Adaptive Activation Functions

Does choice of activation function matter in smaller Langaunge Models?

Filip Ignijić

Supervisor: 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 21, 2024

Name of the student: Filip Ignijić
Final project course: CSE3000 Research Project
Thesis committee: prof. dr. Thomas Abeel, dr. Maliheh Izadi, prof. dr. 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 10M 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 [21] has revolutionized the AI landscape and enabled the development of commercial large language models (LLMs) like ChatGPT. However, as these models continue to grow in size, current trends forecast running out of high-quality data required for optimal performance [22]. 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. Therefore, this research aims to investigate 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 that can ensure privacy and be used for spellcheck, predictive typing, conversational assistance, etc.

The activation function in a neural network determines if a neuron should 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 [15]. As the era of LLMs unfolded, the development of activation functions continued to evolve, resulting in over 400 documented activation functions [12]. Despite this progress, literature typically compares new activation functions against their immediate predecessors and usually uses all the latest state-of-the-art configurations (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 purposes of the research, we will modify Hugging face implementations of GPTNEO [7] and ROBERTA [8] 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 [25]. More details on the setup are provided in section 4.

ToDo: Contributions.

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) [9].

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 [4].

The famous paper "Attention is All You Need" [21] 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 combining the properties of RELU and SIGMOID function. It balances between linearity and nonlinearity, allowing it to handle both positive and negative inputs gracefully [10]. Despite these advancements, continuous innovation leads to alternatives like Gated GELU GEGLU, noted for its effectiveness [20], also used in last year's winner of the BabyLM challenge [18].

However, a recently published paper suggests a return to ReLU [14], further confusing 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 [14]. 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 [20] [10]. Highlighting this finding 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. [16], 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 [13]. 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 [1]. 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 and in different domain, addressing the gap in the current literature.

3 Approach

The transformer structure (as introduced in section 2) allows the default activation functions 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 [7] and ROBERTA [8].

3.1 Choosing activation functions

The activation functions evaluated were Rectified Linear Unit (RELU) [1], Sigmoid Linear Unit (SILU) [10], Gated SiLU with learnable parameters (SWISH) [5], Parametric ReLU (PRELU) [2], Gaussian Error Linear Unit (GELU) (baseline models), ADAPTABLE GELU. Additionally, the Kolmogorov–Arnold Networks KAN NETWORK [13] 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 $\mathrm{GELU}(x) = x \cdot \Phi(x)$, 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:

$$\mathrm{GELU}(x) = 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right)$$
[10].

This function will serve as the baseline for comparing all other activation functions, with a particular focus on evaluating it against the novel Learnable GELU.

ReLU and PReLU

RELU was considered state-of-the-art at the time of the original transformer paper [21], but has since been surpassed by other activation functions.

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

RELU will be used for comparison with PRELU and GELU. The comparison with GELU is motivated by the findings of *I. Mirzadeh et al.* [14], which suggests that the use of ReLU is acceptable as the impact of activation functions diminishes with increasing model size. The objective is to assess the extent to which RELU underperforms compared to GELU when applied to 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.

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

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 [10] but was found to perform worse than the GELU activation function. It is defined as

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

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.* [17]. It was implemented as proposed in a paper

$$\mathrm{swish}(x) = x \cdot \mathrm{silu}(\alpha \cdot x)$$

where α 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.

[1] PyTorch Library, "torch.nn.ReLU," *PyTorch Documentation*, https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html

[2] PyTorch Library, *torch.nn.ReLU*, PyTorch Documentation, https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html.

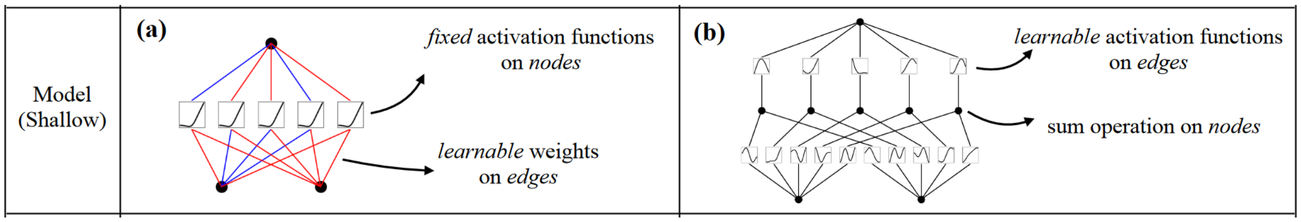
Model (Shallow)
(a)
fixed activation functions on *nodes*
learnable weights on *edges*
(b)
learnable activation functions on *edges*
sum operation on *nodes*

Figure 1: KAN vs MLP [13]

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:

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

where α is a learnable parameter.

