70/30 train-test split.

Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750

Hidden Layer Experiments

Learning Rate

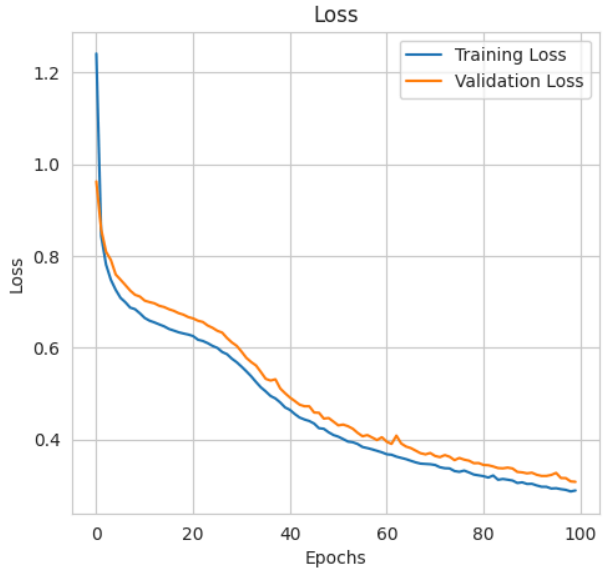


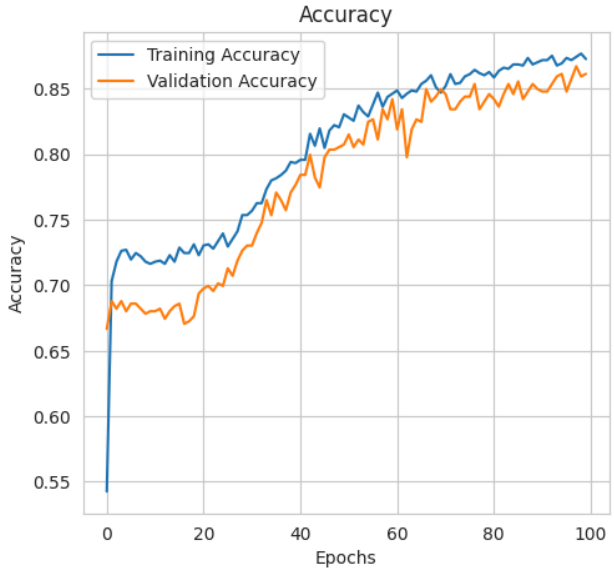


Name: Kosisochukwu Emeka-Obinabo

