1 Introduction

The transformer architecture [9] has revolutionized the AI landscape and enabled the development of large language models (LLMs). However, as these models continue to grow in size. Current trends are forecating running out of high-quality data required for optimal performance [10]. This limitation stimulas the initiative to explore the impact of architectural decisions on smaller models, which this project aims to investigate. By understanding how to optimize smaller models, we can ensure progress of LLMs even with the projected lack of high-quality data.

The choice of activation functions has historically played a crucial role in the advancement of neural networks, such as the shift from Sigmoind activation functions to ReLU (Rectified Linear Unit), which significantly improved training speeds. As the era of large language models (LLMs) unfolded, the development of activation functions continued to evolve, resulting in over 400 documented activation functions [2]. Despite this progress, literature typically compares new activation functions against their immediate predecessors, leading to a gap in the literature: there is no comprehensive comparison of multiple activation functions under consistent conditions. This inconsistency is primarily due to variations in datasets and model sizes used in different studies, highlighting the need for a systematic evaluation. More specifically, with the trend of increasing model sizes, the investiation of impact activation functions have on smaller models was left neglected. This research aims to address this gap by exploring the impact of activation functions on smaller-scale language models with around 10 million parameters.

(Structure of the paper here?)

2 Background and related work

Activation functions are used to introduce non-linearity into neural networks, allowing them to model complex relationships in the data. They are applied to the nodes of the network and are essential for enabling the network to learn and perform a wide range of tasks, beyond what linear models can achieve.

The famous paper "Attention is All You Need" [9] used the state-of-the-art activation function at the time, ReLU. Since then, no significant improvements were made until the introduction of GELU, which quickly became the default activation function in most of the state-of-the-art LLMs [3]. The popularity of GELU stems from its ability to enhance model performance without introducing an efficiency overhead. Despite these advancements, continuous innovation leads to alternatives like GeGLU, noted for its effectiveness [8], also used in last year's winner of the BabyLM challenge [7]. However, a recently published paper suggests a return to ReLU [5], adding further confusion to the search for the optimal activation function. Fortunately, it also provides some clues that guide further exploration and motivate this research.

The paper suggests that suggests that the impact of activation functions diminishes as the model size increases, evident in models with over a billion parameters [5]. This also explains the initial move away from ReLU, since all the research on activation alternatives was done on models with the size of approximately 100 million parameters. Highlighting this finding further motivates the need to investigate activation functions in 10 million parameter models. The impact of activation functions is expected to be more significant in smaller models, and until now, decisions have been made with the trend of increasing model size in mind.

Furthermore, another gap appears in the research on activation functions with trainable parameters (adaptible activaition functions). The possible explanation for this could be the tradeoff between additional trainable parameters and performance. However, this was primarily studied in larger models. The only paper found on adaptive activation functions in LLMs is by Rajanand et al. [6], which found that adaptive activation functions outperform static ones in text-to-text machine translation [NOTE: add the exact numbers from the paper], a task closely related to language models. This suggests that adaptive activation functions could be beneficial for smaller models, but further research is needed to confirm this hypothesis. Given these insights, this research will explore the impact of various activation functions on smaller-scale language models with around 10 million parameters. Hypothesising that at smaller scales, the choice of activation function is crucial and having learnable parameters could be beneficial.

Kolmogorov-Arnold Networks (KAN) represent a recent development in neural network architecture, where activation functions are applied on edges instead of nodes [4]. This approach has been shown to outperform traditional neural networks in some tasks, particularly in scientific applications such as solving partial differential equations. However, at the time of this literature review, it has yet to be tested on language models. The primary benefit of KAN is the optimization of activation on each edge using splines. A spline is a piecewise-defined polynomial function used in interpolation and approximation to create smooth curves through a set of points [ADD REFERENCE]. With a spline on each edge, each edge can have its own custom activation function, trained separately and uniquely shaped. In contrast, adaptive activation functions have the same shape but different gradients. Although this comes with the drawback of an increased number of trainable parameters. This research will experiment with applying KAN to language modeling to assess its efficacy at smaller scales, filling the gap in current literature.

