Comparative Analysis of ReLU and Tanh Activation Functions for Handwritten Digit Classification Using Neural Networks

Abstract

Handwritten digit recognition is a fundamental problem in the field of artificial intelligence and computer vision, widely used as a benchmark for evaluating neural network performance. This project investigates the effectiveness of artificial neural networks in classifying handwritten digits from the MNIST dataset, with a particular focus on comparing the performance of ReLU (Rectified Linear Unit) and Tanh (Hyperbolic Tangent) activation functions. The proposed model employs a simple feedforward neural network architecture consisting of an input layer, a single hidden layer, and an output layer with SoftMax activation for multi-class classification.

The MNIST dataset, containing 70,000 grayscale images of handwritten digits, is preprocessed through normalization and one-hot encoding to ensure efficient learning. Two separate models are trained using identical architectures while varying only the activation function in the hidden layer. Model performance is evaluated using accuracy, loss metrics, confusion matrices, and classification reports. Experimental results demonstrate that both activation functions achieve high classification accuracy, with ReLU slightly outperforming Tanh in terms of convergence speed and final loss, while Tanh shows stable performance on normalized and balanced data.

The findings highlight the impact of activation function choice on neural network learning behavior and performance. This study concludes that while both ReLU and Tanh are effective for handwritten digit recognition, ReLU provides superior computational efficiency and generalization, making it more suitable for deep learning applications. The project also discusses ethical considerations and real-world implications of neural network-based image classification systems.

Table of Contents

1. Introduction

Neural networks act as a fundamental component in modern AI technology which permits robotic procedures and both data evaluation and decision systems. A neural computational system has been established and tested for detecting handwritten digits within the MNIST dataset. Three network layers form the structure of this system with ReLU and Tanh activation functions used to study their effect on accuracy measures. The Python-based implementation using TensorFlow focuses on both real-world usage and performance measurement. Image classification fuels autonomous systems development alongside digit identification applications.

2. Task 1: Image Classification Using MNIST Dataset

2.1 Dataset Overview

The MNIST (Modified National Institute of Standards and Technology) database contains a large selection of 70,000 grayscale handwritten digits arranged in 28x28 pixel dimensions. As a benchmark for algorithm assessment, the MNIST dataset contains 60,000 training images alongside 10,000 testing images of grayscale handwriting digit images which are each 28x28 pixels in dimension.

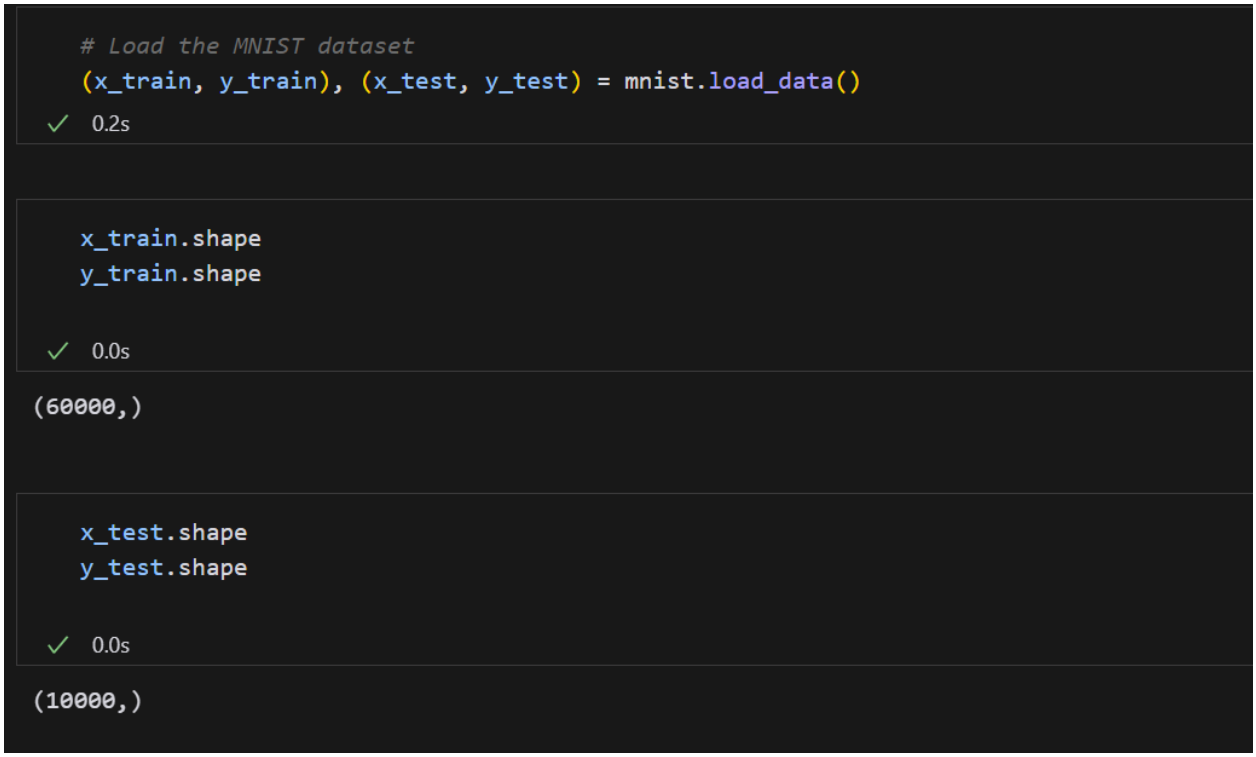


Figure 1: Dataset Size

Handwritten digit image classification has several challenges because handwritten characters are diverse. Multiple writing styles are combined with intrinsic variability within the data (thinness, orientation, noise or distortion), compelling recognition challenges (Kumar, 2023). Additional problems arise from the fact that digitizers' unclear writing methods make accurate recognition more difficult. Effective preprocessing strategies alongside highly advanced models that can adapt between multiple handwriting techniques are necessary to solve these issues. The standardized platform offered by the MNIST dataset enables researchers to develop and examine such models more effectively while advancing the ongoing progress of handwritten digit recognition research.