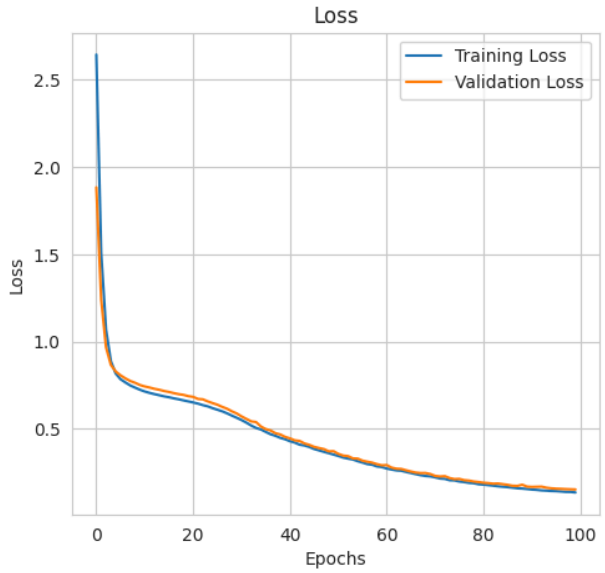
Student ID: 23035750

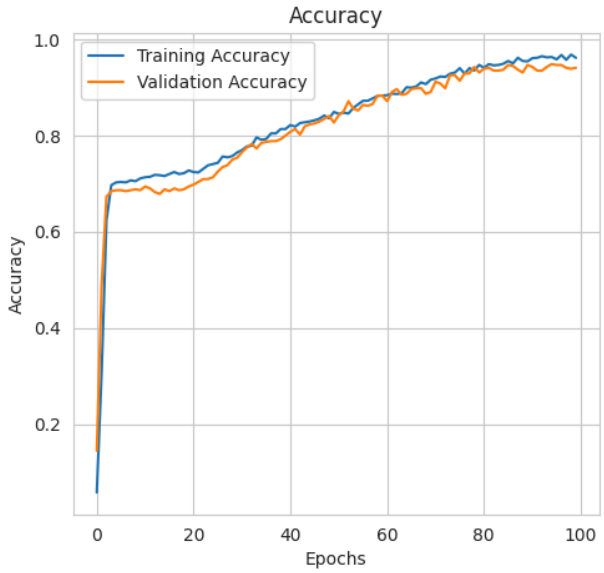
SeLU Model





Swish Model



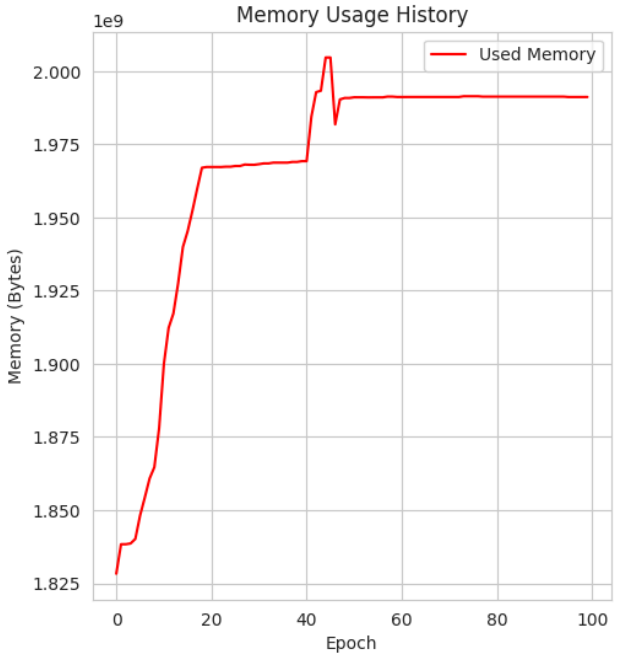


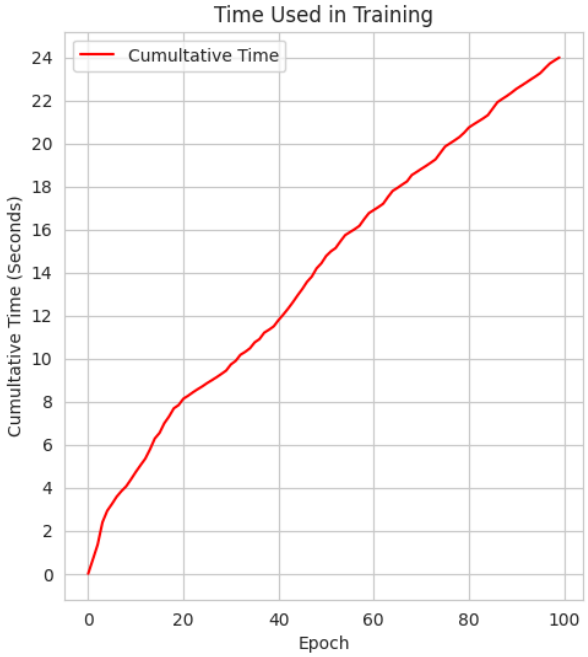
