Expanded Gating Ranges Improve Activation Functions

Allen Hao Huang
Machine Learning and Optimization Laboratory
EPFL

Abstract

Activation functions are core components of all deep learning architectures. Currently, the most popular activation functions are smooth ReLU variants like GELU and SiLU. These are self-gated activation functions where the range of the gating function is between zero and one. In this paper, we explore the viability of using arctan as a gating mechanism. A self-gated activation function that uses arctan as its gating function has a monotonically increasing first derivative. To make this activation function competitive, it is necessary to introduce a trainable parameter for every MLP block to expand the range of the gating function beyond zero and one. We find that this technique also improves existing self-gated activation functions. We conduct an empirical evaluation of Expanded ArcTan Linear Unit (xATLU), Expanded GELU (xGELU), and Expanded SiLU (xSiLU) and show that they outperform existing activation functions within a transformer architecture. Additionally, expanded gating ranges show promising results in improving first-order Gated Linear Units (GLU).

1 Introduction

Activation functions are crucial for introducing non-linearities in deep neural networks [Goodfellow et al., 2016]. Without them, neural networks would essentially function as linear models, unable to capture complex patterns and relationships in data. Early neural networks employed activation functions that produced outputs within a bounded range, such as the binary threshold unit [McCulloch and Pitts, 1943], logistic sigmoid, and hyperbolic tangent [Rumelhart et al., 1986]. However, using these activation functions in deep networks often led to the vanishing gradient problem, which adversely affected performance.

The Rectified Linear Unit (ReLU) [Glorot et al., 2011, Agarap, 2018] became popular by addressing the limitations of traditional bounded activation functions. ReLU is computed by applying the binary threshold unit to the input and then multiplying the result by the input. It avoids the vanishing gradient problem by maintaining a constant, non-zero gradient for positive inputs, leading to faster convergence and improved training efficiency. Smooth ReLU variants, such as the Gaussian Error Linear Unit (GELU) [Hendrycks and Gimpel, 2016] and Sigmoid Linear Unit (SiLU / Swish) [Hendrycks and Gimpel, 2016, Ramachandran et al., 2017, Elfwing et al., 2017], further improved upon ReLU by introducing continuously differentiable functions to replace the binary threshold unit.

Contributions. In this work, our objective is to deepen the understanding of existing activation functions and introduce improvements. Our contributions are as follows:

- **Favorable Properties.** We identify the properties that make activation functions effective. Our findings challenge the conventional belief that ReLU-like properties are necessary for good performance.

Preprint. Under review.

- **Viability of Arctan.** We demonstrate the potential of using arctan as a gating mechanism. Arctan is competitive with and can even outperform existing gating functions in multiple settings.
- **Improving Self-Gated Activation Functions.** We demonstrate that expanded gating ranges can enhance self-gated activation functions. We propose Expanded ArcTan Linear Unit (xATLU), Expanded GELU (xGELU), and Expanded SiLU (xSiLU), and show that they can outperform GELU and SiLU.
- **Improving First-Order Gated Linear Units.** We demonstrate that expanded gating ranges can enhance first-order GLU and help bridge their performance gap with second-order GLU. We propose the first-order Expanded ArcTan GLU (xATGLU), Expanded GEGLU (xGEGLU), and Expanded SwiGLU (xSwiGLU), and show that they can achieve competitive performance with second-order GEGLU and SwiGLU.

2 Preliminaries

2.1 Self-Gated Activation Functions

Incorporating a self-gating mechanism within the activation function enhances gradient flow during training, leading to more stable and efficient learning dynamics within neural networks. Activation functions such as ReLU, GELU, and SiLU can be classified as self-gated activation functions. Gating functions typically output values within a bounded range, usually between zero and one, and can be interpreted as being responsible for controlling the flow of information. Self-gated activation functions can be expressed in the form:

$$a(x) = g(x) \times x \tag{1}$$

Where x is the input, $a(x)$ is the self-gated activation function, and $g(x)$ is the gating function. The gating functions and ranges of popular activation functions, such as ReLU, GELU, and SiLU, are listed in Table 1.