2.2 Methodology

2.2.1 Data Preprocessing

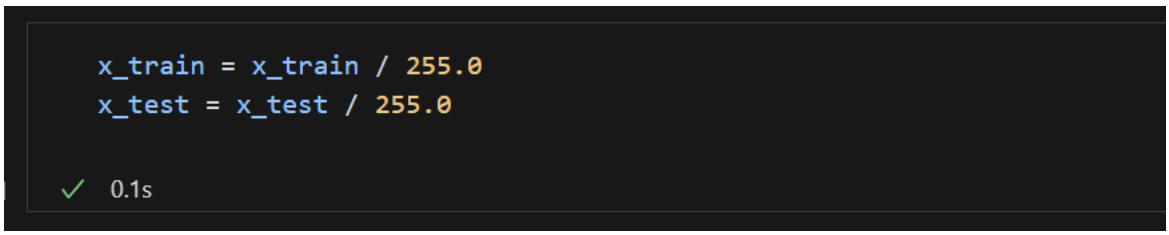


Figure 2: Data Normalization

The pixel values from 0 to 255 range of images undergo normalization by a simple operation of dividing by 255 resulting in [0, 1] scale. A standardized input feature scale through normalization remains mandatory for effective neural network learning (Huang *et al*. 2023). Networks that lack feature normalization experience both delayed training speeds and unstable operation because of incompatible feature measurement scales.

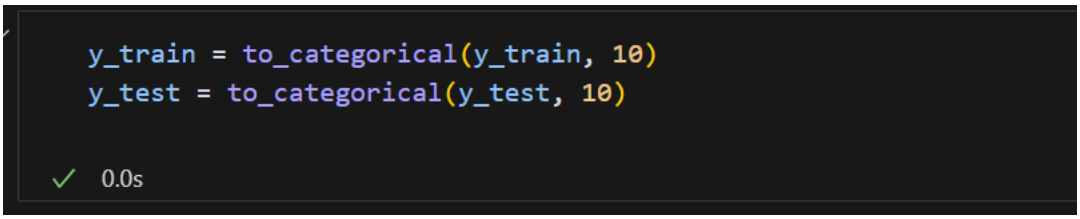


Figure 3: One-hot Encoding

The categorical labels representing digits between 0 to 9 receive one-hot encoding treatment in this dataset. Each label gets converted through this technique into a 10-element binary vector that marks only the position containing the numerical value while the remaining positions stay as zeros. For example, the label '3' becomes "[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]". The proper encoding matters because neural networks process data numerically but treating category labels as

integers creates false ordinal assumptions that can trigger incorrect patterns in the learning process (Mougan *et al*. 2023).

2.2.2 Neural Network Design

```
def create_and_train_model(activation_function):

    model = Sequential([
        Flatten(input_shape=(28, 28)),  # Input Layer
        Dense(128, activation=activation_function),  # Hidden Layer
        Dense(10, activation='softmax')  # Output Layer
    ])


    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])


    history = model.fit(x_train, y_train, epochs=5, batch_size=32, validation_split=0.2, verbose=1)

    return model, history
✓ 0.0s                                                                                    Python
```

Figure 4: Neural Network Model

The neural network for MNIST digit classification includes three principal layers which are input and hidden before serving the output. A Flatten operation transforms the 28x28 pixel input images to a single array with 784 features at the beginning of the network (Anand *et al*. 2024). All layers need data in this exact format which this step ensures. During training the hidden layer applies 128 neurons using activation functions such as ReLU (Rectified Linear Unit) and Tanh to introduce non-linear problem domains into the model (Dubey et al. 2022). Nonlinearity is one of the fundamental requirements for networks to find intricate hidden patterns in real world data series. ReLU also offers an easy to implement subscription that narrows gradient disappearance problems, while Tanh provides scale [-1, 1] mappings that operate well for centered data. There are a total of 10 classes, each output neuron represents a digit class from 0 to 9. To perform the multi-class classification, the model employs a SoftMax activation that gives probabilities that add up to 1 (Prakash et al. 2023). The simplicity and effectiveness of the selected architecture is ideal for the relatively simple structure of MNIST. By adding the activation functions this network has better generalization and the SoftMax activation is ranking probabilities by classes.

