Adaptive Activation Functions

Does choice of activation function matter in smaller Langaunge Models?

Filip Ignijić

Supervisor(s): Aral de Moor, Responsible Professors: Maliheh Izadi, Arie van Deursen

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 12, 2024

Name of the student: Filip Ignijić
Final project course: CSE3000 Research Project
Thesis committee: Thomas Abeel, Maliheh Izadi, Arie van Deursen, Aral de Moor

Abstract

The rapid expansion of large language models (LLMs) driven by the transformer architecture has introduced concerns about the lack of high-quality training data. This study investigates the role of activation functions in smaller-scale language models, specifically those with approximately 10million (M) parameters, to ensure sustained progress in LLM development despite data limitations. Activation functions, crucial for neural network performance, have evolved significantly, but comprehensive comparisons under consistent conditions remain scarce, especially for smaller parameter count models. This research systematically evaluates traditional and novel activation functions, including learnable variants, and introduces the Kolmogorov-Arnold Network (KAN) to language modeling. Using Hugging Face implementations of GPT-Neo and RoBERTa models, this study assesses performance impacts through the BabyLM evaluation pipeline. TODO: Add results and conclusions.

1 Introduction

The transformer architecture [22] has revolutionized the AI landscape and enabled the development of commerical large language models (LLMs) like ChatGPT. However, as these models continue to grow in size, current trends forcast running out of high-quality data required for optimal performance [23]. This limitation stimulates the initiative to improve sample efficiency motivated by the observation that LLMs are exposed to orders of magnitude more information than a human in their lifetime, yet are certainly not leveraging all this information nearly as efficiently. Thereore, this reserachs aims to investiate the impact of architectural decisions on smaller models which are often left neglected. By understanding how to optimize smaller models, we can ensure the progress of LLMs even with the projected lack of high-quality data. Furthermore, smaller models can be deployed on edge devices can insure privacy and be used for spellcheck, predictive typing, conversational assistance etc.

The activation function in a nerual netowrk, determines if neuron sould be activated or not. 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 simplified computation, improved feature learning and mitigated vanishing gradient problem [16]. As the era of LLMs unfolded, the development of activation functions continued to evolve, resulting in over 400 documented activation functions [13]. Despite this progress, literature typically compares new activation functions against their immediate predecessors and usually using all the latest state-of-the-art configureation (bigger and bigger models) leading to a gap in the literature: there is no comprehensive comparison of multiple activation functions under consistent model sizes, furthermore, with the trend of increasing model sizes, the investigation of the impact activation functions have on smaller models is neglected. This research aims to address the aforementioned gap by exploring the impact of existing and novel activation functions on smaller-scale language models with approximately 10 million (10M) parameters.

For purpuses of the research, we will modify Hugging face implementations of GPTNEO [8] and ROBERTA [9] with selected existing activation functions and a novel Learnable GELU activation function, set the hyperparameters to amount to a total size of approximately 10M trainable parameters and evaluate them on the BabyLM evaluation pipeline [26]. More details on the setup provided in section 4.

This paper provides the evolution and bacground of activation functions in section 2 and the approach of the investiation in section 3. It describes the experimental setup using in section 4. The results are presented and analysed in section 5, and the discussion addresses performance outcomes and future research directions in section 6. Conslusion and responsible research considerations are presented in sections 7 and 8, respectively.

2 Background and related work

Transformers comprise multiple layers, each crucial 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) [10].

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 [5].

The famous paper "Attention is All You Need" [22] used the state-of-the-art activation function at the time, Rectified Linear Unit RELU. Since then, no significant improvements were made until the introduction of Gaussian Error Linear Unit 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 by comining the properties of RELU and SIGMOID function. It balances between linearity and non linearity, allowing it to handle both positive and negative inputs gracefully [11]. Despite these advancements, continuous innovation leads to alternatives like Gated GELU GEGLU, noted for its effectiveness [21], also used in last year's winner of the BabyLM challenge [19]. However, a recently published paper suggests a return to ReLU [15], 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 [15]. This also explains the initial move away from RELU, since the research on activation alternatives was done on models with the sizes of 100+ million parameters [21] [11]. Highlighting this find-

ing further motivates the need to investigate activation functions in smaller 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 trade-off between additional trainable parameters and performance. However, this was primarily studied in larger models. To our knowledge the only paper on adaptive activation functions in LLMs is by Rajanand et al. [17], which found that adaptive activation functions outperform static ones in text-to-text machine translation, a task also performed by 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. Hypothesizing that at smaller scales, the choice of activation function is crucial, having learnable parameters could be beneficial, while the added parameters from the function's parametrization remain relatively minimal compared to the total model size.