Table 1: **Self-Gated Activation Functions.** ReLU, GELU and SiLU are self-gated activation functions. ReLU uses the binary threshold unit ($x > 0$) as its gating function, GELU uses the standard Gaussian Error CDF $\phi(x)$ as its gating function, and SiLU uses the logistic sigmoid $\sigma(x)$ as its gating function. All the gating functions have a range between zero and one.

$a(x)$	$g(x)$	$g(x)$ range
ReLU	$x > 0$	$[0, 1]$
GELU	$\phi(x)$	$(0, 1)$
SiLU	$\sigma(x)$	$(0, 1)$

Traditionally, the search for activation functions in neural networks has relied heavily on trial and error, with researchers exploring various functions and evaluating their performance empirically [LeCun et al., 1998, Nair and Hinton, 2010, Clevert et al., 2016]. In this paper, we aim to identify the favorable properties of existing activation functions to guide the design of new and improved activation functions.

2.2 Potentially Favourable Properties

GELU and SiLU are currently the state-of-the-art activation functions. We identify the following shared properties that are potentially useful:

1. $g(x)$ is monotonically increasing.
2. $g(x)$ is continuously differentiable.
3. $g(x)$ has a range between zero and one.
4. $a(x) \to 0$ as $x \to -\infty$ and $a(x) \to x$ as $x \to \infty$.
5. $a(x)$ has gradients that can be below zero and above one.

We assume properties 1 and 2 are beneficial and do not run experiments to test them. A monotonically increasing $g(x)$ ensures that more important information is preserved and propagated through the activation function. $g(x)$ is monotonically increasing by convention, although it is possible to construct functionally equivalent activation functions with $g(x)$ being monotonically decreasing. $g(x)$ being continuously differentiable ensures smoother gradient flow during backpropagation and explains why smooth ReLU variants like GELU and SiLU outperform ReLU.

We run experiments to test the benefits of properties 3, 4, and 5. Property 4 can also be interpreted as $a(x)$ being a ReLU-like function. It is important to note the distinction between properties 3 and 4: although property 4 implies property 3, the converse is not true. The first derivatives of both GELU and SiLU have two turning points, which allow for gradients below zero and above one.

2.3 Rescaling Function Ranges

Activation functions appear to require specific gating ranges to perform effectively. For instance, ReLU, GELU and SiLU all have a gating range between zero and one. New gating functions can be created by rescaling the range of bounded functions to this specific gating range. This approach has precedence in existing activation functions. For example, $\phi(x)$, the gating function of GELU, can be interpreted as the result of rescaling $\text{erf}(x/\sqrt{2})$ from the range of $(-1, 1)$ to $(0, 1)$. Similarly, $\sigma(x)$, the gating function of SiLU, can be interpreted as the result of rescaling $\tanh(x)$ from the range of $(-1, 1)$ to $(0, 1)$.

To rescale the range of a function $f(x)$ from $(\min_{\text{old}}, \max_{\text{old}})$ to $(\min_{\text{new}}, \max_{\text{new}})$, we can use the following linear transformation:

$$f_{\text{new}}(x) = (f(x) - \min_{\text{old}}) \left(\frac{\max_{\text{new}} - \min_{\text{new}}}{\max_{\text{old}} - \min_{\text{old}}} \right) + \min_{\text{new}} \tag{2}$$

This transformation adjusts the output range of the function $f(x)$ to the desired new range, facilitating the creation of new gating functions from bounded functions.

2.4 Gated Linear Units

Gated Linear Units (GLU) [Dauphin et al., 2017] are activation functions defined as the component-wise product of two inputs, where one input is passed through a non-linearity. The key difference between GLU and self-gated activation functions is that self-gated activation functions compute the component-wise product of a single input with itself after it has been passed through a non-linearity.

