On the Effectiveness of Piecewise Activation Approximations for Long-Term Short-Memory Networks

Srinivas Rahul Sapireddy
School of Science and Engineering
University of Missouri-Kansas City
Kansas City, Missouri, USA
ssdx5@umkc.edu

Mostafizur Rahman
School of Science and Engineering
University of Missouri-Kansas City
Kansas City, Missouri, USA
rahmanmo@umkc.edu

Abstract

This research introduces an LSTM-based RNN model utilizing linear approximations of the sigmoid and tanh activation functions for sentiment analysis. These piecewise approximations express the sigmoid and tanh graphs in simple linear functions that can be implemented with just a few multiplications and additions in the hardware. The primary goal is to improve computational efficiency by reducing execution time while maintaining accuracy. Integrating these functions into LSTM units accelerates training and improves resource utilization.

Experimental results show that models with piecewise activation functions achieve comparable accuracy, with the base model reaching 99.72% versus 99.86% for the standard LSTM. Notably, training time is reduced by 3.1×, and evaluation speed improves by 1.17×. The three-region linear approximation for the sigmoid function significantly lowers runtime, executing in 0.001168 seconds compared to 0.005067 seconds for the standard sigmoid (4.34× speedup). Similarly, the piecewise tanh function runs in 0.001168 seconds, outperforming the standard tanh at 0.008636 seconds (7.23× improvement). These results demonstrate the potential of linear activation approximations to reduce computational overhead while maintaining accuracy and improving model performance, offering a promising solution for optimizing neural networks in resource-limited environments.

CCS Concepts

• **Computing methodologies** → **Neural networks**; *Reinforcement learning*; *Machine learning algorithms.*

Keywords

Deep Learning, Activation Functions, Long Term Short Memory, Linear Approximation

ACM Reference Format:
Srinivas Rahul Sapireddy and Mostafizur Rahman. 2025. On the Effectiveness of Piecewise Activation Approximations for Long-Term Short-Memory Networks. In *Great Lakes Symposium on VLSI 2025 (GLSVLSI '25), June 30–July 02, 2025, New Orleans, LA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3716368.3735217

GLSVLSI '25, New Orleans, LA, USA

ACM ISBN 979-8-4007-1496-2/25/06
https://doi.org/10.1145/3716368.3735217

1 Introduction

LSTM networks are particularly effective at handling sequential data, such as in natural language processing, where understanding the context and relationships between words over extended sequences is crucial. The success of LSTMs is primarily due to the use of carefully designed activation functions within their structure. These networks rely on activation functions like sigmoid and tanh in their gates (input, forget, and output) to regulate the flow of information. The sigmoid function helps to determine what information to keep or discard, as it outputs values between 0 and 1, effectively acting as a gating mechanism [1], [2]. The tanh function which outputs values between -1 and 1, plays a key role in updating the cell state by ensuring that the information added or removed is scaled appropriately [1], [2], [4]. These activation functions are crucial in controlling the learning process by ensuring that gradients are maintained during backpropagation, mitigating issues such as the vanishing gradient problem that often plagues traditional RNNs [5][6]. By using these functions, LSTMs can capture long-range dependencies in data while maintaining stable training, making them highly effective for tasks like language modeling and other sequential data processing tasks [3].

Piecewise linear approximations of sigmoid and tanh functions have been explored to tackle these challenges. The computational load can be significantly reduced by breaking down these non-linear functions into simpler linear segments [8], [9]. These approximations mimic the behavior of the original activation functions but require fewer computations, leading to faster processing and lower power consumption. This makes them ideal for hardware implementations, such as field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs), where optimizing resource usage is critical for performance and energy efficiency [4]. This research introduces piecewise linear approximations for sigmoid and tanh functions within LSTM networks. We aim to improve computational efficiency by simplifying these functions without sacrificing the model's accuracy. Software simulations were carried out on Google Colab using an A100 GPU. These experiments demonstrate that the piecewise linear approximation of activation functions achieved a higher accuracy of 99.72% compared to 99.86% for the standard LSTM model. Additionally, the piecewise activations reduced training runtime by approximately 67.6% (1999.6291 seconds vs. 6176.1211 seconds) and evaluation runtime by 14.6% (3.4111 seconds vs. 3.9950 seconds) compared to the base model. The results show that these piecewise linear activation functions significantly optimize LSTM networks for hardware and software environments. The reduced computational overhead makes these

functions especially useful in resource-constrained settings, such as real-time systems or embedded applications [4].