Kolmogorov-Arnold Networks (KAN) represent a recent development in neural network architecture, where activation functions are applied on edges instead of nodes [14]. 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 study, 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 piece-wise-defined polynomial function used in interpolation and approximation to create smooth curves through a set of points [2]. With a spline on each edge (see figure 1), 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. However, 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, addressing the gap in the current literature.

3 Approach

The transfomer structure (as introduced in section 2) allows the default activation function to be switched out with different activation functions for testing, enabling a direct comparison of their performance while keeping the rest of the architecture the same. To explore the effectiveness of various activation functions, we will modify the existing Hugging face implementations of GPTNEO [8] and ROBERTA [9].

3.1 Choosing activation functions

The activation functions to be evaluated are Rectified Linear Unit (RELU) [cite], Sigmoid Linear Unit (SILU) [cite], Gated SiLU with learnable parameters (SWISH) [6], Parametric ReLU (PRELU) [cite], Gaussian Error Linear Unit (GELU) (basline models), ADAPTABLE GELU and NO ACTIVATION FUNCTION. Additionally, the Kolmogorov–Arnold Networks KAN NETWORK [14] 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, since calculating $\Phi(x)$ is computationally expensive, it is instead calculated with the TANH approximation as:

$$\text{GELU}(x) = 0.5x\left(1 + \tanh\left(\sqrt{\tfrac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right)$$

[11].

This function will be used as a baseline for comparison with all other activations.

ReLU and PReLU

RELU was considered state-of-the-art at the time of the original transformer paper [22], but has since been surpassed by other activation functions.

$$\text{ReLU}(x) = \max(0, x)$$

RELU will be used as a baseline for comparison with PRELU and GELU. The comparison with GELU is motivated by the findings of *I. Mirzadeh et al.* [15], which suggests that the use of ReLU is acceptable as the impact of activation functions diminishes with increasing model size. The objective is to determine how much worse RELU peforms compared to GELU when used with smaller models.

PRELU is a variant of RELU that incorporates learnable parameters, allowing the activation function to adaptably learn the optimal slope for negative values.

$$\text{PReLU}(x) = \max(0, x) + a\min(0, x)$$

where a is a learnable parameter. This introduces x learnable parameters where x is the FFNs' intermediate size in the transformer. The objective is to evaluate whether adding a learnable parameter to RELU can enhance performance or increase the training time.

SiLU and Swish

SILU was evaluated on LLMs in the original GELU paper [11] but was found to perform worse than the GELU activation function. It is defined as

$$\text{SiLu}(x) = x \cdot \sigma(x), \text{ where } \sigma(x) \text{ is the logistic sigmoid.}$$

It will be used only as a baseline comparison for the *Swish* activation function, which is its adaptable counterpart with learnable parameters, aiding the objective of exploring the impact of adding learnable parameters to activation functions.

Swish is a self-gated activation function that was proposed by *Ramachandran et al.* [18]. It was implemented as proposed in a paper

$$\text{swish}(x) = x \cdot \text{silu}(\beta \cdot x)$$

where β is a learnable parameter. The objective is to evaluate whether the SWISH activation function outperforms the non adaptable SILU to evaluate the impact of adding learnable parameters to activation functions.

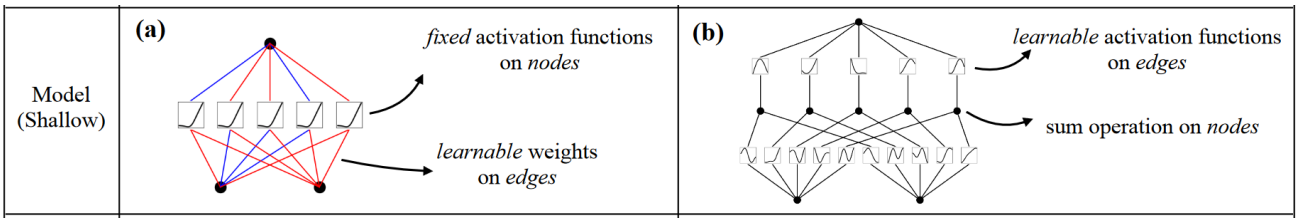

Figure 1: KAN vs MLP Z. Liu et al., "KAN: Kolmogorov-Arnold Networks." arXiv, May 02, 2024. doi: 10.48550/arXiv.2404.19756.