We experiment with first-order GLU of the form:

$$a(x, y) = g(x) \times y \tag{3}$$

We experiment with second-order GLU of the form:

$$a(x, y) = g(x) \times x \times y \tag{4}$$

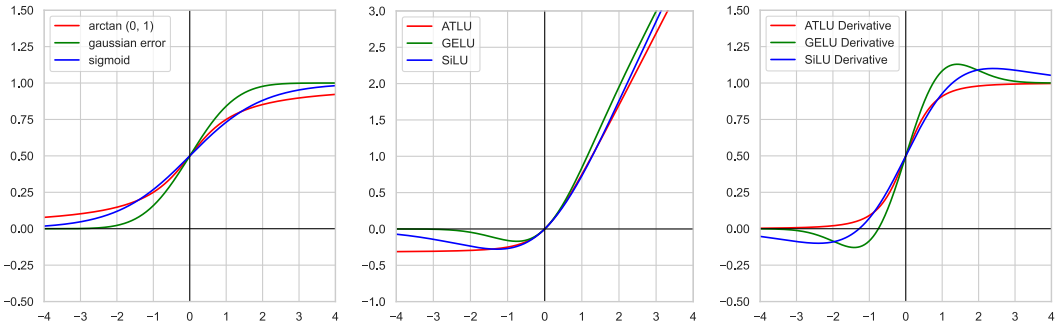
An example of a first-order GLU is the original GLU where $g(x)$ is $\sigma(x)$. Second-order GLU [Shazeer, 2020] introduce an additional multiplicative interaction compared to first-order GLU and tend to have better performance. Popular second-order GLU include ReGLU, where $g(x)$ is $x > 0$, GEGLU, where $g(x)$ is $\phi(x)$, and SwiGLU, where $g(x)$ is $\sigma(x)$.

3 Methodology

3.1 ArcTan Linear Unit

The arctan function, also known as the inverse tangent, is continuously differentiable, monotonically increasing, and has a range of $(-\frac{\pi}{2}, \frac{\pi}{2})$. Given these properties, we explore the viability of using the arctan function as a gating mechanism. We define ArcTan Linear Unit (ATLU) as a self-gated activation function that uses the arctan function scaled to the range of (0,1) as its gating function:

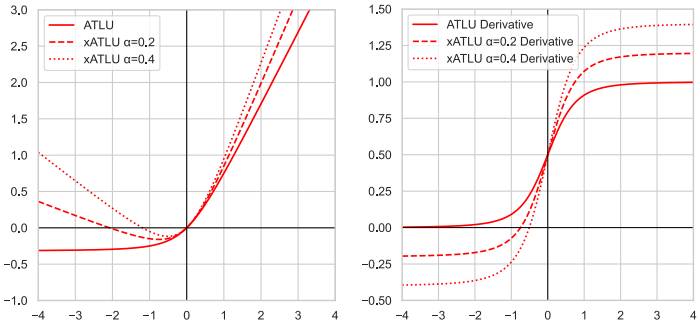
$$\text{ATLU}(x) = x \times \frac{\arctan(x) + \frac{\pi}{2}}{\pi} \tag{5}$$



(a) (b) (c)

Figure 1: **Comparison of ATLU, GELU, and SiLU. (a)** Graph of gating functions for ATLU, GELU, and SiLU. All are continuously differentiable, monotonically increasing, and have a gating range of $(0, 1)$. **(b)** Graph of ATLU, GELU, and SiLU. ATLU differs from GELU and SiLU in that it is not ReLU-like: it does not converge to 0 as x approaches negative infinity and does not converge to x as x approaches positive infinity. **(c)** Graph of first derivatives for ATLU, GELU, and SiLU. The first derivative of ATLU is monotonically increasing and does not have values below 0 or above 1, unlike GELU and SiLU.

A visualization of ATLU, its gating function, and its first derivative is provided in Figure 1. ATLU satisfies properties 1, 2, and 3 but does not satisfy properties 4 and 5, as listed in Section 2.2. Unfortunately, ATLU empirically performs poorly compared to GELU and SiLU, and performs on par with ReLU. Therefore, we deduce that either converging to the same values as ReLU is important and/or allowing gradients below zero and above one is necessary for the effectiveness of an activation function.