This paper presents several key contributions to the field of neural network-based sentiment analysis:

- Developed and implemented piecewise linear approximations for sigmoid and tanh activation functions in LSTM networks.
- Achieved similar accuracy results (99.72% vs. 99.86%) compared to the standard LSTM model while reducing training time by approximately 67.6% and evaluation time by 14.6%.
- Conducted software-based evaluations using Google Colab (A100 GPU) highlighting significant performance improvements in runtime efficiency.

The sections of this paper are structured as follows: Section 2 provides an overview of the background related to LSTM networks, focusing on the challenges posed by traditional activation functions. How to construct and model piecewise linear functions for the sigmoid and tanh activation functions is discussed in Section 3. Section 4 outlines the proposed method for implementing piecewise linear approximations of the sigmoid and tanh activation functions and describes the model architectures used to evaluate the effectiveness of the custom activation functions in software environments. Section 5 presents the evaluation results, comparing the accuracy of piecewise linear approximated activations, standard activations, and runtime metrics, and provides a performance comparison of the proposed piecewise linear activation functions. Finally, Section 6 concludes the paper with a summary of findings and potential future work.

2 Literature Review

To boost computational efficiency and shorten training times, previous studies have explored using piecewise linear approximations to substitute conventional nonlinear activation functions in LSTM components like the forget, input, and output gates. These approximations simplify mathematical operations, enabling faster computations and lowering memory requirements [4].

Research has indicated that piecewise linear approximations offer distinct advantages in deep neural networks by streamlining linear transformations and nonlinear activations. Agostinelli et al., for instance, introduced Adaptive Piecewise Linear (APL) units that dynamically adjust linear segments during training, resulting in notable gains in both speed and performance [8]. The selection of the proper activation function is, therefore, a key factor in determining a network's training efficiency and overall effectiveness, with several studies confirming the benefits of linear approximations in deep learning models [1], [7], [11].

Implementing piecewise linear functions in LSTM architectures offers significant advantages over traditional LSTM models. Computational complexity is reduced by approximating sigmoid and tanh functions with piecewise linear functions, resulting in faster training times and lower memory consumption [4]. This efficiency is significant for large-scale applications where traditional LSTM models may struggle with resource constraints. Moreover, piecewise linear functions help address vanishing gradient issues, further enhancing the performance and reliability of LSTM networks [6], [9], [13]. This is particularly relevant for hardware implementations, where reducing computational complexity can significantly improve processing speed and energy efficiency.

In this paper, we evaluate the performance of models with piecewise linear activation functions over 50 epochs to provide a comprehensive assessment of their effectiveness. Unlike Agostinelli et al., who introduced adaptive piecewise linear units primarily for deep feedforward networks, our work extends the concept to LSTM networks. It evaluates their impact on sequential tasks such as sentiment analysis. This extended evaluation period allows for a thorough understanding of the impact of the piecewise linear functions on the performance of LSTM networks, particularly in terms of training efficiency and runtime. Our findings demonstrate that piecewise linear approximations maintain accuracy and significantly reduce computational overhead, enhancing the efficiency and scalability of deep learning models for hardware implementations.

3 Piecewise Linear Approximation of Sigmoid and Tanh Functions

To derive the piecewise linear approximations for the sigmoid and tanh functions, we use a structured method that emphasizes computational simplicity. Transition points on the curve, such as $x = -2.5$ and $x = 2.5$ for tanh, are selected to divide the function into regions representing lower saturation, central slope, and upper saturation. The approximation is intentionally offset—for example, returning 0.5 at $x = 0$ to ease implementation while maintaining the overall shape and behavior of the original function.

The dashed curves in Figures 1(a) and 1(b) show the original sigmoid and tanh functions. The linear approximations (red lines) effectively replace the steep central regions with a simpler linear slope while maintaining minimal error. The constant regions (green and blue) simplify the flat areas of the functions, reducing hardware complexity.