Adaptable GELU
The GELU activation function will be parameterized with learnable parameters. The implementation will be based on the PyTorch GELU implementation[cite] with TANH approximation, which, out of the box, does not support learnable parameters. This approach appears to be novel, as no prior research has been found that explores the impact of adding learnable parameters to the GELU activation function. The new GELU activation function adds a learnable parameter as a scaling factor and is defined as follows:

$$0.5 \cdot \beta \cdot \mathrm{x} \cdot \left(1.0 + \tanh\left(\sqrt{\frac{2.0}{\pi}} \cdot (\mathrm{x} + 0.044715 \cdot \mathrm{x}^3)\right)\right)$$

where β is a learnable parameter.

This activation functions adds a total of 2048 learnable parameters. The objective is to evaluate whether the ADAPTABLE GELU activation function outperforms the standard GELU activation function to evaluate the impact of adding learnable parameters to activation functions.

No Activation Function
The NO ACTIVATION FUNCTION will be used as a baseline for comparison with all other activation functions. This baseline will help determine the impact of using an activation function compared to not using one.

KAN-Network
The KAN NETWORK is a novel activation function that has shown promising results in the literature. The implementation of the KAN NETWORK is available by original authors, but has shown to be problematic and slow during our research. To address these issues, we will utilize an efficient-KAN implementation [1], which was the most popular optimization on GitHub of the original KAN paper at the time of this study. 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, we will set these parameters to 3 and 5, respectively. The configuration aims to balance performance and computational efficiency, ensuring that the KAN implementation is both effective and practical for the experiments. FFN layers from GPT-NEO and roBERTa both use MLPs in their implementation, which can be directly replaced with efficient-KAN implementation. As KAN network increases the amount additional trainable parameters in second linear layer by a factor of 15 **ToDo: Explain this caluclation**, the intermediate size for models with KAN NETOWRK was decreased to 256 to keep the number of parameters around 10M as for other models. The objective is to evaluate the performance of the KAN network compared to the other activation functions and to determine whether it is a viable alternative for LLMs. **ToDo: Talk about interpretabiloty of kan**

4 Experimental setup

4.1 Research questions

Through the experiment, we aim to answer the following research questions:

- *Is the choice of activation function relevant to the performance of smaller models with 10M parameters?* We compared the pre-trained baseline models and models with *SiLU* and *ReLU* activation functions and compared the results of evaluation on BabyLM evaluation pipeline [26].
- *How does the addition of learnable parameters to the activation function improve the performance of the model?* We modified static activation functions to include learnable parameters, pre-trained them, and evaluated them on the BabyLM evaluation pipeline [26].
- *Do FFNs using KAN-networks outperform FFNs using MLP networks?* We used efficient-kan implementation of KAN-networks [1], pre-trained them and evaluated them on BabyLM evaluation pipeline [26].

4.2 Dataset

We used the TinyStoreis dataset for pre-training. It's a dataset of short stories, that contains words that a typical 4-year-old would likely understand, generated by GPT-3.5 and GPT-4. It has been shown that they can be used to train LMs that are around 10M parameters and can still generate coherent stories. [7].

4.3 Models

We used Hugging face implementations of GPTNeoForCausalLM [8] and RobertaForMaskedLM [9] models. GptNEO is GPT2-based decoder model, while roBERTa is based on Google's BERT model from 2018. We used these models as baselines for our experiments. The hyperparameters used for the models can be seen in table 1. The change in intermediate size for KAN-MLP implementnation was made due to increased number of learnable parameters in the activation function and the need to keep the total number of parameters in the model approximately constant. The final parameter counts shown in table 2.