(a) (b)

Figure 2: **Visualisation of xATLU.** (a) Graph of xATLU for various fixed values of α. Increasing α makes the activation function converge towards more positive values in both directions. A similar effect occurs when applied to GELU and SiLU. (b) Graph of the first derivative of xATLU for various fixed values of α. Increasing α increases the range of the first derivative. A similar effect occurs when applied to GELU and SiLU.

3.2 Expanded Gating Ranges

We modify ATLU to allow gradient values below zero and above one by expanding the range of $g(x)$. This is done by introducing a single trainable scalar α, initialized to 0, for every MLP block that rescales the gating range to $(-\alpha, 1 + \alpha)$. The rationale for this specific design choice is covered in model ablations in Section 4.3. We name this variant Expanded ATLU (xATLU) and express it as follows:

$$\text{xATLU}(x, \alpha) = x \times \left(\frac{\arctan(x) + \frac{\pi}{2}}{\pi} \times (1 + 2 \times \alpha) - \alpha \right) \tag{6}$$

Note that while the expression can be further simplified, this form provides better clarity. The effect of different fixed values of α on the activation function is illustrated in Figure 2. This same idea can be applied to GELU and SiLU to obtain xGELU and xSiLU. We also test the effectiveness of this idea on $g(x)$ in the GLU setting.

4 Results

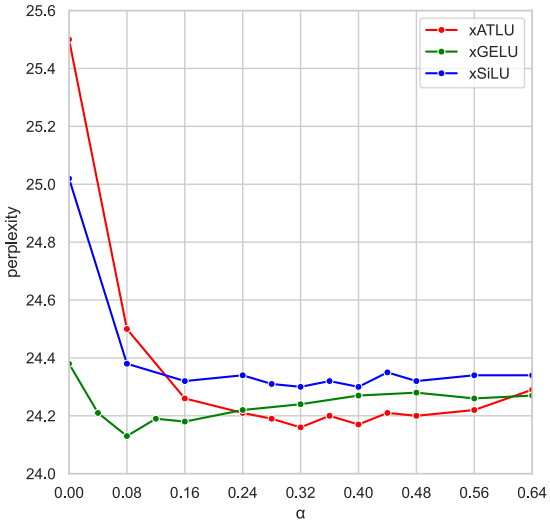


Figure 3: **Effect of Expanded Gating Ranges.** Experiments analysing the effect of using a fixed scalar value for α. Note that using trainable scalar values performs better. The baseline activation functions ATLU, GELU and SiLU are at $\alpha = 0$. Increasing α improves the performance of xATLU, xGELU and xSiLU, allowing them to surpass the performance of GELU and SiLU.

We conduct experiments on standard transformer-based [Vaswani et al., 2017] autoregressive language modeling using code derived from Andrej Karpathy's nanoGPT implementation [Karpathy, 2022]. The experiments are run on single A100 GPUs on the OpenWebText2 dataset [Pilehvar et al., 2022], modifying only the activation function used within the MLP block. For standard MLP blocks, we use an MLP ratio of 4, and for gated MLP blocks, we use an MLP ratio of 8/3. We report the mean and standard error of the last 5 recorded perplexities over 3 different seeds. Full details on experiment setups are given in Appendix A.1.

4.1 Self-Gated Activation Functions

We first run small scale experiments to analyze the impact of fixed values of α for xATLU, xGELU, and xSiLU. The experiment setup is given in Appendix A.1.1 and the results are visualized in Figure 3. We make the following observations:

Arctan is a Viable Gating Function. ATLU is not a competitive activation function but expanded gating ranges allow xATLU to outperform GELU and SiLU.

Expanded Gating Ranges Improve GELU and SiLU. Expanded gating ranges allow xGELU and xSiLU to outperform GELU and SiLU.

ReLU-like Properties are Not Necessary. The performance of xATLU, xGELU, and xSiLU suggests that the optimal gating range is not between 0 and 1, and that the activation function does not need to converge to the same values as ReLU. We find that using a gating range of $(-\alpha, 1 + \alpha)$ improves performance.

Negative Gradient Flow is Necessary. Expanded gating ranges benefit xATLU the most, followed by xSiLU and xGELU. We theorize that there is an optimal amount of gradient flow, mainly negative as suggested by model ablations in Section 4.3, that α is responsible for controlling.