This activation function 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.

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 the original authors but has shown to be problematic and slow during our research. Those issues were addressed in efficient-KAN[3] implementation, 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 and the order of piece-wise polynomials. 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 the KAN network increases the number of trainable parameters in the FFNs from 2 million to just over 8 million, the intermediate size for models with the KAN network was reduced to 256 to maintain approximately 2 million parameters in the FFNs. Although this reduction in intermediate size limits the expressiveness of the FFNs, the splines in the KAN network might compensate for this bottleneck. 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 MLPs in LLMs. Additionally, the KAN NETWORK offers the unique capability to extract symbolic representations of the learned activation functions from splines. It also provides the ability to retrain or further train specific parts of the network if necessary. Unfortunately, due to time constraints and the current implementation, this aspect will not be explored in this study. For further discussion, see Section 6.

[3]Blealtan, *Efficient-KAN, GitHub Repository*, 2024. Available at: https://github.com/Blealtan/efficient-kan (Accessed: 2024-06-03).

4 Experimental setup

4.1 Research questions

Through the experiment, we aimed 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 baseline models (with GELU activation) and models with *ReLU* activation function, which is GELUs predacessor.
- *How does the addition of learnable parameters to the activation function improve the performance of the model?* We modified static activation functions GELU, SILU and RELU to include learnable parameters and which gave us ADAPTIVE GELU, SWISH and PRELU respectively.
- *Do FFNs using KAN-networks outperform FFNs using MLP networks?* We used efficient-kan implementation[3] of KAN-networks.

All of the setups above were evaluated on the BabyLM evaluation pipeline [25].

4.2 Models

We used Hugging face implementations of GPTNEO [7] and ROBERTA [8] models. GPT NEO is a decoder model, while ROBERTA is encoder based model. We chose these models as they are well-established without and without the newer optimizations found in more recent other architectures, providing simplicity that is more suitable the scope of this project. Additionally, testing on encoder and decoder architectures increases the generalizability the research. GPT-Neo and Roberta also come in pre-trained form, however as we do not have the resources to train them to that extent, we pretrain on our own, smaller dataset so we have a baseline to compare to. The hyperparameters used for the models can be seen in table 1. The change in intermediate size for KAN-MLP implementation was made due to the 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 count for models using KAN was 11 million, whereas the parameter count for the other models was 9 million[4].

4.3 Activation functions

ReLU and SiLU are provided by used hugging face implementations and can be set via a configuration file. The rest of the activation functions were implemented as described in Section 3 and are available in the replication package[5].

4.4 Dataset

We used the TinyStories [6] dataset for pre-training. It is a dataset of short stories, with a total of 2.1 million samples with an average of 175.4 words per story, containing words that a typical 4-year-old would likely understand, generated by GPT-3.5 and GPT-4. We used them as it has been shown that training on reduced dataset complexity exhibits better natural language understanding than GPT-2 (125M) at a fraction of the training cost[5].

4.5 Evaluation Setting and Metrics

Evaluation pipeline

We used the BabyLM evaluation pipeline [25] to evaluate the models. The pipeline consists of three components: BLiMP, GLUE, and SuperGLUE. BLiMP is a benchmark used to evaluate the linguistic capabilities of language models. It consists of pairs of minimally different sentences, with one sentence in the pair containing a grammatical error. Performance is measured by the model's ability to correctly assign a higher likelihood to the grammatically correct sentence. [26] [27]. GLUE and SuperGLUE are benchmarks that assess the performance of language models across a range of tasks focused on text understanding and reasoning. The BabyLM pipeline integrates select tasks from both GLUE and SuperGLUE[5] [24] [23].

Statistical significance testing

To ensure the results are statistically significant, we trained each model 6 times with different seeds. The following section justifies the choices made for the statistical significance testing, based on and summarised from The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing [3].
Since the distribution of the test statistic is not known, we had to choose approaches from a nonparametric family of approaches, which is split into two categories. Sampling-based tests and sampling-free tests. Sampling-free tests hold less statistical power, therefore we decided to use sampling-based tests.