4.4 Evaluation Setting and Metrics

Evaluation pipeline

We used the BabyLM evaluation pipeline [26] to evaluate the models. The pipeline consists of three components: BLiMP, GLUE and SuperGLUE. BLiMP is a benchmark for evaluating the linguistic capabilities of language models, consisting of 17 metrics, each specific in syntax, morphology or semantics. Models were evaluated zero-shot, by comparing the probabilities of the sequences in a minimal pair, under the assumption that the acceptable sequence will be considered more likely than its unacceptable counterpart. The final score was computed as an average of those 17 metrics. [27] [28]. GLUE and SuperGLUE are benchmarks for evaluating the performance of language models on a variety of natural language understanding tasks. GLUE consists of 9 tasks, while SuperGLUE consists of 8 tasks. The final score was computed as the average of the scores 7/9 GLUE tasks and 3/8 SuperGLUE tasks [25] [24].

Statistical significance testing

TODO: Make this section more reader friendly.

To ensure the results are statistically significant, we trained each model 6 times with diffrent seeds. The following section justifies the choices made for the statistical significance testing, based and summarised from The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing [4].

Since the distribution of the test statistic is not known, we had to choose approaches from non parametric family of approaches. Which is split in two categories. Sampling based tests and sampling free tests. Sampling free tests hold less statistical power, therefore we decided to use sampling based tests.

Sampling based tests compensate for the lack of distribution information with resampling. They are computationally more expensive, but can be less effective for smaller sets of results. We used bootstrapping as a sampling based test. It is a resampling method that involves drawing samples with replacement from the original results. These samples are used to approximate the distribution of the statistic. We use that to calculate mean difference between the two models being compared and confidence intervals of the results. We used 10 000 samples for bootstrapping in our experiments. The exact implementation used, available in the replication package [TODO: ref].

4.5 Hardware

All the models were trained and evaluated on a single NVIDIA A100 GPU with 4 CPUs and 24GB of memory on DelftBlue cluster.

5 Results

ToDo: Fix the postition of all those tables bellow. All results of evaluations on BLiMP and GLUE datasets are shown in Table 5.

After performing bootstrap resampling for 10 000 samples, we calculated the mean difference and confidence interval to make each of the comparisons mentioned in section 4.

Comparison of Baseline (GELU) model with Adaptive GELU model showed that the mean difference is negative for both BLiMP and GLUE benchmarks, showing that the Adaptive GELU model performed slightly better than the Baseline (GELU) model. However the confidence intervals for both benchmarks contain zero, which means that the difference is not statistically significant. The results are shown in Table 3.

Boostrap comparing static activations ReLU and SiLU with their adaptive counterparts PReLU and Swish showed that the mean difference is positive for both BLiMP and GLUE benchmarks, showing that the static activation functions performed slightly better than the adaptive activation functions. However the confidence intervals for both benchmarks contain zero, which means that the difference is not statistically significant. The results are shown in Table 4.

The highest GLUE score was achieved by the BERT-KAN model with 63.65, the highest BLiMP score was achevied by the GPT-KAN model with 63.43. **ToDo: Further discuss after all statistical tests are done.**

Across all models, the training times were relatively consistent, with two outliers: KAN network models and ReLU models. The prolonged training time for the ReLU models might be attributed to the busy DelftBlue nodes, whereas the extended time for the KAN models is expected. We will compute the standard deviations for all models once we obtain all the results. Excluding the two outliers, the standard deviation in training times was 3 minutes and 33 seconds. **ToDO: Add results of other comparisons. (evaluations still running on DelftBlue)**

Parameter	GPT Neo	RoBERTa
Embedding Parameters		
Vocab Size	10,000	10,000
Hidden Size	512	512
Max Position Embeddings	512	513
Blocks (Attention & FFN)		
Number of Layers	2	2
Attention Types	global and local	global
Number of Attention Heads	4	4
Window Size	256	N/A
Intermediate Size	1024 (**256 for KAN-MLP**)	1024 (**256 for KAN-MLP**)