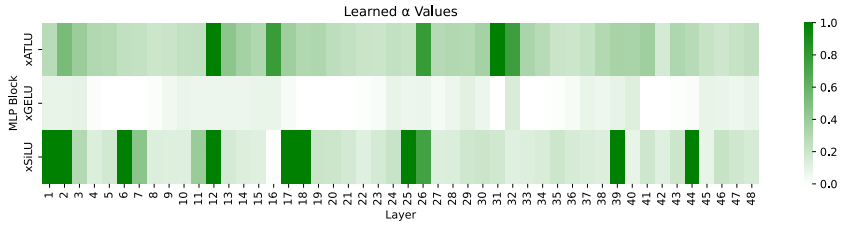


Figure 4: **Comparison of trainable α weights.** We use a heatmap to visualize learned α weights for depth 48 transformer models trained on OpenWebText2 using xATLU, xGELU and xSiLU. Note that we place no restrictions on the values that α can take and the learned values are all positive, which means α is expanding the gating ranges.

Table 2: **Performance of Self-Gated Activation Functions on OpenWebText2.** We report the activation function, the number of parameters, and the perplexity. xATLU, xGELU, and xSiLU outperform GELU and SiLU. xATLU outperforms xGELU and xSiLU.

$a(x)$	$g(x)$ Range	Depth	#Parameter (M)	Perplexity (↓)
ATLU	$(0, 1)$	12	124	18.13 ± 0.10
GELU	$(0, 1)$	12	124	17.58 ± 0.10
SiLU	$(0, 1)$	12	124	17.65 ± 0.10
xATLU	$(-\alpha, 1 + \alpha)$	12	124	$\mathbf{17.36 \pm 0.10}$
xGELU	$(-\alpha, 1 + \alpha)$	12	124	17.47 ± 0.10
xSiLU	$(-\alpha, 1 + \alpha)$	12	124	17.46 ± 0.10
GELU	$(0, 1)$	24	209	16.05 ± 0.09
xATLU	$(-\alpha, 1 + \alpha)$	24	209	$\mathbf{15.71 \pm 0.08}$
xGELU	$(-\alpha, 1 + \alpha)$	24	209	15.79 ± 0.08
xSiLU	$(-\alpha, 1 + \alpha)$	24	209	15.78 ± 0.09
GELU	$(0, 1)$	48	379	14.31 ± 0.08
xATLU	$(-\alpha, 1 + \alpha)$	48	379	$\mathbf{13.93 \pm 0.08}$
xGELU	$(-\alpha, 1 + \alpha)$	48	379	14.05 ± 0.08
xSiLU	$(-\alpha, 1 + \alpha)$	48	379	14.00 ± 0.07

We run larger experiments using a trainable scalar for α for xATLU, xGELU and xSiLU. The experiment setup is given in Appendix A.1.2, the results are shown in Table 2, and a visualization of trained α weights in Figure 4. We make the following observations:

xATLU, xGELU, and xSiLU Outperform GELU and SiLU. The experiments suggest that expanded gating ranges can improve existing activation functions.

xATLU Outperforms xGELU and xSiLU. We theorize that this is due to ATLU having a monotonically increasing first derivative, which results in more favorable training dynamics for xATLU. The first derivatives of GELU and SiLU have similar properties, leading to similar performance for xGELU and xSiLU.

4.2 Gated Linear Units

We run experiments to evaluate the effectiveness of expanded gating ranges for first and second-order GLU. Additionally, we explore the use of arctan scaled to the range (0, 1) in the GLU setting, which we name the ArcTan Gated Linear Unit (ATGLU). The experiment setup is given in Appendix A.1.3, and the results are shown in Table 3. We make the following observations:

Expanded Gating Ranges Improve First-Order GLU The results are similar to self-gated activation functions. First-order ATGLU has poor performance. First-order xATGLU, xGEGLU and xSwiGLU outperforms first-order ATGLU, GEGLU, and SwiGLU. First-order xATGLU outperforms first-order xGEGLU and xSwiGLU.