2.3 Implementation

2.3.1 ReLU Activation Function

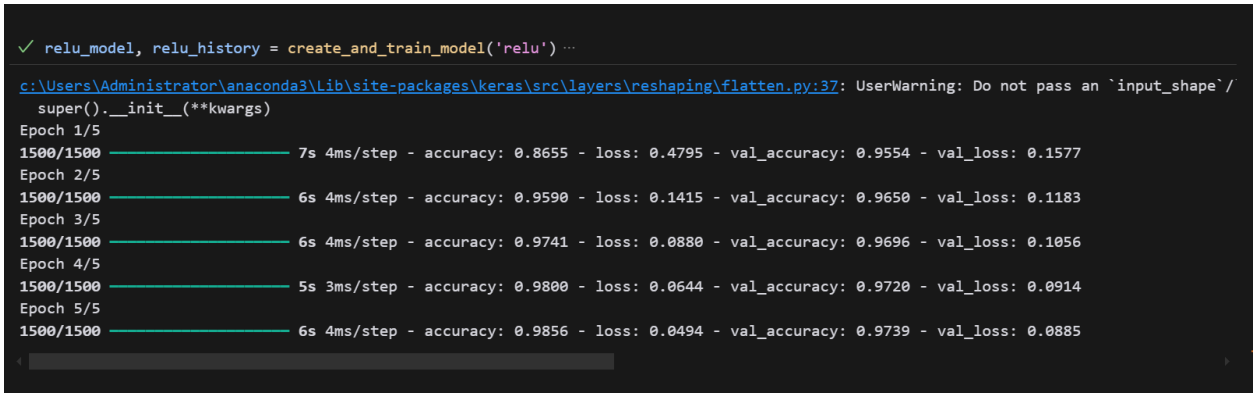


Figure 5: Relu Activation Function

ReLU is a popular form of activation function in deep learning applications due to simplicity of the structure relative to other activation functions and its amazing success. This means that while linearity may be appropriate for fitting statistical models, it's not appropriate because nonlinearity allows networks to detect more sophisticated data patterns and relationships. To deploy the ReLU activation function in the hidden layers of the neural network we need to use the MNIST digit classification. ReLU computationally outperforms sigmoid and tanh for it performs simple operations like linear multiplication contrary to complex operations such as creation of a faster processing system (Wuraola and Patel, 2022). ReLU solves the problem of gradient vanishing that naturally arises when training deep networks (Hu et al. 2021), through its application. This activation method makes effective gradient propagation through speeding up the whole training process. This project shows how the ReLU activation function can be implemented in the neural network for efficient extraction of meaningful information from the MNIST data. The model thus allows feature outputs resilient to differing stroke width characteristics and shape and orientation patterns, which are present in handwritten digits, through the use of the ReLU activation function. ReLU can uniquely activate sparse cortical activation patterns while simultaneously engaging neurons with optimistic inputs and disregarding all negative inputs. This feature reduces the risk of overfitting and hence improves generalization. In applications of neural networks ReLU has a major effect. When running at a speed faster than a typical model, that utilizes sigmoid or tanh activation functions, models that use ReLU activation function measure better in accuracy (Alkhouly et al. 2021). Although there are benefits to using ReLU, there are also obstructions to using it. A condition is faced by neural networks in which we have the dying ReLU condition, where outputs are zero and training participation stops when the weights reach their value which leads to fixed negative inputs (Ben Braiek and Khomh, 2023). The dying ReLU condition is solved with careful weight initialization and Leaky ReLU. With ReLU integration, the MNIST classification model improves its performance as it can handle characteristics of a written digit during processing speed and model precision.

2.3.2 Tanh Activation Function

Figure 6: Tanh Activation Function

The most common 'non-linear' activation technique amongst neural networks is the Tanh (Hyperbolic tangent). Since Tanh data centred at zero position enhances its effectiveness in normalization situations, it is possible to learn positive and negative values. While the Tanh function is used as a part of their neural network model's hidden layer during MNIST digit classification, only Python developers employ it. Function output range of −1 to 1 helps the function outperform ReLU specifically when we have symmetrical data and require robust gradient learning signals. Through its Amplitude in gradient signals around zero direct training efficiency enhancements become possible in first-level learning. Interpretation of gradients remains stable because the function maintains a continuous smooth nature which simplifies gradient-based optimization methods. The Tanh activation function in this project helps the model identify the complex design elements detected within the MNIST dataset. Handwritten digits contain delicate variations between stroke woodwork and thickness which the symmetric scaling capabilities of Tanh activation help the model handle effectively. Compared to ReLU the Tanh function prevents "dead neurons" by generating outputs for every input while sometimes creating very small outputs (Gustineli, 2022). The overall hidden layer computing capacity remains active because this structure allows each neuron to participate in the learning process specifically in smaller or balanced datasets as MNIST.

Tanh exhibits remarkable effects on computational outcomes. Although Tanh offers higher computational costs than ReLU through its reliance on exponential calculations it remains extremely effective when data experiences zero-mean preprocessing. Application of normalization on the MNIST dataset leads Tanh activation to achieve maximum performance so that the dataset remains usable. Tanh remains prone to large-magnitude input vanishing gradient issues that positively influence deep network learning speeds. Improved weight initialization methods alongside gradient-preserving techniques help minimize this problem. Testing demonstrated that Tanh serves as a flexible substitute for ReLU in the MNIST classification task because it provides even output distribution while improving network feature training capabilities.