Table 1: Comparison of Parameters for GPT Neo and RoBERTa

Model Name	Number of Parameters
GPT with GELU (baseline)	9.0M
BERT with GELU (baseline)	9.0M
BERT with Learnable-GELU	9.0M
GPT with Learnable-GELU	9.0M
GPT with ReLU	9.0M
BERT with ReLU	9.0M
BERT with PReLU	9.0M
GPT with PReLU	9.0M
GPT with SiLU	9.0M
BERT with SiLU	9.0M
BERT with Swish	9.0M
GPT with Swish	9.0M
GPT with KAN MLP	11.0M
BERT with KAN MLP	11.0M

Table 2: Model Names and Number of Parameters

	Blimp	Glue
Mean difference	-0.135	-0.878
Confidence intervals	$[-1.16, 0.636]$	$[-2.674, 0.604]$

Table 3: Mean differences of combined GPT-Neo and RoBERTa scores for Basline (GELU) and Adaptive (GELU) models

6 Discussion

6.1 Implications

Our resutls show that choice of activation functions remains irrelavant even at smaller scales. All the comparisons in section 5 show that the differences are not statistically significant. This suggests that the choice of activation function does not have a significant impact on the performance of language models with 10 million parameters. This is in contrast to the findings of Mirzadeh et al. [15], who suggested that the impact of activation functions diminishes as the model size increases. Our results suggest that the impact of activation functions is negligiable even at smaller scales. Furthermore, the training durations of models employing various activation functions showed minimal differences, indicating that the selection of one activation function over another does not significantly impact training efficiency.

	Blimp	Glue
Mean difference	0.442	1.355
Confidence intervals	$[-1.85, 2.3]$	$[-0.35, 3.05]$

Table 4: Mean differences of combined GPT-Neo and RoBERTa scores for Static and Adaptive activation functions.

Model	Seed	Blimp	Glue	Time
BERT Baseline GELU	42	54.94	49.22	
BERT Baseline GELU	2	59.5	59.9	1h 41m 32s
BERT Baseline GELU	3	59.6	56.6	1h 41m 4s
BERT Baseline GELU	4	59.3	57	1h 41m 5s
BERT Baseline GELU	5	59.1	57.5	1h 42m 21s
GPT Baseline GELU	42	59.05	58.22	
GPT Baseline GELU	1	58	58.5	1h 43m 35s
GPT Baseline GELU	2	56.6	57.8	1h 45m 19s
GPT Baseline GELU	3	56.6	59.1	1h 41m 44s
GPT Baseline GELU	4	59.2	60.5	1h 40m 12s
GPT Baseline GELU	5	57.4	59.6	1h 42m 32s
BERT Adaptive GELU	42	59.4	57.1	1h 58m 39s
BERT Adaptive GELU	1	59.8	56.2	1h 45m 56s
BERT Adaptive GELU	2	59.3	59.7	1h 45m 55s
BERT Adaptive GELU	3	59	57	1h 45m 35s
BERT Adaptive GELU	4	58.2	57	1h 45m 56s
BERT Adaptive GELU	5	59.8	59.2	1h 45m 39s
GPT Adaptive GELU	42	56.8	60.2	1h 43m 41s
GPT Adaptive GELU	1	56.3	58.8	1h 41m 26s
GPT Adaptive GELU	2	57.4	58.7	1h 42m 53s
GPT Adaptive GELU	3	58.7	59.6	1h 41m 11s
GPT Adaptive GELU	4	56	59.2	1h 41m 27s
GPT Adaptive GELU	5	58.3	59.3	1h 41m 19s
Other models (not evaluated on multiple seeds yet)				
BERT ReLU	42	58.5	57.2	2h 24m 46s
BERT PReLU	42	57	57.6	1h 47m 27s
GPT ReLU	42	56.6	59.9	2h 17m 4s
GPT PReLU	42	59.2	56.1	1h 42m 28s
BERT SiLU	42	58.3	57.5	1h 45m 4s
BERT Swish	42	58	57.8	1h 43m 8s
GPT SiLU	42	56.3	59.9	1h 39m 35s
GPT Swish	42	53.7	57.6	1h 39m 22s
BERT KAN	42	56.69	63.65	2h 52m 11s
GPT KAN	42	63.43	48.8	3h 23m 35s

Table 5: All evaluation results

The comparison of results between GELU and the novel Learnable GELU further corroborates the findings of *Dror et al.* [4], which emphasize the importance of statistical significance analysis in evaluating various architectural decisions in language models. Our initial findings indicated an improved