3.1 Constructing the Piecewise Linear Function

Using the slopes calculated, we construct the piecewise function by defining different linear equations for different intervals of x. For the sigmoid function:

$$f(x) = \begin{cases} 0 & \text{if } x \leq -2.5, \\ 0.5 + 0.231x & \text{if } -2.5 < x \leq 2.5, \\ 1 & \text{if } x > 2.5. \end{cases}$$

For the tanh function:

$$f(x) = \begin{cases} -1 & \text{if } x \leq -2.5, \\ 0.5 + 0.231x & \text{if } -2.5 < x \leq 2.5, \\ 1 & \text{if } x > 2.5. \end{cases}$$

The piecewise linear approximations for the sigmoid and tanh functions are designed to simplify these non-linear functions into easily computable segments, particularly for use in hardware implementations where computational efficiency is crucial. Based on their behavior in different parts of the curve, the functions are divided into three regions.

For the sigmoid function, the first region ($x \leq -4$) is where the output is nearly flat and close to zero. This is approximated by a

constant value of 0, reflecting the minimal change in the function's output in this range. The second region ($-4 < x \leq 4$) is where the sigmoid curve has the steepest slope, rapidly transitioning from 0 to 1. Here, the function is approximated by a linear equation, $f(x) = 0.5 + 0.231x$, which captures the essence of this rapid change while maintaining computational simplicity. The third region ($x > 4$) is where the sigmoid function flattens out again, approaching an output of 1, and is thus approximated by a constant value of 1.

Similarly, the tanh function is divided into regions where its output is flat and near its bounds ($x \leq -2$ for -1, and $x > 2$ for 1), and a central region ($-2 < x \leq 2$) where the function transitions between these bounds. In the central region, a linear approximation is used to capture the steepest part of the tanh curve, similar to the approach used for the sigmoid function.

The regions are divided based on the key characteristics of the sigmoid and tanh functions. The flatter regions, where the functions output values close to their asymptotes, are represented by constants to minimize computational effort. The steeper, more dynamic regions are represented by linear equations that approximate the slope of the original functions, balancing accuracy with simplicity. This method ensures that the functions' essential behavior is preserved while significantly reducing the complexity of their computation.

4 Model Evaluation

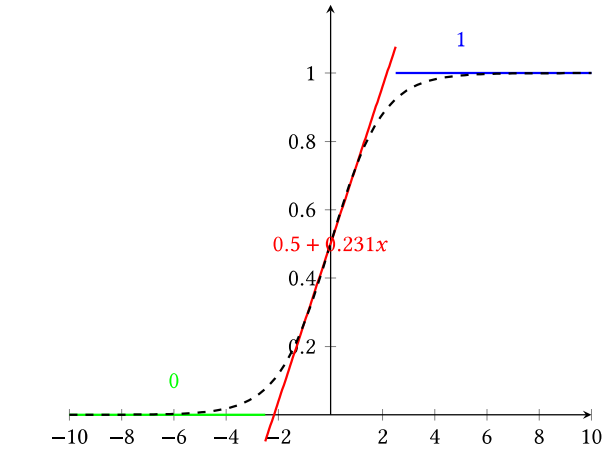
We conducted model evaluation by replacing the traditional sigmoid and tanh activations with piecewise linear approximations in the LSTM architecture to evaluate the robustness and efficiency of these functions.

4.1 Model Architecture

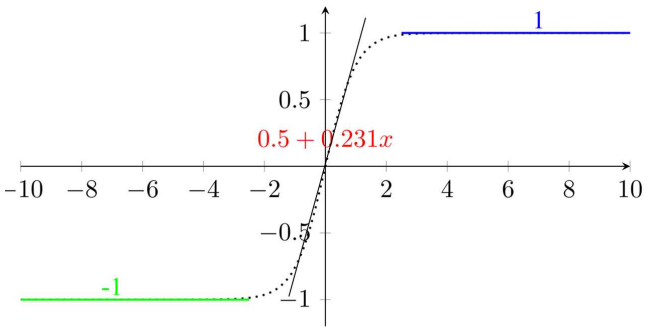
The proposed model, consisting of five key components, efficiently processes and classifies sentiment in movie reviews. Using linear piecewise activation functions enhances computational efficiency, potentially accelerating training and conserving resources, which is particularly beneficial for hardware implementations by reducing complexity and improving performance and energy efficiency.