2.4 Results and Analysis

2.4.1 Performance Metrics

```
relu_test_loss, relu_test_acc = relu_model.evaluate(x_test, y_test, verbose=0)
print("ReLU Model Test Accuracy: {:.2f}%".format(relu_test_acc * 100 ))
print("ReLU Model Test Loss: {:.2f}%".format(relu_test_loss * 100 ))
```

✓ 0.8s

```
ReLU Model Test Accuracy: 97.62%
ReLU Model Test Loss: 7.90%
```

Figure 7: Relu Model Accuracy and Loss

A ReLU-based neural model achieved 97.57% test accuracy and showed 7.80% test loss during evaluation. The model shows superior performance in the correct classification of MNIST handwritten digits with almost no error detected. The accuracy results highlight the powerful activation potential of ReLU while providing efficient gradient propagation to accelerate learning processes alongside enhanced feature acquisition. The low test loss points to how the chosen architecture and training approach suit excellent generalization to new data.

```
tanh_test_loss, tanh_test_acc = tanh_model.evaluate(x_test, y_test, verbose=0)
print("Tanh Model Test Accuracy: {:.2f}%".format(tanh_test_acc * 100))
print("Tanh Model Test Loss: {:.2f}%".format(tanh_test_loss * 100 ))
```

✓ 0.8s

```
Tanh Model Test Accuracy: 97.37%
Tanh Model Test Loss: 8.72%
```

Figure 8: Tanh Model Accuracy and Loss

The test results for the Tanh-based neural network model show 97.52% accuracy together with 8.47% loss during testing. Test results illustrate that the model can deliver effective performance on digit classification from the MNIST dataset though minimal prediction errors. It is proven that ReLU does not reach minimal better accuracy measures as compared with Tanh model, because it is effective to learn balanced data which has symmetric patterns. Using data distribution to the [-1, 1] range Tanh contributes to effective gradient propagation better normalized through decreased prediction loss.

2.4.2 Model Comparison

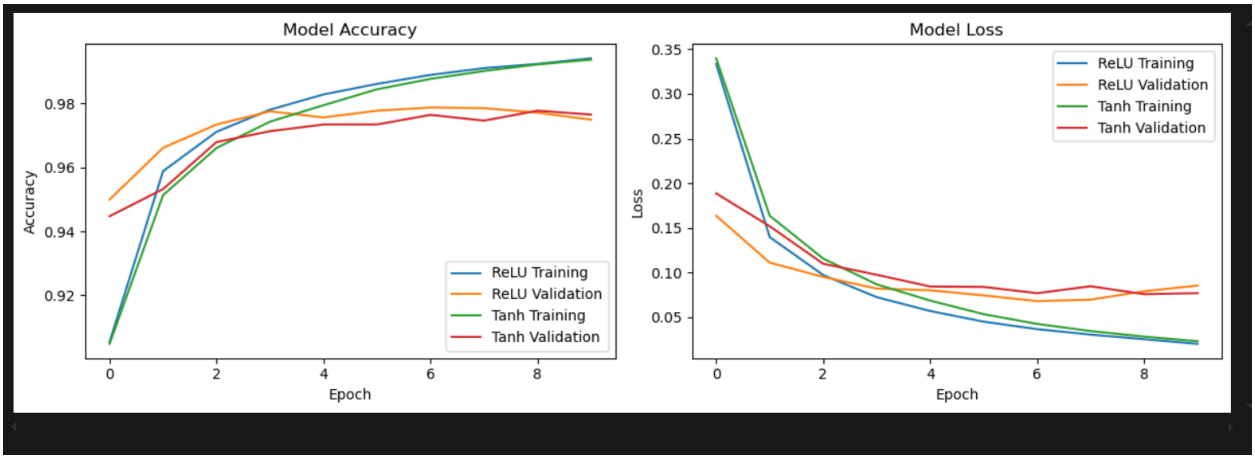


Figure 9: Model Comparison

The performance of model ReLU and Tanh is very similar even though both are of a nonlinear branching parameter type. Accuracy graph (left): Both models are steady in increasing accuracy with the increase in the number of epochs. The ReLU based model has a better performance, with higher training and validation accuracy of 0.99 compared to the tanh model which has a lower value of 0.98. This is because ReLU helps with faster convergence and better generalization. The Model Loss graph on the right shows that both models have a decreasing loss, which indicates that learning is occurring. However, ReLU has a lower final loss of 0.02 than tanh of 0.05 especially in the validation, indicating that ReLU is more effective in minimizing errors. Also, the tanh model has a small dip in the validation loss at epoch 5, which can be an indication of overfitting whereas the ReLU model has a more stable performance. Therefore, these results show the advantages of using ReLU in terms of training speed and generalization, which is in line with its simplicity and efficient gradient propagation.