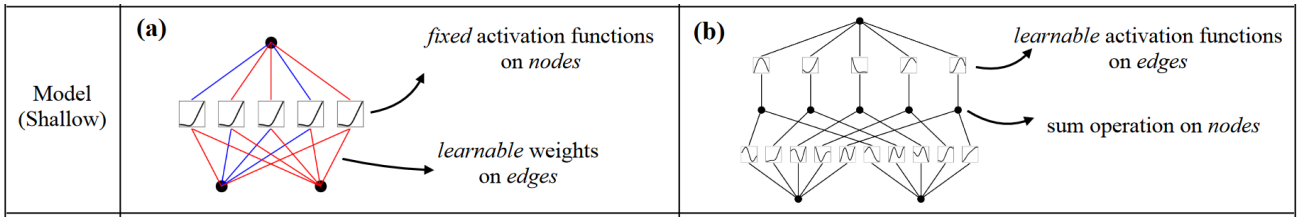

Figure 1: KAN vs MLP [3]

3 Approach

Transformers are composed of multiple layers, each playing a crucial role in processing input data and generating meaningful representations. Among these layers, the Feed-Forward Neural Network (FFN) layer typically consists of two linear transformations with an activation function in between, effectively forming a simple Multi-Layer Perceptron (MLP). This structure allows the default activation function to be switched out with different activation functions for testing, enabling a direct comparison of their performance while keeping everything else the same. To explore the effectiveness of various activation functions, I will modify existing implementations of prominent models like GPT-NEO and RoBERTa from the Hugging Face library

3.1 Choosing activation functions

The activation functions to be evaluated are: ReLU, SiLU, Swish, PReLU, GELU, GEGLU, learnable GELU, and learnable GEGLU. Additionally, the KAN network will be compared against all of these options.

GELU

Currently, the most popular activation function in LLMs is also used as the default activation function in the baseline models GPT-NEO and roBERTa. It is a smooth approximation of ReLU originally defined as $\text{GELU}(x) = x \cdot \Phi(x)$, where $\Phi(x)$ is the Cumulative Distribution Function for the Gaussian Distribution. For optimization purposes, it is implemented as $\text{GELU}(x) = 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right)$ [1].

ReLU and PReLU

ReLU was considered state of the art in the time of original transformer paper [9], but has since been surpassed by other activation functions. I will implement a model

SiLU and Swish

GELU and Learnable GELU

GEGLU and Learnable geglu

Motivate choices of activation functions To explore the effectiveness of various activation functions, I will modify existing implementations of prominent models like GPT-NEO and RoBERTa from the Hugging Face library. According to Mirzadeh et al. (2023), I am interested in evaluating how much worse ReLU performs compared to PReLU, which includes learnable parameters. Additionally, I will test the Swish activation function by starting with SiLU (a baseline version of Swish) and then comparing it to Swish. Furthermore, I will investigate GeGLU by comparing it to the standard GELU activation function. To thoroughly understand the impact, I will also parameterize GELU with learnable components and benchmark it against its standard form. Finally, I will compare these findings against the performance of the KAN network to establish a comprehensive understanding.

The implementation of KAN, as described by the authors of the paper, is available but has shown to be problematic and slow. To address these issues, I will utilize an efficient KAN implementation. This approach requires careful selection of certain parameters, specifically the number of grid intervals (int) and the order of piecewise polynomials (k). Based on recommendations from the paper, I will set these parameters to 3 and 5, respectively. This configuration aims to balance performance and computational efficiency, ensuring that the KAN implementation is both effective and practical for the experiments.