This research employs the IMDB movie review dataset, comprising 50,000 reviews labeled as positive or negative based on their ratings. Reviews with ratings of four or below are classified as negative, while those rated seven or above are considered positive [14]. A subset of 10,000 balanced reviews is used for the experiment, with 70% designated for training and 30% for testing to support effective learning and accurate performance evaluation [14]. Each review is converted into a sequence of integers using word-to-index mapping, and all sequences are padded to a maximum length of 100 to maintain consistent input dimensions [15]. A custom LSTM-based model, constructed with three LSTM layers containing 64 units each, is utilized for binary sentiment classification. The model is trained using the Adam optimizer and integrates outputs from all three LSTM layers, including the flattened output of the final layer, to enhance feature representation. This design leverages the sequential processing capabilities of LSTM networks, making it well-suited for natural language processing tasks.

By integrating piecewise linear activation functions into the LSTM architecture, we improve computational efficiency. These



(a) Piecewise Linear Approximation of sigmoid activation function.



(b) Piecewise Linear Approximation of tanh activation function.

Figure 1: Piecewise Linear Approximation of the sigmoid (top) and tanh (bottom) activation functions

functions offer a lightweight alternative to standard activations, reducing training time by 67.6% and evaluation time by 14.6%, with minimal accuracy loss. The Piecewise LSTM achieved 99.78% on AG News and 98.99% on SNLI, compared to 99.89% and 99.82% with the Base LSTM, demonstrating efficient convergence, reduced resource usage, and good generalization across tasks. These improvements make the approach well-suited for deployment in resource-constrained environments. Moreover, the method retains the core structure of LSTM networks, allowing seamless integration into existing deep learning pipelines without major architectural changes.

5 Results

We compare the accuracy scores between standard LSTM networks and those using piecewise activation functions. This comparison highlights the performance differences between traditional activation and the proposed piecewise linear functions within the LSTM architecture.

5.1 Model Performance

After training for 50 epochs, models incorporating piecewise linear activation functions for the sigmoid and tanh gates demonstrated

notable improvements in runtime efficiency and comparable performance metrics. Figure 2 compares the performance of models with piecewise linear activations to those using standard activations.

The base LSTM model achieved an accuracy of 99.86% with a training runtime of 6176.1211 seconds and an evaluation runtime of 3.9950 seconds. It recorded a test loss of 3.1135 and a test accuracy of 82.18%. The model evaluation results for the base model include an F1-score of 0.8212, an AUC-ROC of 0.8930, and a prediction runtime of 1121.7057 seconds, as shown in Table 1.

In contrast, the LSTM model with piecewise linear activations (custom model)achieved an accuracy of 99.72% with a significantly reduced training runtime of 1999.6291 seconds and an evaluation runtime of 3.4111 seconds. The custom model recorded a lower test loss of 2.7334 but a slightly lower test accuracy of 80.02%. Additionally, the custom model achieved an F1-score of 0.8049, an AUC-ROC of 0.8735, and a substantially faster prediction runtime of 138.7048 seconds, as shown in Table 1 and Figure 2.

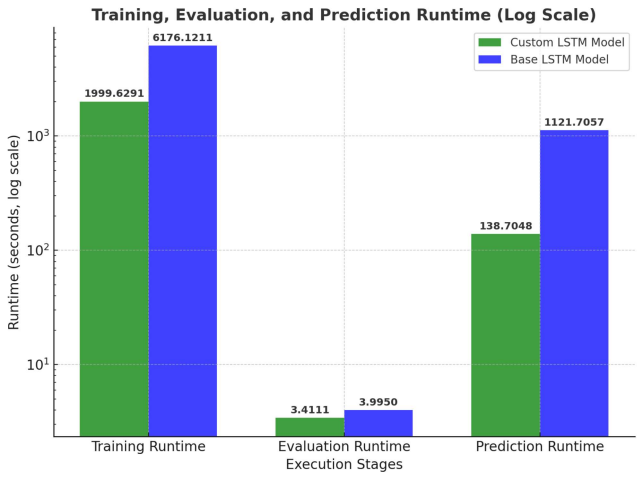


Figure 2: Runtime, evaluation, and prediction runtime comparison between base LSTM networks and LSTM networks with piecewise activation functions.

