

# 1 Introduction

The transformer architecture [?] 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 [?]. 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 Sigmoid 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 [?]. 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" [?] 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 [?]. 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 [?], also used in last year's winner of the BabyLM challenge [?]. However, a recently published paper suggests a return to ReLU [?], 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 the impact of activation functions diminishes as the model size increases, evident in models with over a billion parameters [?]. 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 (adaptable activation 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. [?], which found that adaptive activation functions outperform static ones in text-to-text machine translation, 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 [?]. 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.

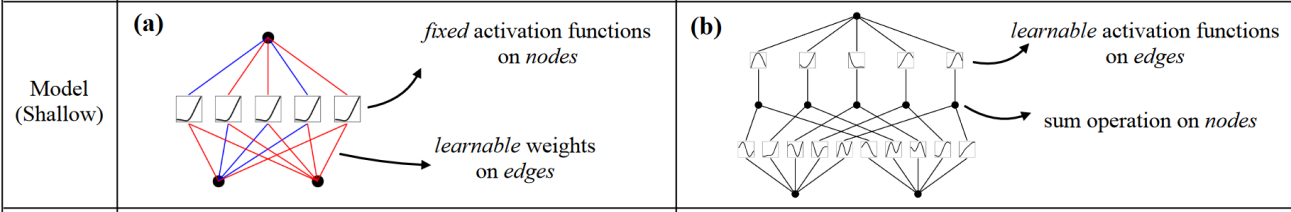


Figure 1: KAN vs MLP [3]

### 3 Approach

To achieve this, I will leverage transformers, specifically focusing on their layered architecture. 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 is particularly significant. It typically consists of two linear transformations with an activation function in between, effectively forming a simple Multi-Layer Perceptron (MLP). The choice of activation function within these layers is critical as it directly impacts the model's performance and ability to learn complex patterns.

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 Experimental setup

**\*\* Research Questions \*\***

- Does choice of activation function matter in smaller language models? - sub1: How do adaptable activation functions with learnable parameters compare to their static counterparts? - sub2: Does use of KAN network instead of MLP improve performance of smaller language model?

**\*\* Dataset \*\*** - tiny stories - why tiny stories and not some other dataset

**\*\* Models \*\*** - Hugging face GPT-NEO as baseline encoder and RoBERTa as baseline decoder

**\*\* Other utils \*\*** - - <https://github.com/Blealtan/efficient-kan>  
- <https://pytorch.org/docs/stable/generated/torch.nn.PReLU.html>  
- <http://arxiv.org/abs/1710.05941> - assumptions or constants use when applying kan network and activation functions

**\*\* Hardware \*\*** - A100

**\*\* Evaluation \*\*** - BLiMP - - SuperGLUE