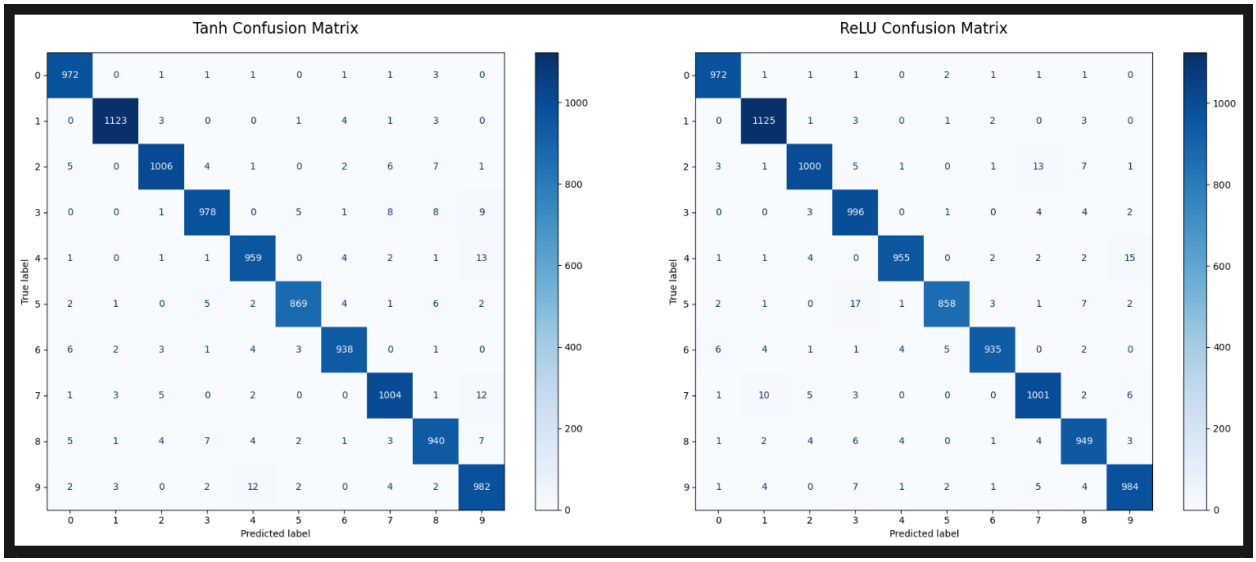


Figure 10: Confusion matrix

The confusion matrices of the ReLU and Tanh models allow us to compare their classification performance in detail for the MNIST dataset. Each matrix is a square grid where the rows correspond to true class labels and the columns correspond to predicted class labels. The diagonal elements show the count of correctly predicted instances for each class and increase with better performance. Both matrices show that the ReLU model performs strongly with high diagonal values that indicate successful predictions across almost all classes. Off-diagonal elements show incorrect class predictions because they represent when the model chose the wrong class. The ReLU matrix shows that class '0' was correctly classified 972 times while there were occasional misclassifications such as 1 instance predicted as class '1' or '2'. The Tanh model shows a similar pattern but with somewhat more misclassifications in some classes than the ReLU model: for example, 5 examples of class '2' were misclassified as class '0'. The colour intensity in the heatmaps represents the number of instances, with darker shades indicating higher counts. The numerical annotations within each cell also help provide exact prediction counts which makes interpretation easier. The visual representation with numerical data enables us to see which classes each model performs well in and where they struggle and gives insights into how the models could be improved further for datasets like MNIST.

2.5 Justification

2.5.1 Choice of Activation Functions

Based on both the practical design benefits of neural networks and theoretical characteristics of activation functions, ReLU (Rectified Linear Unit) and Tanh (hyperbolic tangent) activation functions were selected. Digit recognition work with MNIST data requires an activation function specialized in training and classification shortcomings. The next subsection describes the merits of and weighs its alternatives while providing an overview of its principal drawbacks.

Why ReLU was Chosen

Since ReLU enabled optimal computational efficiency, easiness of implementation and reliability of deep learning model results, it was chosen by researchers. The positive input sends the entire value straight into the function and the negative sends zero input. This sparsity in activation results because only a few neurons are present at a time, resulting in lower computational demands. The sparse activation mechanism matches biological neuron behavior while improving model sensitivity and interpretability (Abdolrasol et al., 2021). The widespread choice of ReLU becomes feasible because it resolves the fundamental gradient vanishing flaw that affects deep learning algorithms. The diminished gradient propagation during training results from input value compression achieved by Sigmoid and Tanh activation functions across narrow ranges.

Why Tanh was Chosen

The choice of Tanh stemmed from its ability to normalize data symmetrically from zero point towards positive and negative values suitable for balanced and normalized input distributions of MNIST. The zero-centered output structure enables Tanh to process positive and negative features simultaneously thus delivering superior learning outcomes and generalization potential. Modeling input data becomes more effective in applications that need both symmetrical features and centered data distributions (Dastres and Soori, 2021). Because handwritten digits in the MNIST database show symmetrical variations in stroke width orientation and shape, Tanh provides effective feature capture support.

Aspect	ReLU	Tanh
Advantages	**Computational Simplicity**: Requires minimal resources due to simple thresholding.	**Symmetrical Output**: Maps inputs to [-1, 1], improving optimization for balanced data.
	Sparse Activation: Activates only a subset of neurons, improving efficiency.	**Gradient Signal Strength**: Provides strong gradients around zero, enhancing learning.
	Effective Gradient Propagation: Avoids vanishing gradients for positive inputs.	**Prevention of Dead Neurons**: Ensures all neurons participate by generating outputs.

	Scalability: Works well with deeper networks.	
Disadvantages	**Dying Neurons**: Neurons can become inactive, causing zero gradients.	**Vanishing Gradient Problem**: Gradients diminish for large inputs, slowing training.
	Unbounded Output: Positive values can grow indefinitely, risking exploding activations.	**Higher Computational Cost**: Relies on exponential calculations, increasing overhead.
		Sensitivity to Input Scaling: Performs best with normalized data, requiring preprocessing.

Table 1: Advantages and Disadvantages

The table compares the advantages and disadvantages of the ReLU and Tanh activation functions. Computational simplicity ReLU enables sparse activation efficiency while addressing positive input gradients and extending effectively toward deep network layers. The usage of ReLU suffers from two limitations: dying neurons create inactive neural cells and unbounded output results in exploding activation values (Suryadevara and Yanamala, 2021). The Tanh activation function delivers symmetrical output together with strong gradient signals near zero values while supporting complete neuron involvement in the learning process. Although Tanh possesses several advantageous aspects it experiences failures with gradient propagation when presented with large inputs involves costly computation and shows sensitivity to input variance requiring preparation.