Name: Kosisochukwu Emeka-Obinabo

Student ID: 23035750

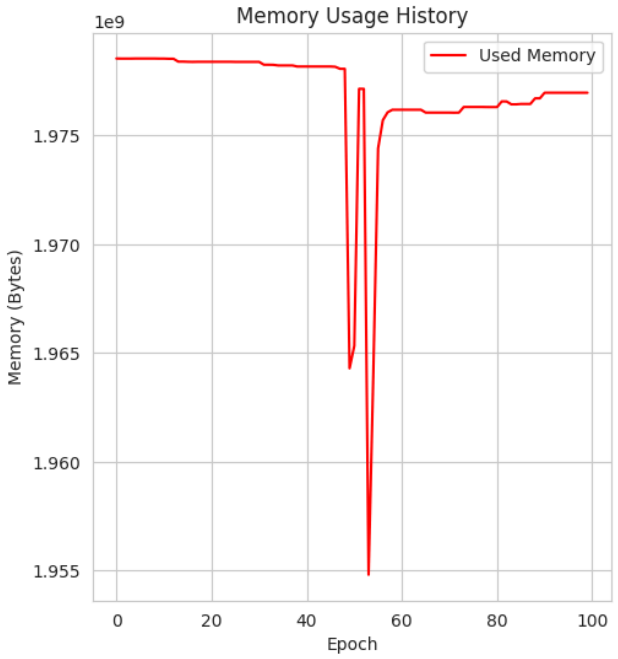
Time & Memory Footprint

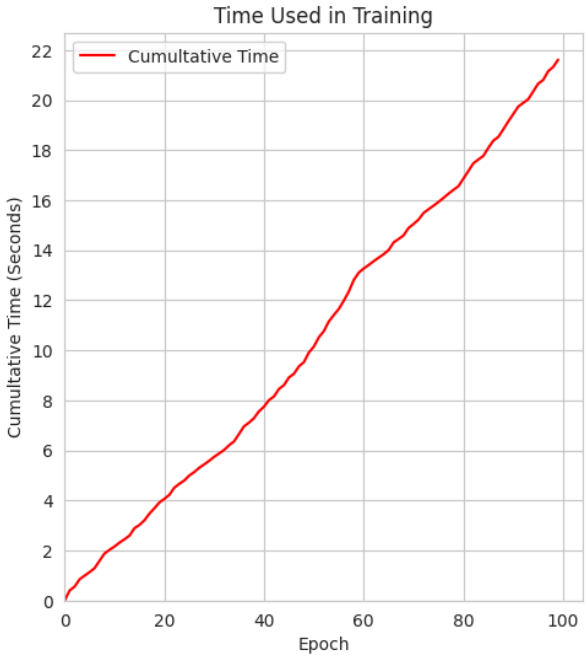
Linear - Compute Time & Memory





ReLU - Compute Time & Memory