4 Results

Model	BLiMP	G
BERT-Learnable-GELU-9.0M-2L-4H-512C-1024I	59.4	
GPT-Learnable-GELU-9.0M-2L-4H-512C-1024	56.782	6
Same baselines		
GPT-9.0M-2L-4H-512C-1024I	55.6	5
BERT-9.0M-2L-4H-512C-1024I	49.1	5
Best baselines GLUE		
GPT-11.0M-3L-4H-512C-1024I-0	(55.17)	
BERT-11.0M-3L-4H-512C-1024I	(49.49)	5
GPT-PReLU-9.0M-2L-4H-512C-1024I	57.0%	
BERT-PReLU-9.0M-2L-4H-512C-1024I	59.2%	
GPT-SiLU-9.0M-2L-4H-512C-1024I	56.3%	
GPT-ReLU-9.0M-2L-4H-512C-1024I	56.6%	
BERT-SiLU-9.0M-2L-4H-512C-1024I	58.3%	
BERT-ReLU-9.0M-2L-4H-512C-1024I	58.5%	
GPT-Swish-9.0M-2L-4H-512C-1024I	53.7%	
BERT-Swish-9.0M-2L-4H-512C-1024I	58.0%	
BERT-kan2-11.0M-2L-4H-512C-384I	56.69%	6
KAN2-GPT-11.0M-2L-4H-512C-256I	63.43%	48
Learnable GELU Seeds		
GPT-Learnable-GELU-seed4-9.0M-2L-4H-512C-1024I	56.0%	
GPT-Learnable-GELU-seed1-9.0M-2L-4H-512C-1024I	56.3%	
GPT-Learnable-GELU-seed2-9.0M-2L-4H-512C-1024I	57.4%	
BERT-Learnable-GELU-seed4-9.0M-2L-4H-512C-1024I	58.2%	
GPT-Learnable-GELU-seed5-9.0M-2L-4H-512C-1024I	58.3%	
GPT-Learnable-GELU-seed3-9.0M-2L-4H-512C-1024I	58.7%	
BERT-Learnable-GELU-seed3-9.0M-2L-4H-512C-1024I	59.0%	
BERT-Learnable-GELU-seed2-9.0M-2L-4H-512C-1024I	59.3%	
BERT-Learnable-GELU-seed5-9.0M-2L-4H-512C-1024I	59.8%	
BERT-Learnable-GELU-seed1-9.0M-2L-4H-512C-1024I	59.8%	

Table 1: Comparison of models across BLiMP and GLUE datasets

References

[1] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv*, arXiv:1606.08415, Jun. 2023.

[2] V. Kunc and J. Kléma. Three decades of activations: A comprehensive survey of 400 activation functions for neural networks. *arXiv*, arXiv:2402.09092, 2024.

[3] M. Lee. Gelu activation function in deep learning: A comprehensive mathematical analysis and performance. *arXiv*, arXiv:2305.12073, Aug 2023.

[4] Z. Liu and Others. Kan: Kolmogorov-arnold networks. *arXiv*, arXiv:2404.19756, May 2024.

[5] I. Mirzadeh and Others. Relu strikes back: Exploiting activation sparsity in large language models. *arXiv*, arXiv:2310.04564, Oct 2023.

[6] A. Rajanand and P. Singh. Erfrelu: Adaptive activation function for deep neural network.

[7] D. Samuel, A. Kutuzov, L. Øvrelid, and E. Velldal. Trained on 100 million words and still in shape: Bert meets british national corpus. *arXiv*, arXiv:2303.09859, May 2023.

[8] N. Shazeer. Glu variants improve transformer. *arXiv*, arXiv:2002.05202, Feb 2020.

[9] A. Vaswani and Others. Attention is all you need. *arXiv*, arXiv:1706.03762v5, Dec 2017.

[10] P. Villalobos, J. Sevilla, L. Heim, T. Besiroglu, M. Hobbhahn, and A. Ho. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. *arXiv*, arXiv:2211.04325, Oct 2022.