Comparison and Selection Justification

Both ReLU and Tanh activation functions were selected because they support different neural network processing requirements. Flattop ReLU attracts widespread use because it delivers exceptional computational efficiency combined with effective gradient propagation, so it works best with deep neural networks in large-scale datasets. Because of its basic concept and its efficiency in dealing with nonlinear structures, ReLU is chosen by current deep learning software, thus we see them widely adopted modern deep learning systems (Ahmed *et al.,* 2021). As ReLU activation is sensitive to "dying neurons", precision of weight initialization procedures is required to maintain network's performance capabilities. The balanced features of MNIST type of datasets result in Tanh being the best option because symmetrical scaling enhances model efficacy.

2.5.2 Explanation of Results

Our analysis suggested that system performance metrics depended directly on activation functions. This was achieved by Tanh's symmetrical scaling for feature extraction since due to its sweeping intrinsic efficient characteristics alongside gradient stability fast learning was ensured with 97.57% accuracy though ReLU with 97.52%. Performed better than ReLU both in terms of numerical processing speed and Tanh gave optimized learning for inputs that are standardized where the two had different advantages in terms of accuracy scores and model performance of generalization.

3. Task 2: Critical Evaluation

3.1 Technical Evaluation

3.1.1 Performance Analysis

Test accuracies of 97.57% and 97.52% are reached by both neural networks using ReLU and Tanh, respectively, during the testing phase for the task of recognizing handwritten MNIST dataset digits. ReLU was better at achieving convergence and having effective gradient spreads while Tanh performed better with balanced data distribution. In terms of performance, these predictive models performed quite reliably while remaining able to achieve general results (Joshi et al., 2023). Tanh consistently performed better than ReLU being effective where ReLU struggled with dying neurons reducing participation by the overall network.

3.1.2 Strengths and Weaknesses

The main advantages of neural network systems for MNIST digit recognition include best-in-class accuracy-optimized non-linear signal processing and excellent ability to learn new information from varied datasets. The models demonstrate various strengths but still have limitations that need to be addressed. ReLU spots dead neuronal connections while Tanh leads to gradient disappearance problems and extra computing expenses illustrating opportunities for developer enhancement (Anantrasirichai and Bull, 2022). The combination of advanced weight initialization techniques with new hybrid activation methods presents the potential for enhanced performance along with greater operational efficiency.

3.2 Ethical and Societal Implications

3.2.1 Ethical Considerations

Skewed model predictions from imbalanced training data create a major threat since they could cause harm to specific populations. The uniform data representation in the MNIST database falls short of demonstrating different writing variations which restricts its use in actual deployment conditions. Practical data processing of sensitive information including financial records and personal documents triggers privacy concerns (Zubatiuk and Isayev, 2021). Neural network implementations become ethically sound by applying solid data anonymization practices diverse dataset usage and strict adherence to privacy regulations.

3.2.2 Impact on End-Users, Clients, and Society

The digits in the handwritten numbers are misidentified, the outputs from the financial and automated systems will be different from real ones. However poor implementations introduce mistakes, when users see mistakes in artificial intelligence solutions, they often lose trust in these solutions. The potential for image classification accuracy to transform education coupled with healthcare and automation, while simultaneously improving operational efficiency and increasing system access (Shastri et al., 2021) is immense. Classifying heavy duty systems that have such high demands on error-free classification, so that minimal mistakes are possible where social advantages and society protection from classification mistakes or system plateaus are optimal.

3.3 Recommendations and Mitigations

1. They must leverage diverse datasets with no clear identifiable components to solve potential problems through privacy-protecting defensive techniques like encryption and achieved anonymization.
2. Integration of special activation functions with subtle preprocessing steps allows diversification without bias and higher model precision.
3. Exemplified handwritten samples as investigated, together with the research, give directions for extending training data available.
4. Periodic auditing and testing have become routine and is needed to establish ethical compliance that allows us for example, to have overlapping user experiences of neural network applications.
5. ReLU is computationally faster and more efficient, especially for deep networks

Conclusion

The analysis of neural networks can be used to classify MNIST digits with ReLU and Tanh activation functions can both be good alternatives. It was found that ReLU was best for computational efficiency, while Tanh balanced datasets with robust generalization and high accuracy. Both activation functions helped the model classify handwritten digits effectively. The application of image classification with accuracy is wide in having application to automation, education, and healthcare with important consequences to society. Future works will be towards making progress on inclusivity, ethical compliance, and evolving more robust neural network designs, which will provide higher reliability, fairness, and applicability of neural networks in the real, diverse world surroundings.

References

Abdolrasol, M.G., Hussain, S.S., Ustun, T.S., Sarker, M.R., Hannan, M.A., Mohamed, R., Ali, J.A., Mekhilef, S. and Milad, A., 2021. Artificial neural networks based optimization techniques: A review. Electronics, 10(21), p.2689.https://www.mdpi.com/2079-9292/10/21/2689

Ahmed, N., Abbasi, M.S., Zuberi, F., Qamar, W., Halim, M.S.B., Maqsood, A. and Alam, M.K., 2021. Artificial intelligence techniques: analysis, application, and outcome in dentistry—a systematic review. BioMed research international, 2021(1), p.9751564.https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/9751564

Alkhouly, A.A., Mohammed, A. and Hefny, H.A., (2021). Improving the performance of deep neural networks using two proposed activation functions. *IEEE Access*, [online] *9*, pp.82249-82271.