Model Name	Training Time
GPT Baseline GELU	1h 42m 41s ± 1m 43s
BERT Baseline GELU	1h 41m 27s ± 32s
BERT Adaptive GELU	1h 47m 56s ± 5m 13s
GPT Adaptive GELU	1h 41m 59s ± 1m 02s
GPT-ReLU	2h 17m 4s
BERT-ReLU	2h 24m 46s
BERT-Swish	1h 43m 8s
GPT-Swish	1h 39m 22s
BERT-PReLU	1h 47m 27s
GPT-PReLU	1h 42m 28s
KAN2-GPT	3h 23m 35s
BERT-kan2	2h 52m 11s
GPT-SiLU	1h 39m 35s
BERT-SiLU	1h 45m 4s

Table 6: Mean training times with standard deviations ToDo: add other SDs and mean times when other evaluations complete.

performance of Learnable GELU, but subsequent analysis revealed these improvements to be statistically insignificant.

ToDo: Add more specific implications of other comparisions.

Models with KAN-Networks do seem to perform better than models with traditional MLPs, as shown by the highest scores achieved by the BERT-KAN and GPT-KAN models. This suggests that the KAN architecture could be beneficial for language models, even at smaller scales. However, the increased number of trainable parameters and increased training times is a significant drawback that needs to be considered.

6.2 Threats to validity

Internal validity examines certainty that the observed results are due to independent variable and not other factors. External validity concerns the extent to which the findings of the study can be generalized to contexts outside the study. Construct validity refers to the extent to which the the mearument tools are appropriate for the study.

Internal validity

To ensure that our results were not due to chance, we trained models using six different random seeds. While this sample size is relatively small, it was a necessary compromise given our resource constraints. To address this limitation, we employed bootstrapping with 10,000 samples to calculate confidence intervals and mean differences between models. Additionally, to verify that our findings were not model-specific, we utilized two distinct types of models: an encoder (roBERTa) and a decoder (GPT-Neo). Nevertheless, it is important to note that all models were trained on the same dataset, which may have introduced some bias. The train times could have been affected by busy delft blue nodes, but across models trained on multiple seeds, the training times were relatively consistent with standard deviation of only 9 minutes and 52 seconds accorss all models including the ones with KAN - Network.

Another potential threat to validity is that the implementation of adaptable activation functions added 2048 learnable parameters to each model, which could have influenced the results. However, given that 2048 parameters constitute a relatively small proportion in models with 9M, and the results were not statistically significant, this impact is likely minimal.

External validity

All the models were trained for only one epoch due to computational constraints, resulting in none of the models fully converging. This may have impacted the performance of the models and the significance of the results, as differences might be more pronounced or minimized with additional epochs. In a real-world scenario, models would typically be trained for more epochs. Additionally, the models utilized in this study are not the latest state-of-the-art, which may affect the generalizability of the results. Furthermore, the TinyStories dataset used for pre-training, which comprises only short fictional stories, may not be representative of the datasets typically used for production-level language models. **ToDO: talk about kan implementation**

Construct validity

The evaluation pipeline used in this study, BabyLM, primarily focuses on grammatical tasks, which may not fully capture the comprehensive capabilities of language models. However, considering the scope of this research, the tasks evaluated by the BabyLM evaluation pipeline are suitable. Since BabyLM is specifically designed for the evaluation of smaller language models, it aligns well with our study's focus. Thus, it is an appropriate tool for assessing the impact of activation functions on these models.

6.3 Future work

I hope this research will inspire further studies to give acctivation functions less attention. Before publishing new activation functions, researchers should conduct multiple runs and perform statistical significance tests to ensure the robustness of their findings.

The KAN architecture is a relatively new concept, and the implementation used in this study may not have been optimal. Although we utilized more efficient and optimized implementation as suggested by the original paper [14], the training process was less stable compared to baseline models (see Figure 2), and the models did not converge as expected. This instability may have impacted the performance of the KAN MODELS, suggesting that future research should investigate this issue further. Despite its potential, this promising direction concerning activation functions and its interpretability benefits were not fully explored in this study due to time constraints.