[4] These sections were written based on the guidelines provided by the research group supervisor and discussions with the other group members.

[5]Replication package: https://github.com/AISE-TUDelft/tiny-transformers/tree/main/code/filip

Sampling-based tests compensate for the lack of distribution information with resampling, which makes them computationally more expensive. We used bootstrapping as a sampling-based test. It is a resampling method that involves drawing samples with replacements from the original results. These samples are used to approximate the distribution of the statistics. We use that to calculate the means and 95% confidence interval of BLiMP and GLUE scores for each of the models. We used 10 000 samples for bootstrapping in our experiments. The exact implementation used, is available in the replication package[6].

4.6 Hardware

All the models were trained and evaluated on a single NVIDIA A100 or NVIDIA V100 GPU[6] with 4 CPUs and 24GB of memory on DelftBlue cluster [2].

Parameter	**GPT Neo**	**RoBERTa**
Embedding Parameters		
Vocab Size	10,000	10,000
Hidden Size	512	512
Max Pos. Embeddings	512	512
Blocks (Attn & FFN)		
Num. of Layers	2	2
Attn Types	global, local	global
Num. of Attn Heads	4	4
Window Size	256	N/A
Intermed. Size	1024 (256 for KAN-MLP)	1024 (256 for KAN-MLP)

Table 1: Parameters for GPT Neo and RoBERTa

5 Results

The evaluation results for the BLiMP and GLUE datasets are displayed in Figures 2 and 3. We calculated the mean scores and 95% confidence intervals for BLiMP and GLUE after performing bootstrap resampling for 10,000 samples.

For the first research question, which compares the baseline GELU activation function with its predecessor RELU, the results are inconclusive. The differences in scores between the two activation functions are statistically insignificant. With the GLUE means even being higer for both roBERTa and NEO models are marginally higher with the RELU activation function, the confidence intervals overlap, indicating no significant difference.

Regarding the second research question, which compares static activation functions with their adaptive counterparts, the findings are similarly inconclusive. The differences between static and adaptive activation functions are minor, and the overlapping confidence intervals suggest no statistical significance.

For the third research question, which was only evaluated on the GPT-NEO model due to time constraints, the results show a statistically significant difference. The KAN model

[6]For details, see Table **??**.