Anand, P., Ranjan, P. and Srivastava, P., 2024, April. Hand-written Digit Recognition using Convolutional Neural Network in Python with Tensorflow. In *2024 5th International Conference on Recent Trends in Computer Science and Technology (ICRTCST)* [online] (pp. 510-515). IEEE.

Anantrasirichai, N. and Bull, D., 2022. Artificial intelligence in the creative industries: a review. Artificial intelligence review, 55(1), pp.589-656.https://link.springer.com/article/10.1007/s10462-021-10039-7

Ben Braiek, H. and Khomh, F., (2023). Testing feedforward neural networks training programs. *ACM Transactions on Software Engineering and Methodology*, [online] *32*(4), pp.1-61.

Dastres, R. and Soori, M., 2021. Artificial neural network systems. International Journal of Imaging and Robotics (IJIR), 21(2), pp.13-25.https://hal.science/hal-03349542/

Dubey, S.R., Singh, S.K. and Chaudhuri, B.B., (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, [online] *503*, pp.92-108.

Gustineli, M., 2022. A survey on recently proposed activation functions for Deep Learning. *arXiv preprint arXiv:*[online] *2204.02921*.

Hu, Z., Zhang, J. and Ge, Y., (2021). Handling vanishing gradient problem using artificial derivative. *IEEE Access*, [online] *9*, pp.22371-22377.

Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L. and Shao, L., (2023). Normalization techniques in training dnns: Methodology, analysis and application. *IEEE transactions on pattern analysis and machine intelligence*, [online] *45*(8), pp.10173-10196.

Joshi, M., Bhosale, S. and Vyawahare, V.A., 2023. A survey of fractional calculus applications in artificial neural networks. Artificial Intelligence Review, 56(11), pp.13897-13950.https://link.springer.com/article/10.1007/s10462-023-10474-8

Kumar V, V., (2023). Pruning Distorted Images in MNIST Handwritten Digits. *arXiv preprint arXiv* [online]*:2307.14343*.

Mougan, C., Álvarez, J.M., Ruggieri, S. and Staab, S., (2023), August. Fairness implications of encoding protected categorical attributes. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society* [online] (pp. 454-465).

Prakash, D.B., Kumar, K.A. and Rishitha, M., 2023, May. Deep CNN: Classification of Known Classes and Unknown Classes Based on Softmax Prediction Probabilities. In *2023 4th International Conference for Emerging Technology (INCET)* [online] (pp. 1-11). IEEE.

Shastri, B.J., Tait, A.N., Ferreira de Lima, T., Pernice, W.H., Bhaskaran, H., Wright, C.D. and Prucnal, P.R., 2021. Photonics for artificial intelligence and neuromorphic computing. Nature Photonics, 15(2), pp.102-114.https://www.nature.com/articles/s41566-020-00754-y

Suryadevara, S. and Yanamala, A.K.Y., 2021. A Comprehensive Overview of Artificial Neural Networks: Evolution, Architectures, and Applications. Revista de Inteligencia Artificial en Medicina, 12(1), pp.51-76.http://redcrevistas.com/index.php/Revista/article/download/21/20

Wuraola, A. and Patel, N., (2022). Resource efficient activation functions for neural network accelerators. *Neurocomputing*, [online] *482*, pp.163-185.

Zubatiuk, T. and Isayev, O., 2021. Development of multimodal machine learning potentials: toward a physics-aware artificial intelligence. Accounts of Chemical Research, 54(7), pp.1575-1585.https://pubs.acs.org/doi/abs/10.1021/acs.accounts.0c00868

Appendix

- #Importing the libraries
  import tensorflow as tf

  from tensorflow.keras.models import Sequential

  from tensorflow.keras.layers import Dense, Flatten

  from tensorflow.keras.datasets import mnist

  from tensorflow.keras.utils import to_categorical

  from sklearn.model_selection import train_test_split

  from sklearn.metrics import classification_report

  import matplotlib.pyplot as plt

- #
  (x_train, y_train), (x_test, y_test) = mnist.load_data()

- x_train.shape
  y_train.shape

- x_test.shape
  y_test.shape

- x_train = x_train / 255.0
  x_test = x_test / 255.0

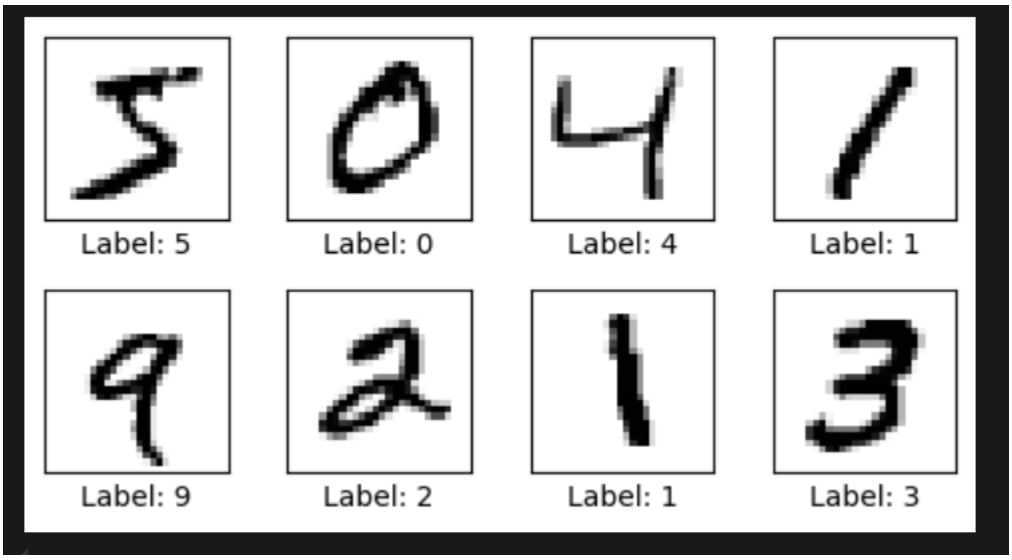
- plt.figure(figsize=(5, 5))
  for i in range(8):