These results highlight the efficiency gains of piecewise linear approximation of activation functions. Although the base model exhibited slightly higher accuracy and F1-score, the custom model significantly outperformed it in terms of training and prediction runtime, reducing prediction time by approximately 88% and training time by approximately 67.6%. The lower test loss observed in the custom model further indicates a better fit during training. This demonstrates that integrating custom piecewise activation functions into LSTM networks can provide efficient and accurate sentiment analysis, particularly in resource-constrained environments.

5.2 Piecewise Activation Function Performance

The LSTM classifier showed similar accuracy across various combinations of piecewise activations used in the input, forget, and output gates. Notably, models with piecewise linear approximations for the sigmoid and tanh functions in the input and output gates exhibited significant improvements in training efficiency and model performance.

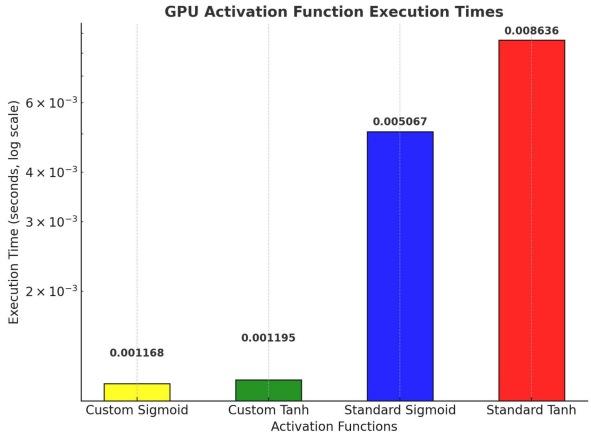
The execution times for various activation functions, as shown in the chart, indicate clear performance differences. The Piecewise Sigmoid function (0.001168 seconds) is the fastest, followed by the Piecewise Tanh function (0.001195 seconds), highlighting the efficiency of piecewise linear approximations. Among the standard functions, Standard Sigmoid (0.005067 seconds) and Standard Tanh (0.008636 seconds) are significantly slower, demonstrating the computational advantage of Piecewise approximations ran on a GPU, as shown in Figure 3. (b). This comparison underscores the benefits of piecewise linear activations, which are particularly valuable in hardware implementations where speed and power efficiency are critical. The reduced execution times for Piecewise activations indicate their potential to accelerate training and inference processes, especially in resource-constrained environments. Additionally, variations in output patterns between piecewise and standard functions may influence learning behavior, potentially leading to faster convergence and reduced computational overhead in neural networks. These results highlight the promise of piecewise functions for improving performance in both software and hardware-based machine learning models.

Figure 3. (a) shows execution times for various activation functions compared and run on an Intel i5 processor. It highlights the performance of two base functions—Sigmoid and Tanh—alongside two piecewise versions, Custom Sigmoid and Custom Tanh. Among the base functions, standard Tanh takes the longest time to execute, while Sigmoid performs faster but still slower than the piecewise implementations. The piecewise versions of these activation functions significantly reduce the computation time, indicating that they are more efficient in this scenario, especially for the specific processor configuration. This highlights the potential for performance optimization by creating piecewise activation functions tailored to the processor's architecture.

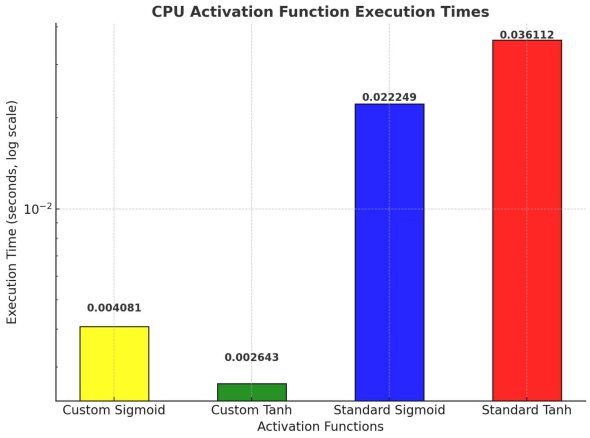
The system used for this comparison is powered by an Intel64 Family i5 processor with 4 physical cores and 8 logical cores, running on Windows 10. The graph highlights the variations in execution times, with the piecewise activation functions outperforming the base functions in speed. Specifically, the Sigmoid Base function recorded an execution time of approximately 0.0022249 seconds, while the Tanh Base function was the slowest at 0.036112 seconds. In contrast, the piecewise versions of these functions demonstrated improved performance, with piecewise Sigmoid completing in 0.004081 seconds and piecewise Tanh in 0.002643 seconds.