Table 3: **Performance of Gated Linear Units on OpenWebText2.** We report the gated linear unit and order, the number of parameters, and the perplexity. We refer to GEGLU as second-order GEGLU, SwiGLU as second-order SwiGLU and the original GLU as first order SwiGLU. Expanded gating ranges benefits first-order GLU but not second-order GLU. First-order ATGLU, GEGLU, SwiGLU and second-order ATGLU appear to be able to match the performance of second-order GEGLU and SwiGLU.

$a(x)$	Order	$g(x)$ Range	Depth	#Parameter (M)	Perplexity (↓)
ATGLU	1st	$(0, 1)$	12	124	18.23 ± 0.15
GEGLU	1st	$(0, 1)$	12	124	17.84 ± 0.16
SwiGLU	1st	$(0, 1)$	12	124	17.85 ± 0.15
xATGLU	1st	$(-\alpha, 1 + \alpha)$	12	124	**17.67 ± 0.15**
xGEGLU	1st	$(-\alpha, 1 + \alpha)$	12	124	17.73 ± 0.16
xSwiGLU	1st	$(-\alpha, 1 + \alpha)$	12	124	17.69 ± 0.15
ATGLU	2nd	$(0, 1)$	12	124	17.79 ± 0.15
GEGLU	2nd	$(0, 1)$	12	124	**17.78 ± 0.16**
SwiGLU	2nd	$(0, 1)$	12	124	17.81 ± 0.16
xATGLU	2nd	$(-\alpha, 1 + \alpha)$	12	124	17.82 ± 0.15
xGEGLU	2nd	$(-\alpha, 1 + \alpha)$	12	124	17.80 ± 0.16
xSwiGLU	2nd	$(-\alpha, 1 + \alpha)$	12	124	17.84 ± 0.16
xATGLU	1st	$(-\alpha, 1 + \alpha)$	24	209	**15.55 ± 0.09**
xGEGLU	1st	$(-\alpha, 1 + \alpha)$	24	209	15.62 ± 0.09
xSwiGLU	1st	$(-\alpha, 1 + \alpha)$	24	209	15.57 ± 0.09
ATGLU	2nd	$(0, 1)$	24	209	15.60 ± 0.09
GEGLU	2nd	$(0, 1)$	24	209	15.58 ± 0.09
SwiGLU	2nd	$(0, 1)$	24	209	**15.57 ± 0.09**

Expanded Gating Ranges Do Not Improve Second-Order GLU. Second-order ATGLU is competitive with second-order GEGLU and SwiGLU. This is surprising as arctan needed expanded gating ranges to function properly in the self-gated activation functions and first-order GLU setting. We theorize that both second-order GLU and expanded gating ranges achieve similar effects in facilitating a larger negative gradient flow.

Expanded First-Order GLU Match Second-Order GLU. Our results suggest that expanded gating ranges narrows the performance gap observed between first and second-order GLU. First-order xATGLU, xGEGLU, and xSwiGLU and second-order ATGLU, GEGLU and SwiGLU have similar performance.

4.3 Model Ablations

As expanded gating ranges have the largest impact on ATLU/xATLU, we focus on running model ablations for ATLU/xATLU. The experiment setup is given in Appendix A.1.1 and the results are shown in Table 4. We make the following observations:

Trainable Scalar Over Fixed Scalar. Replacing the trainable scalar α with the best performing fixed scalar α from Figure 3 marginally worsens performance. However, it still significantly outperforms baseline ATLU.

Importance of Negative Gradient Flow. Replacing the gating range of $(-\alpha, 1 + \alpha)$ with $(0, 1 + \alpha)$ significantly worsens performance, α likely plays an important role in increasing the flow of negative gradients. Replacing it with $(-\alpha, 1)$ marginally worsens performance, suggesting that increasing the flow of positive gradients is less important.

Alternative Parameterizations. Replacing the gating range of $(-\alpha, 1 + \alpha)$ with $(-\alpha_1, 1 + \alpha_2)$ results in similar performance. Replacing the scalar weight of α with a per-channel weight, which is model dimension multiplied by mlp ratio, also results in similar performance. We run our experiments with the most simple setup.