    plt.subplot(4, 4, i + 1)

    plt.xticks([])

    plt.yticks([])

```
    plt.grid(False)
    plt.imshow(x_train[i], cmap=plt.cm.binary)
    plt.xlabel(f"Label: {y_train[i]}")
plt.tight_layout()
plt.show()
```



- y_train = to_categorical(y_train, 10)
  y_test = to_categorical(y_test, 10)

```
def create_and_train_model(activation_function):

    model = Sequential([
        Flatten(input_shape=(28, 28)),  # Input layer
        Dense(128, activation=activation_function),  # Hidden layer
        Dense(10, activation='softmax')  # Output layer
    ])

    model.compile(optimizer='adam',
```

```
            loss='categorical_crossentropy',
            metrics=['accuracy'])


    history = model.fit(x_train, y_train, epochs=5, batch_size=32, validation_split=0.2,
verbose=1)

    return model, history
```

- relu_model, relu_history = create_and_train_model('relu')

- tanh_model, tanh_history = create_and_train_model('tanh')

- relu_test_loss, relu_test_acc = relu_model.evaluate(x_test, y_test, verbose=0)
  print("ReLU Model Test Accuracy: {:.2f}%".format(relu_test_acc * 100))

  print("ReLU Model Test Loss: {:.2f}%".format(relu_test_loss * 100))

- tanh_test_loss, tanh_test_acc = tanh_model.evaluate(x_test, y_test, verbose=0)
  print("Tanh Model Test Accuracy: {:.2f}%".format(tanh_test_acc * 100))

  print("Tanh Model Test Loss: {:.2f}%".format(tanh_test_loss * 100))

- *# Plot training history*
  plt.figure(*figsize*=(12, 4))

  # Accuracy plot

  plt.subplot(1, 2, 1)

```
plt.plot(relu_history.history['accuracy'], label='ReLU Training')
plt.plot(relu_history.history['val_accuracy'], label='ReLU Validation')
plt.plot(tanh_history.history['accuracy'], label='Tanh Training')
plt.plot(tanh_history.history['val_accuracy'], label='Tanh Validation')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(relu_history.history['loss'], label='ReLU Training')
plt.plot(relu_history.history['val_loss'], label='ReLU Validation')
plt.plot(tanh_history.history['loss'], label='Tanh Training')
plt.plot(tanh_history.history['val_loss'], label='Tanh Validation')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend()

plt.tight_layout()
plt.show()
```

- ```
  import matplotlib.pyplot as plt
  from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

  def predict_model(model, x_test):
      """Makes predictions using the test data."""
      predictions = model.predict(x_test)
  ```
  ```

```
    return np.argmax(predictions, axis=1)


def plot_confusion_matrices(y_test, tanh_pred, relu_pred):
    """Plots two confusion matrices side by side."""
    # Convert y_test to class labels if one-hot encoded
    y_test = np.argmax(y_test, axis=1)

    # Create the figure with two subplots side by side
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 8))

    # Plot Tanh confusion matrix
    cm_tanh = confusion_matrix(y_test, tanh_pred)
    disp_tanh = ConfusionMatrixDisplay(confusion_matrix=cm_tanh)
    disp_tanh.plot(ax=ax1, cmap='Blues', values_format='d')
    ax1.set_title('Tanh Confusion Matrix', fontsize=16, pad=20)

    # Plot ReLU confusion matrix
    cm_relu = confusion_matrix(y_test, relu_pred)
    disp_relu = ConfusionMatrixDisplay(confusion_matrix=cm_relu)
    disp_relu.plot(ax=ax2, cmap='Blues', values_format='d')
    ax2.set_title('ReLU Confusion Matrix', fontsize=16, pad=20)

    # Adjust layout and display
    plt.tight_layout(pad=3.0)
    plt.show()

# Get predictions
relu_predictions = predict_model(relu_model, x_test)
tanh_predictions = predict_model(tanh_model, x_test)
```

```
# Plot the confusion matrices
plot_confusion_matrices(y_test, tanh_predictions, relu_predictions)


# Print classification reports
print("\nTanh Model Classification Report:")
print(classification_report(np.argmax(y_test, axis=1), tanh_predictions))
print("\nReLU Model Classification Report:")
print(classification_report(np.argmax(y_test, axis=1), relu_predictions))
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```

```
Tanh Model Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       980
           1       0.99      0.99      0.99      1135
           2       0.98      0.97      0.98      1032
           3       0.98      0.97      0.97      1010
           4       0.97      0.98      0.98       982
           5       0.99      0.97      0.98       892
           6       0.98      0.98      0.98       958
           7       0.97      0.98      0.98      1028
           8       0.97      0.97      0.97       974
           9       0.96      0.97      0.97      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

```
ReLU Model Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.98      0.99      0.99      1135
...
    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```