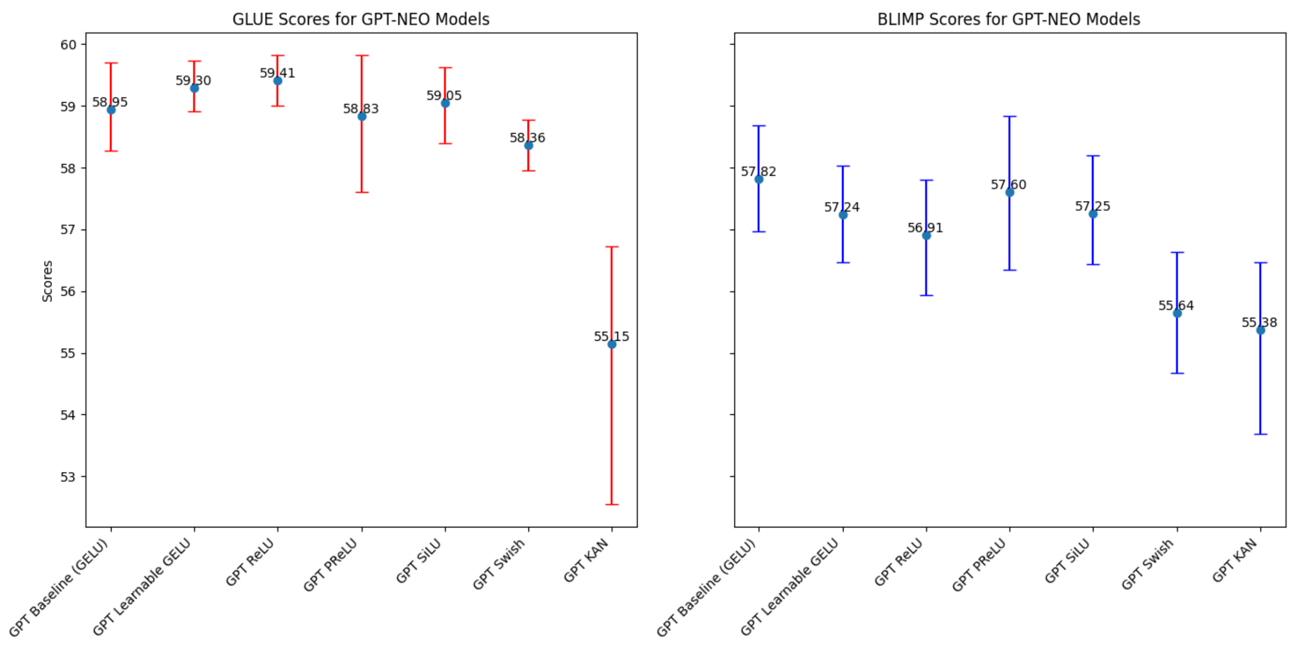

Figure 2: GLUE and BLiMP scores for GPT NEO models with 95% confidence intervals

performed worse than all other models on GLUE and worse than most models on BLiMP.

Training times across all models were consistent when using the same GPU. On the A100 GPU, training times were approximately 1 hour and 40 minutes, while on the V100 GPU, training times were around 2 hours and 20 minutes. Not all experiments were conducted on the A100 GPU due to time constraints and DelftBue [2] availability. The KAN network models were outliers, with training times averaging around 3 hours and 30 minutes on the A100 GPU and 4 hours and 50 minutes on the V100 GPU, which was expected. More details on the training times can be found in Table 3 and Table **??**.

6 Discussion

6.1 Implications

Our results indicate that the choice of activation functions remains irrelevant even at smaller scales. All comparisons in section 5 show statistically insignificant differences, with a few exceptions unrelated to the research questions. This suggests that the activation function choice does not significantly impact the performance of language models with 10 million parameters. This findings add onto findings of Mirzadeh et al. [14], which suggested the impact of activation functions diminishes as model size increases. Our results imply that this impact is negligible even at smaller scales, given our experimental setup and hyperparameters.

Regarding the second research question, the results show that parameterizing activation functions does not affect performance, explaning the lack of literature on adaptable activation functions in LMs. More interestingly, our results of comparison between ReLU and GELU challenge the conclusions of Hendrycks et al. [10], who claimed GELU's superiority over ReLU. Our results align with Mirzadeh et al. [14], who advocate for ReLU's return. While our research does not necessitate a return to ReLU, it also does not provide strong arguments for preferring GELU over ReLU or PReLU.

A likely reason for this outcome is that the models were trained for only one epoch, preventing them from converging, making the results rather inconclusive. See further discusion in Section 6.2 and Section 6.2.

Furthermore, the training durations of models using various activation functions showed minimal differences, indicating that the choice of activation function does not significantly impact training efficiency.

The comparison of results further supports the findings of Dror et al. [3], which emphasize the importance of statistical significance analysis in evaluating architectural decisions in language models. Some of our findings indicated better performance with certain activations, but subsequent analysis revealed these improvements to be statistically insignificant.

GPT models with KAN networks perform worse than those with traditional MLPs, as indicated by their lowest mean scores on GLUE and most BLiMP benchmarks. While this does not necessarily imply that KAN networks should be avoided, it suggests that the setup used in this research does not lead to improvement. Further research is needed to explore the potential of KAN networks in language models. See Section 6.3

6.2 Threats to validity

Internal validity examines the certainty that the observed results are due to independent variables 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 measurement tools are appropriate for the study.