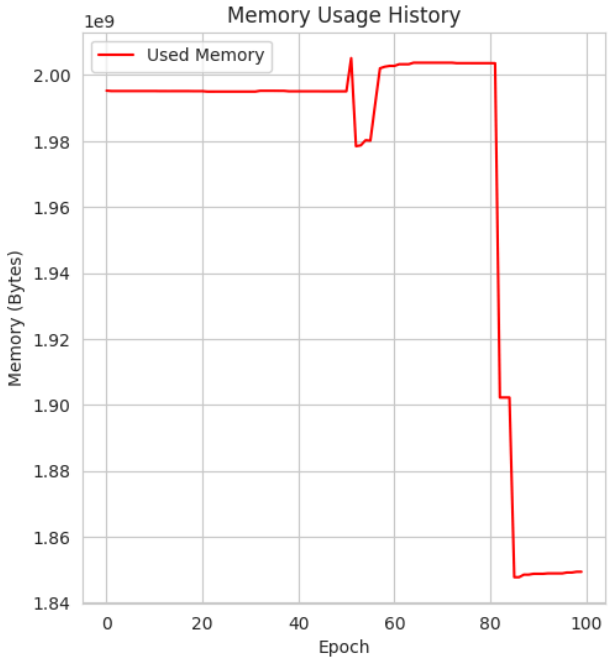


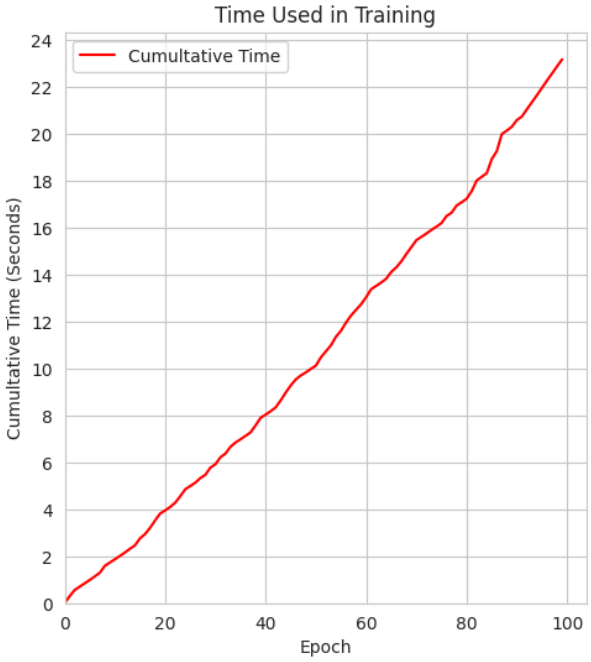


Name: Kosisochukwu Emeka-Obinabo

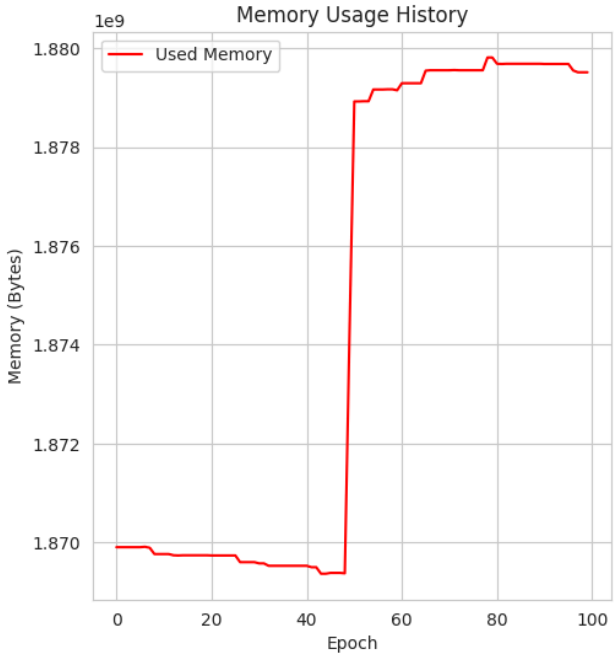
Student ID: 23035750

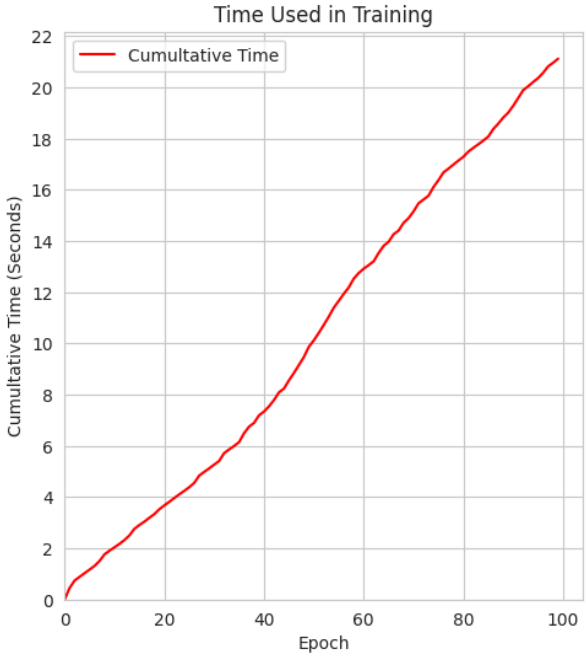