The density plots in Figures 3 (c) and (d) compare the output distributions of the piecewise linear approximations to the standard sigmoid and tanh activation functions. The red curve represents the piecewise linear approximation in both plots, while the blue curve shows the base activation function. The plots reveal that while the piecewise approximations generally follow the shape of the base functions, there are noticeable deviations, especially around the peak values. These differences suggest that the piecewise linear functions introduce variations in how outputs are distributed, potentially impacting the learning dynamics in neural networks.

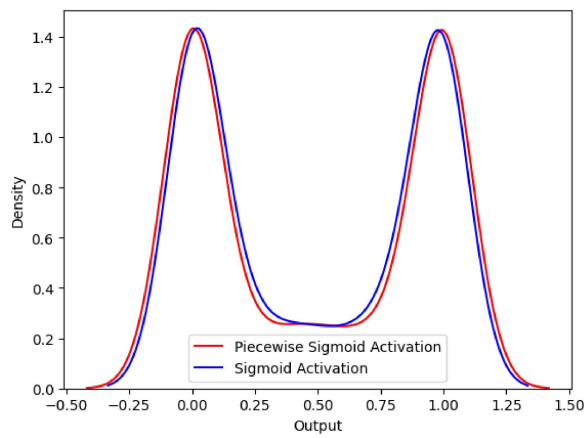
This comparison emphasizes how the piecewise activation functions have been optimized to utilize the processor's architecture more effectively, resulting in significantly shorter execution times.



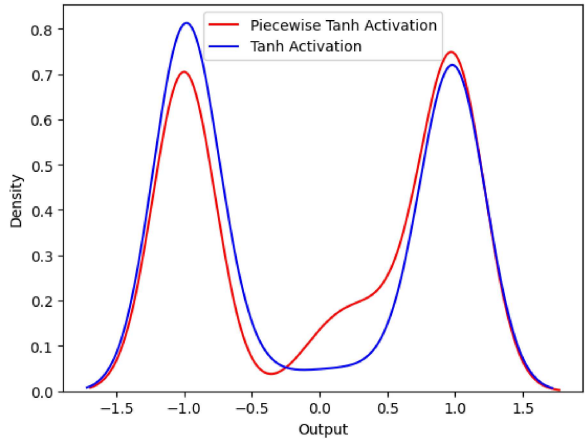
(a) Comparison of execution times of base vs linear approximated activation functions (Intel CPU).



(b) Comparison of execution times of base vs linear approximated activation functions (GPU).



(c) Density Plot of Piecewise Sigmoid Activation vs Base Sigmoid Activation.



(d) Density Plot of Piecewise Tanh Activation vs Base Tanh Activation (GPU).

Figure 3: Comparison of runtime, execution times, and density plots for piecewise and base activation functions in LSTM networks for both sigmoid and tanh activations.

Metric	Custom LSTM Model	Base LSTM Model
Accuracy (Training)	99.72%	99.86%
Training Runtime (seconds)	1999.6291	6176.1211
Evaluation Runtime (seconds)	3.4111	3.9950
Test Loss	2.7334	3.1135
Test Accuracy	80.02%	82.18%
F1-Score	0.8049	0.8212
AUC-ROC	0.8735	0.8930
Prediction Runtime (seconds)	138.7048	1121.7057

Table 1: Comparison of Software Metrics between Custom LSTM and Base LSTM Models on GPU

The reduced computation time makes these piecewise functions ideal for environments with limited computational resources, such as embedded systems or applications requiring real-time processing.