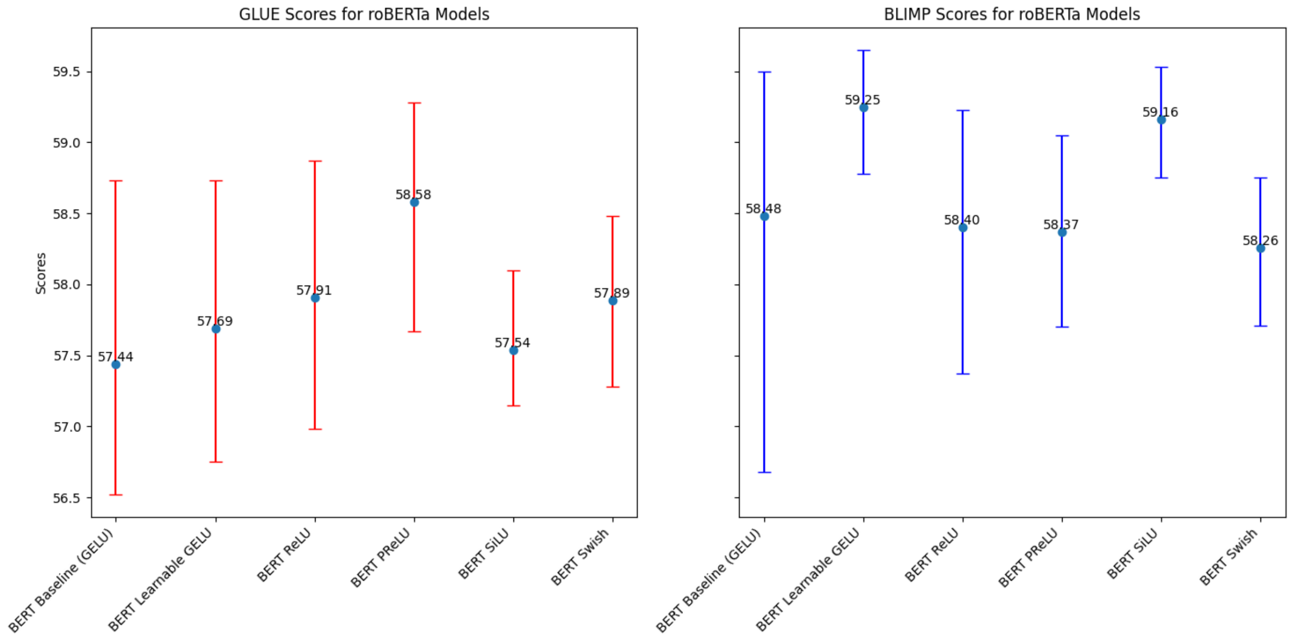

Figure 3: GLUE and BLiMP scores for roBERTa models with 95% confidence intervals

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 training times could have been affected by busy Delft Blue nodes, but across models trained on multiple seeds, the training times were relatively consistent, with a standard deviation of only 9 minutes and 52 seconds across all models, including those with KAN-Networks.

Another potential threat to validity is that the implementation of adaptable activation functions added 2,048 learnable parameters to each model, which could have influenced the results. However, given that 2,048 parameters constitute a relatively small proportion in models with 9 million parameters, 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.

The implementation of the KAN-Network[3] used in this study makes certain assumptions to optimize the originally proposed KAN implementation. One key change is the removal of the learnable scale from each activation function to improve efficiency. Additionally, the interpretability of KANs, a notable feature in the original design, is compromised in this implementation. This is because the critical L1 regularization, which was originally applied to input samples, has been moved to the weights. While this adjustment aligns better with common neural network practices and is compatible with the rest of the transformer architecture, it reduces the interpretability highlighted in the original paper and may also affect performance.

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 activation 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 [13], the training process was less stable compared to baseline models (see Figure 4), 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.

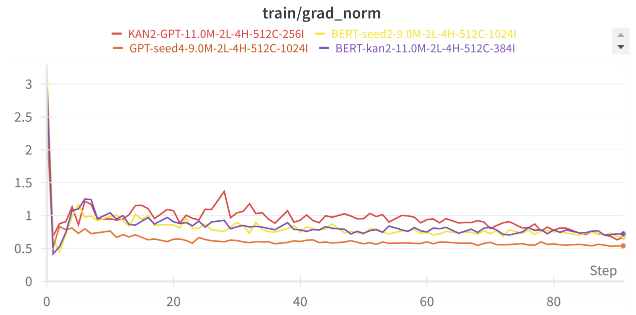

Figure 4: Gradient normalization during training of KAN models. The KAN models show higher variance in gradient norms compared to the baseline models. **ToDo: Fix readability**

7 Conclusions and Future Work

TODO:

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

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 [19]. 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 [11].

9 Acknowledgements

The research was conducted at TU Delft as the final thesis of the Bachelor of Computer Science and Engineering. We would like to thank you for the support of the project's amazing project supervisor, supervising professors, and the entire research project group. Additionally, we thank the DHPC [2] for providing the necessary computational resources for the project.