Table 4: **Ablation of xATLU.** We run model ablations using fixed scalar values for α, using α to only expand the minimum gating or the maximum gating, using different parameters to control minimum gating and maximum gating, and replacing the trainable scalar weight with a trainable per-channel weight.

$a(x)$	$g(x)$ Range	Depth	#Parameter (M)	Perplexity (↓)
ATLU	$(0, 1)$	12	124	18.13 ± 0.10
xATLU	$(-\alpha, 1 + \alpha)$	12	124	**17.36 ± 0.10**
xATLU	$(-0.32, 1.32)$	12	124	17.47 ± 0.10
xATLU	$(-\alpha, 1)$	12	124	17.53 ± 0.11
xATLU	$(0, \alpha)$	12	124	18.08 ± 0.10
xATLU	$(-\alpha_1, 1 + \alpha_2)$	12	124	17.38 ± 0.10
xATLU	$(-\alpha, 1 + \alpha)$ per channel	12	124	**17.36 ± 0.10**

5 Limitations and Future Work

Running Larger Scale Experiments. Due to limited computational resources, our experiments are relatively small in scale. Larger experiments are necessary to determine if xATLU, xGELU, and xSiLU can consistently outperform GELU and SiLU, and if first-order xATGLU, xGEGLU, xSwiGLU, and second-order ATGLU can compete with second-order GEGLU and SwiGLU.

Impact on Activation Sparsity. Activation sparsity [Mirzadeh et al., 2023, Song et al., 2024] has been identified as a potentially important property for both computational and memory efficiency. Expanding gating ranges deviates from a ReLU-like activation function, which may make it more challenging to achieve activation sparsity through techniques like ReLUfication.

Searching for New Gating Functions. Expanded gating ranges increases the search space of viable gating functions. GELU, SiLU, and Mish [Misra, 2020] all have different expressions for their gating functions but result in similarly shaped activation functions and first derivatives. It is likely that there exists gating functions with completely different expressions to arctan, and have similar activation functions and first derivatives to ATLU.

6 Related Work

ReLU-like Activation Functions. The ReLU activation function gained popularity due to its simplicity and efficiency. Most subsequent work on activation functions has adopted several ReLU-like properties [Hendrycks and Gimpel, 2016, Ramachandran et al., 2017, Elfwing et al., 2017, Misra, 2020]. Our findings suggest that expanded gating ranges can improve multiple activation functions, indicating that some previously considered desirable ReLU-like properties may not be necessary.

Trainable Activation Functions. Several prior works have proposed trainable or adaptable activation functions [He et al., 2015, Ramachandran et al., 2017, Apicella et al., 2021]. However, they have limited effectiveness, and are not used over non-trainable activation functions. Our research suggests that introducing a trainable parameter to control the gating range can be beneficial for activation functions.

7 Conclusion

This work aims to enhance the understanding of activation functions. We identified key properties that contribute to the effectiveness of activation functions and demonstrated the viability of using arctan as a gating function. By expanding the gating ranges, we showed that self-gated activation functions such as xATLU, xGELU, and xSiLU can outperform popular activation functions GELU and SiLU. Furthermore, we demonstrated that expanded gating ranges can also improve first-order GLU and help bridge the performance gap with second-order GLU.

8 Acknowledgements

I would like to thank Martin Jaggi for insightful comments and suggestions during the development of this work.

References

Abien Fred Agarap. Deep learning using rectified linear units (relu), 2018.

Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138:14–32, June 2021. ISSN 0893-6080. doi: 10.1016/j.neunet.2021.01.026.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org, 2017.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2016.

Andrej Karpathy. nanogpt. `https://github.com/karpathy/nanoGPT`, 2022. GitHub repository.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Warren S. McCulloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. Relu strikes back: Exploiting activation sparsity in large language models, 2023.

Diganta Misra. Mish: A self regularized non-monotonic activation function, 2020.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807–814. Omnipress, 2010.