SeLU - Compute Time & Memory





Swish - Compute Time & Memory





Name: Kosisochukwu Emeka-Obinabo

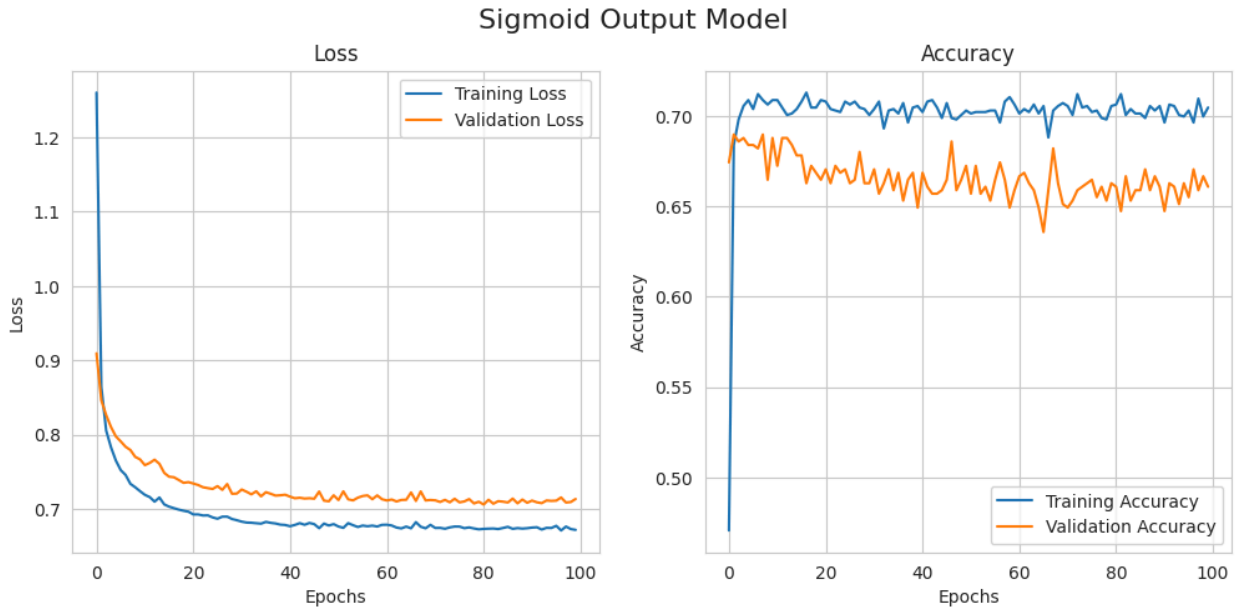
Student ID: 23035750

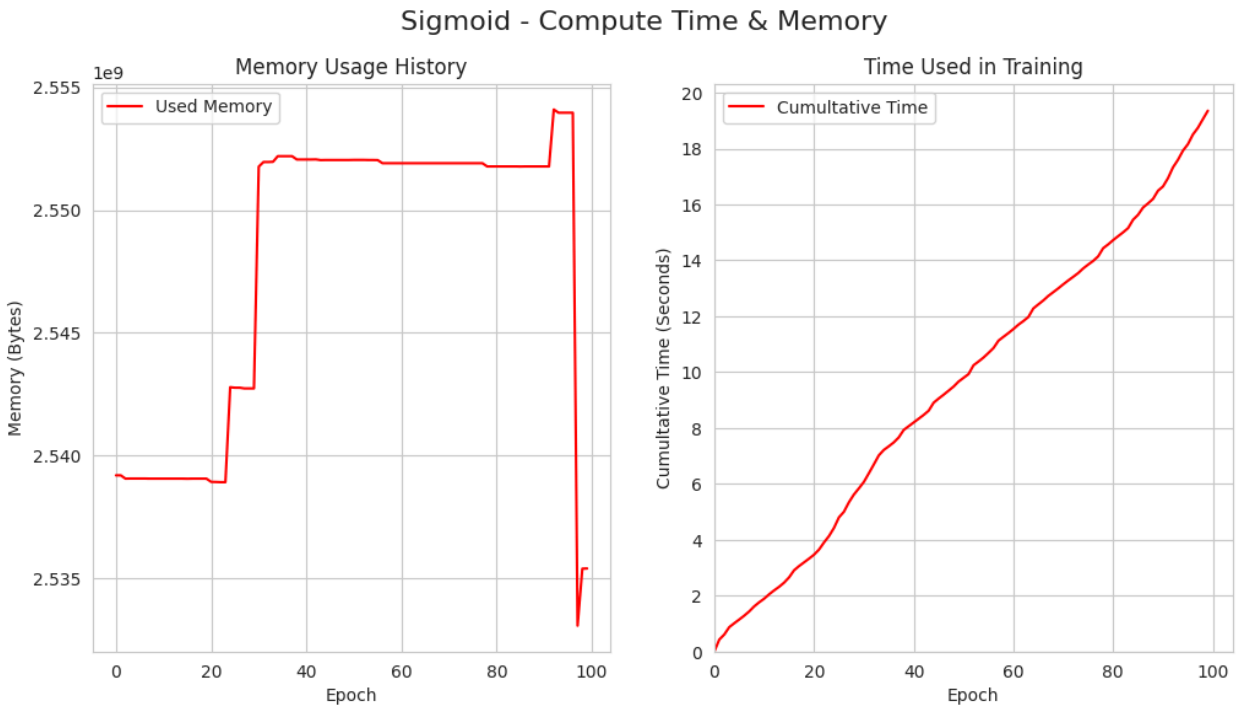
And here's a summary of the results:

Activation Function	Accuracy	Speed of Convergence	Training Time & Memory Usage
Linear	Low accuracy. Could not achieve 90% accuracy	Since there is no non-linearity, it converges almost immediately	It takes the longest of all architectures to complete its epochs, however, it has the smallest memory footprint
ReLU	Achieved the second best train and validation accuracies.	Very fast convergence	Has the second-best memory efficiency and the best training time in this experiment.
SeLU	Comparatively Moderate accuracy of 85%; self-normalizing properties help maintain performance.	Very fast convergence	Moderately high time consumption and memory footprint, due to normalization requirements.
Swish	Best accuracy results; smooth function helps in learning complex patterns.	Competitively fast convergence.	*Moderate time consumption and is more computationally intensive than ReLU due to additional operations..*

Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750

Output Layer Experiments





The benchmark Softmax + Linear architecture remains the control. Comparing it to the Sigmoid architecture, there's no significant immediate difference. However, Softmax achieved a higher validation accuracy. The choice between Softmax and Sigmoid depends on the specific problem: Softmax is ideal for multi-class classification, while Sigmoid is better suited for multi-label scenarios.

Name: Kosisochukwu Emeka-Obinabo
Student ID: 23035750

Caveats and Considerations

These tests were conducted on a Google Colab Standard Notebook. While tracking memory usage, it's important to remember that these measurements reflect the entire VM's consumption, not just the Keras process. Therefore, direct comparisons are most meaningful when made on the same machine under similar conditions. Additionally, due to the sequential nature of the tests and the potential for other processes to use memory on a shared VM, these measurements should be interpreted with caution.
Regarding convergence speed, this experiment used a relatively small dataset and batch size. Larger datasets could provide more opportunities to explore the impact of different activation functions better.

Conclusion

The choice of activation function plays a critical role in the performance of neural networks. While this tutorial doesn't provide an exhaustive list, there are many more activation functions used in niche applications and domains (you can find more in the Keras activation functions documentation).

We've clearly illustrated the importance of activation functions in neural networks, regardless of the specific kernel used. Overall, selecting the appropriate activation function is essential for optimizing model performance and achieving reliable results in machine learning applications.

References

- Diagrams were created using Matplotlib and Excalidraw (https://excalidraw.com).
- Bohanec, M. (1988). Car evaluation [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5JP48
- Maind, S. B., & Wankar, P. W. (2014). Research paper on basics of artificial neural network. International Journal of Research in Information Technology and Communication Configuration, 1(1).
- Baheti, P. (2021, May 27). Neural networks activation functions. V7 Labs. https://www.v7labs.com/blog/neural-networks-activation-functions
- Datacamp. (n.d.). Introduction to activation functions in neural networks. https://www.datacamp.com/tutorial/introduction-to-activation-functions-in-neural-networks
- GeeksforGeeks. (n.d.). Linear regression vs neural networks: Understanding key differences. https://www.geeksforgeeks.org/linear-regression-vs-neural-networks-understanding-key-differences/
- Machine Learning Mastery. (n.d.). Choose an activation function for deep learning. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/
- ScienceDirect. (n.d.). Sigmoid function. Retrieved from https://www.sciencedirect.com/topics/computer-science/sigmoid-function
- Arora, S., & Zhang, Y. (2017). Silu/Swish: A self-gated activation function. arXiv preprint arXiv:1710.05941v1. https://arxiv.org/abs/1710.05941
- Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. arXiv preprint arXiv:1706.02515. https://arxiv.org/abs/1706.02515