References

[1] Arindam Chaudhuri. B-splines. *arXiv:2108.06617*, 2021.

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

[3] 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.

[4] 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.

[5] Steffen Eger, Paul Youssef, and Iryna Gurevych. Is it time to swish? comparing deep learning activation functions across NLP tasks. *arXiv:1901.02671*, 2019.

[6] R. Eldan and Y. Li. Tinystories: How small can language models be and still speak coherent english? *arXiv 2305.07759*, May 2023. Accessed: Nov. 01, 2023.

[7] Hugging Face. Gpt neo, 2024. Accessed: Jun. 03, 2024 https://huggingface.co/docs/transformers/en/model_doc/gpt_neo.

[8] Hugging Face. Roberta, 2024. Accessed: Jun. 03, 2024 https://huggingface.co/docs/transformers/en/model_doc/roberta.

[9] Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv:2203.14680*, 2022.

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

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

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

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

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

[15] 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.

[16] A. Rajanand and P. Singh. Erfrelu: Adaptive activation function for deep neural network. *arXiv 2306.01822*, Jun 2023.

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

[18] 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.

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

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

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

[22] 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.

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

[24] 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.

[25] 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 2301.11796*, January 2023. Accessed: Apr. 15, 2024.

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

[27] 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.

10 Appendix

Table 2: Bootstraped means of GLUE and BLiMP with 95% confidence intervals

Model	Glue Mean	95% CI Glue	Blimp Mean	95% CI Blimp
BERT Baseline (GELU)	57.44	[56.52, 58.73]	58.48	[56.68, 59.5]
BERT Learnable GELU	57.69	[56.75, 58.73]	59.25	[58.78, 59.65]
GPT Baseline (GELU)	58.95	[58.27, 59.7]	57.82	[56.97, 58.68]
GPT Learnable GELU	59.30	[58.92, 59.73]	57.24	[56.47, 58.03]
BERT ReLU	57.91	[56.98, 58.87]	58.40	[57.37, 59.23]
BERT PReLU	58.58	[57.67, 59.28]	58.37	[57.70, 59.05]
GPT ReLU	59.41	[59.0, 59.82]	56.91	[55.93, 57.8]
GPT PReLU	58.83	[57.6, 59.82]	57.60	[56.34, 58.83]
BERT SiLU	57.54	[57.15, 58.1]	59.16	[58.75, 59.53]
BERT Swish	57.89	[57.28, 58,48]	58.26	[57.71, 58.75]
GPT SiLU	59.05	[58.4, 59.63]	57.25	[56.43, 58.2]
GPT Swish	58.36	[57.95, 58,78]	55.64	[54.68, 56.64]
GPT KAN	55.15	[52.54, 56.73]	55.38	[53.68, 56.47]

Table 3: Average pre-train times and standard devitations for all the models. *Note: * indicates models that were trained on V100s instead of A100s due to time constraints and DelftBlue cluster availability.*

Model	Avg. Train. Time
BERT Baseline (GELU)	1h 41m 27s ± 32s
BERT Learnable GELU	1h 47m 56s ± 313s
GPT Baseline (GELU)	1h 42m 41s ± 103s
GPT Learnable GELU	1h 41m 59s ± 102s
*BERT ReLU	2h 25m 23s ± 147s
BERT PReLU	1h 46m 11s ± 46s
*GPT ReLU	2h 20m 27s ± 203s
GPT PReLU	1h 41m 23s ± 40s
BERT SiLU	1h 43m 50s ± 268s
*BERT Swish	2h 20m 4s ± 1086s
GPT SiLU	1h 39m 7s ± 270s
*GPT Swish	2h 31m 4s ± 1032s
*GPT KAN	3h 58m 11s ± 2663s

Table 4: Pre-Train times, BLiMP and GLUE scores for all the modes. *Note: * indicates models that were trained on V100s instead of A100s due to time constraints and DelftBlue cluster availability.*