Mohammad Taher Pilehvar, Rafal Jozefowicz, Alec Radford, Ilya Sutskever, and Dario Amodei. Openwebtext2: An open-source replication of webtext2, 2022. Accessed: 2024-05-20.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Chenyang Song, Xu Han, Zhengyan Zhang, Shengding Hu, Xiyu Shi, Kuai Li, Chen Chen, Zhiyuan Liu, Guangli Li, Tao Yang, and Maosong Sun. Prosparse: Introducing and enhancing intrinsic activation sparsity within large language models, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

A Appendix

A.1 Experiment Setup

Table 5: **Shared Hyperparameters.**

Hyperparameter	Value
Batch Size	500
Weight Decay	0.1
Optimizer	adamw
Beta 1	0.9
Beta 2	0.95
Scheduler	cosine
Warmup Percent	0.02
Mininum Learning Rate Ratio	0.1
Dropout	0.0
Head Dimension	64

A.1.1 Effect of expanded gating ranges on xATLU, xGELU and xSiLU

Table 6: **Effect of expanded gating ranges on xATLU, xGELU and xSiLU.** Small scale experiments using fixed scalar for α to test viability of expanding the gating range. Trained on 3.84B tokens.

Hyperparameter	Value
Iterations	15000
Sequence Length	512
Model Dimension	512
Depth	12
Learning Rate	0.002
Gradient Clipping	0.0

A.1.2 Performance of Self-Gated Activation Functions

Table 7: **Performance of Self-Gated Activation Functions.** Larger scale experiments using trainable scalar for α. Trained on 17.92B tokens. Depth 12 models use 0.002 learning rate and Depth 24/48 models use 0.001 learning rate.

Hyperparameter	Value
Iterations	40000
Sequence Length	768
Model Dimension	768
Learning Rate	0.002/0.001
Gradient Clipping	0.0

A.1.3 Performance of Gated Linear Units

Table 8: **Performance of Gated Linear Units.** Larger scale experiments for GLU using trainable scalar for α. Trained on 17.92B tokens. Depth 12 models use 0.002 learning rate and Depth 24 models use 0.001 learning rate. Uses 0.1 gradient clipping do reduce divergence during training due to the poorer training stability of GLU.

Hyperparameter	Value
Iterations	40000
Sequence Length	768
Model Dimension	768
Learning Rate	0.002/0.001
Gradient Clipping	0.1

A.2 Compute Resources Used

For experiments with 40,000 iterations, sequence length 768, model dimension 768 and running on a single A100 GPU, 12 layer experiments take a day, 24 layer experiments take 2 days and 48 layer experiments take 4 days.

A.3 ATLU/xATLU Derivatives

ATLU first derivative

$$\frac{\arctan\left(x\right)+\frac{\pi}{2}}{\pi}+\frac{x}{\pi\cdot\left(x^{2}+1\right)} \tag{7}$$

xATLU first derivative

$$\alpha\left(\frac{\arctan\left(x\right)+\frac{\pi}{2}}{\pi}+\frac{x}{\pi\cdot\left(x^{2}+1\right)}\right)-\frac{\alpha}{2} \tag{8}$$

B Pseudocode

```
import math
import torch
import torch.nn as nn

class xATLU(nn.Module):
    def __init__(self):
        super(xATLU, self).__init__()
        self.alpha = nn.Parameter(torch.zeros(1))
        self.half_pi = math.pi / 2
        self.inv_pi = 1 / math.pi

    def forward(self, x):
        gate = (torch.arctan(x) + self.half_pi) * self.inv_pi
        return x * (gate * (1 + 2 * self.alpha) - self.alpha)


class xGELU(nn.Module):
    def __init__(self):
        super(xGELU, self).__init__()
        self.alpha = nn.Parameter(torch.zeros(1))

    def forward(self, x):
        gate = (torch.erf(x / math.sqrt(2)) + 1) * 0.5
        return x * (gate * (1 + 2 * self.alpha) - self.alpha)


class xSiLU(nn.Module):
    def __init__(self):
        super(xSiLU, self).__init__()
        self.alpha = nn.Parameter(torch.zeros(1))

    def forward(self, x):
        gate = torch.sigmoid(x)
        return x * (gate * (1 + 2 * self.alpha) - self.alpha)
```