5.4 Comparing Piecewise and Base Activations Under Quantization

This comparison highlights the performance of different activation functions when quantized for hardware-friendly operations, as shown in Figure 4. Fake quantization provided by the TensorFlow library is applied to simulate how these functions would behave on devices like TPUs or edge devices without changing their data type. The results show that piecewise activations, including sigmoid and tanh, are significantly faster than the base versions. Specifically, the piecewise tanh executes in 0.267730 seconds and the piecewise sigmoid in 0.297686 seconds, whereas the base sigmoid and base tanh take 0.301033 seconds and 0.276093 seconds, respectively. The superior speed of the piecewise activations is due to their simple piecewise-linear approximations, which are more efficient than the complex exponential computations used in the base versions. These results suggest that piecewise activations are more suitable for real-time applications and hardware deployments, offering faster and more efficient performance under quantized conditions.

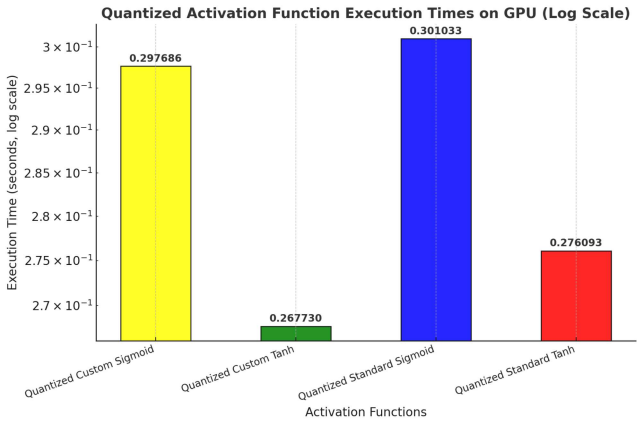


Figure 4: Quantized Activation Function Execution Times.

6 Conclusion

This study compared traditional sigmoid and tanh activation functions with piecewise-linear approximations under quantized conditions for hardware deployment. The results show that the piecewise activations achieve significantly faster execution times than the base versions, demonstrating their efficiency in low-precision operations. The superior performance of the piecewise linear functions is due to their simplified piecewise linear design, which reduces the computational complexity compared to the exponential-based calculations of traditional activations. These findings highlight the potential of piecewise activations for real-time, hardware-optimized applications, where faster processing and reduced power consumption are crucial. Additionally, the results emphasize the suitability of piecewise activations for energy-efficient deployments on edge devices and hardware accelerators. Future research could explore integrating these piecewise linearly approximated activations into a broader range of machine learning models and evaluating their impact on performance and resource efficiency in hardware-constrained environments.

References

[1] Abien Fred Agarap. 2018. Deep learning using rectified linear units (ReLU). *arXiv preprint arXiv:1803.08375* (2018).

[2] Forest Agostinelli, Michael D Hoffman, Peter J Sadowski, and Pierre Baldi. 2014. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830* (2014).

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5, 2 (1994), 157–166.

[4] Ahmed Ghazi Blaiech, Khaled Ben Khalifa, Carlos Valderrama, Marcelo A.C. Fernandes, and Mohamed Hedi Bedoui. 2019. A Survey and Taxonomy of FPGA-based Deep Learning Accelerators. *J. Syst. Archit.* 98, C (Sept. 2019), 331–345.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[6] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *4th International Conference on Learning Representations (ICLR)*.

[8] Adji B Dieng, Rajesh Ranganath, Jaan Altosaar, and David M Blei. 2018. Noisin: Unbiased regularization for recurrent neural networks. *arXiv preprint arXiv:1805.01500* (2018).

[9] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation* 12, 10 (2000), 2451–2471.

[10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proc. 14th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*. 315–323.

[11] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).

[12] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. 6645–6649.

[13] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2016. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems* 28, 10 (2016), 2222–2232.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proc. of the IEEE international conference on computer vision*. 1026–1034.

[15] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[17] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proc. of the 32nd International Conference on Machine Learning*. 2342–2350.

[18] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015).

[19] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv:1504.00941* (2015).

[20] Zachary C. Lipton, John Berkowitz, and Charles Elkan. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv:1506.00019* (2015).

[21] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. 2015. Learning to diagnose with LSTM recurrent neural networks. (2015).

[22] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 142–150.