Model	**Seed**	**Blimp**	**Glue**	**Time**
BERT Baseline (GELU)	42	54.94	49.22	1h 41m 16s
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	1h 42m 44s
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 Learnable GELU	42	59.4	57.1	1h 58m 39s
BERT Learnable GELU	1	59.8	56.2	1h 45m 56s
BERT Learnable GELU	2	59.3	59.7	1h 45m 55s
BERT Learnable GELU	3	59	57	1h 45m 35s
BERT Learnable GELU	4	58.2	57	1h 45m 56s
BERT Learnable GELU	5	59.8	59.2	1h 45m 39s
GPT Learnable GELU	42	56.8	60.2	1h 43m 41s
GPT Learnable GELU	1	56.3	58.8	1h 41m 26s
GPT Learnable GELU	2	57.4	58.7	1h 42m 53s
GPT Learnable GELU	3	58.7	59.6	1h 41m 11s
GPT Learnable GELU	4	56	59.2	1h 41m 27s
GPT Learnable GELU	5	58.3	59.3	1h 41m 19s
*BERT ReLU	42	58.5	57.2	2h 24m 46s
*BERT ReLU	1	58	58.7	2h 22m 40s
*BERT ReLU	2	56.1	59.6	2h 25m 13s
*BERT ReLU	3	59.2	56.3	2h 25m 51s
*BERT ReLU	4	58.7	58.8	2h 29m 52s
*BERT ReLU	5	59.9	56.8	2h 23m 59s
*GPT ReLU	42	56.6	59.9	2h 17m 4s
*GPT ReLU	1	58.5	58.7	2h 16m 42s
*GPT ReLU	2	56.1	59.6	2h 21m 34s
*GPT ReLU	3	57.4	59.3	2h 20m 47s
*GPT ReLU	4	55	58.9	2h 25m 58s
*GPT ReLU	5	57.9	60.1	2h 20m 39s
BERT SiLU	42	58.3	57.5	1h 45m 4s
BERT SiLU	1	58.9	58.9	1h 41m 41s
BERT SiLU	2	59.8	57.4	1h 52m 26s
BERT SiLU	3	59.6	57.1	1h 41m 40s
BERT SiLU	4	59	57.3	1h 40m 43s
BERT SiLU	5	59.4	57	1h 41m 26s
GPT SiLU	42	56.3	59.9	1h 39m 35s
GPT SiLU	1	59.3	57.7	1h 37m 5s
GPT SiLU	2	55.9	58.5	1h 48m
GPT SiLU	3	57.9	59.7	1h 36m 55s
GPT SiLU	4	57.2	58.9	1h 37m
GPT SiLU	5	56.9	59.6	1h 36m 7s
BERT Swish	42	58	57.8	1h 43m 8s
*BERT Swish	1	57.7	57.3	2h 26m 13s
*BERT Swish	2	57.2	58.9	2h 27m 10s
*BERT Swish	3	58.9	56.8	2h 27m 48s
*BERT Swish	4	59	58.8	2h 28m 27s
*BERT Swish	5	58.8	57.7	2h 27m 39s
GPT Swish	42	53.7	57.6	1h 39m 22s
*GPT Swish	1	55	58.6	2h 19m 34s
*GPT Swish	2	56	59.2	2h 20m 51s
*GPT Swish	3	56.4	57.9	2h 21m 43s
*GPT Swish	4	57.2	57.8	2h 21m 30s
*GPT Swish	5	55.6	59	2h 23m 24s
BERT PReLU	42	57	57.6	1h 47m 27s
BERT PReLU	1	59.8	59.6	1h 45m 25s
BERT PReLU	2	58.2	58.1	1h 45m 30s
BERT PReLU	3	58.7	56.9	1h 46m 37s
BERT PReLU	4	58	59.5	1h 45m 54s
BERT PReLU	5	58.5	59.8	1h 46m 14s
GPT PReLU	42	59.2	56.1	1h 42m 28s
GPT PReLU	1	56.2	59.1	1h 41m 7s
GPT PReLU	2	55.5	60.5	1h 40m 40s
GPT PReLU	3	56.6	59.5	1h 41m 50s
GPT PReLU	4	59.6	58.2	1h 40m 48s
GPT PReLU	5	58.5	59.6	1h 41m 24s
GPT KAN	42	63.43	48.8	3h 23m 35s
*GPT KAN	1	52.7	56.5	3h 22m 36s
*GPT KAN	2	53.8	56.6	4h 54m 37s
GPT KAN	3	54.9	57	4h 52m 29s
GPT KAN	4	54.2	55.4	3h 23m 10s
GPT KAN	5	53.2	56.7	3h 52m 41s