7 Conclusions and Future Work

TODO:

- brielfy repeat the RQs
- metion the the implications of results and tie them back to the RQs

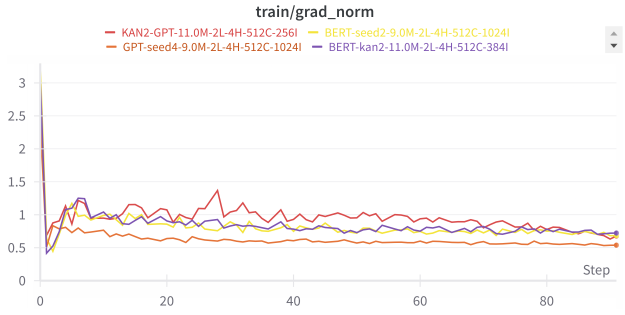

Figure 2: Gradient normalization during training of KAN models. The KAN models show higher variance in gradient norms compared to the baseline models. **ToDo: Fix readability**

8 Reponsible research

To prevent test set contamination, we pre-trained our models on datasets that were separate from those used for evaluation. This addresses an issue highlighted in recent works, where pre-training on test sets can artificially inflate performance metrics and call into question the validity of the results [20]. Ensuring distinct separation between training and evaluation datasets maintains the integrity of our findings and contributes to the reliability of our research.

In conducting this research, we ensured transparency and reproducibility by sharing the experimental setup under section 4. The datasets and models used are publicly available and can be found in the references. Furthermore, all the implemented code is available on GitHub? We adhered to scientific integrity by fabrication, and plagiarism, ensuring that everything reported accurately with proper citations.

Guided by the Netherlands Code of Conduct for Research Integrity, we incorporated principles of honesty, transparency, and responsibility, ensuring our research practices align with the highest standards. We followed the educational and normative framework from chapters 2 and 3 of the Code, emphasizing good research practices that promote a responsible research environment [12].

9 Acknowledgements

The research was conducted at TU Delft as the final thesis of the Bachelors of Computer Science and Engineering. We would like to thank for support of the project amazing project supervisor, supervising professors and the entire research research project group. Additionally, we thank the DHCP [3] for providing the neccesarry computational resrouces for the project.

References

[1] Blealtan. Efficient-kan, github repository, 2024. Accessed: 2024-06-03.

[2] Arindam Chaudhuri. B-splines.

[3] Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2, 2024.

[4] Rotem Dror, Gil Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia, Jul 2018. Association for Computational Linguistics, Association for Computational Linguistics.

[5] Shiv Ram Dubey, Satish Kumar Singh, and B. B. Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *arXiv preprint arXiv:2109.14545*, Jun 2022.

[6] Steffen Eger, Paul Youssef, and Iryna Gurevych. Is it time to swish? comparing deep learning activation functions across NLP tasks.

[7] R. Eldan and Y. Li. Tinystories: How small can language models be and still speak coherent english? *arXiv*, May 2023. Accessed: Nov. 01, 2023.

[8] Hugging Face. Gpt neo, 2024. Accessed: Jun. 03, 2024.

[9] Hugging Face. Roberta, 2024. Accessed: Jun. 03, 2024.

[10] Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space.

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

[12] KNAW, NFU, NWO, TO2-Federatie, Vereniging Hogescholen, and VSNU. Nederlandse gedragscode wetenschappelijke integriteit, 2018.

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

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

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

[16] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, volume 27, page 814, 2010.

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

[18] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *arXiv*, arXiv:1710.05941, Oct. 2017.

[19] 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.

[20] Rylan Schaeffer. Pretraining on the test set is all you need. *arXiv*, September 2023. doi: 10.48550/arXiv.2309.08632.

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

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

[23] 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.

[24] A. Wang et al. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv*, February 2020.

[25] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv*, February 2019. Accessed: Jan. 25, 2024.

[26] A. Warstadt, L. Choshen, A. Mueller, A. Williams, E. Wilcox, and C. Zhuang. Call for papers – the babylm challenge: Sample-efficient pretraining on a developmentally plausible corpus. *arXiv*, January 2023. Accessed: Apr. 15, 2024.

[27] A. Warstadt et al. Blimp: The benchmark of linguistic minimal pairs for english. *arXiv*, February 2023. Accessed: Jan. 22, 2024.

[28] Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell, editors, *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore, December 2023. Association for Computational Linguistics.