KAN-MoE: Mixture of Experts with Kolmogorov-Arnold Networks

Adam Bielecki, Jan Małaśnicki, Jurand Prądzyński, Stanisław Świątkowski
{ap.bielecki, j.malasnicki, j.pradzynski, sa.swiatkowski}@student.uw.edu.pl

Supervisor: Jan Ludziejewski

Abstract

Kolmogorov-Arnold Networks (KAN) present a promising alternative to standard Multi-Layer Perceptrons (MLP), offering advantages in scalability and interpretability. Meanwhile, the Mixture of Experts (MoE) Transformer represents a state-of-the-art architecture in Large Language Models (LLMs), traditionally utilizing MLP-based networks as experts. In this study, we investigated the impact of replacing MLP experts with KANs. Our findings suggest that this modification generally results in inferior performance, indicating that KAN may not be beneficial in this context.

1 Introduction

The current machine learning (ML) revolution is being driven by Large Language Models (LLMs), which are the most powerful ML architecture known to date. Their widespread use makes it crucial to improve their performance, as it has a significant impact on various industries and academic fields. LLMs were made possible by the Attention mechanism introduced by (Vaswani et al., 2017). This, combined with a specialized pre-training technique, has redefined the way we perceive AI. LLMs are pre-trained by predicting the next token in a sequence of tokenized text. With the vast amount of text available on the Internet, overfitting on small datasets is no longer a concern. According to scaling laws (Hoffmann et al., 2022), (Kaplan et al., 2020), larger models and longer training lead to better results, meaning that our limitations are only based on computational resources. While waiting for more powerful hardware, part of the research has focused on optimising LLMs' efficiency. Mixture of Experts (MoE), which employs sparse activation approach, has been one of the most successful solutions to this problem (Fedus et al., 2022a). In MoE, instead of having one MLP after the Attention block, we have many MLP networks called Experts and use only some of them for each token. It has been found that MoE architectures outperform dense counterparts with the same number of active parameters (Fedus et al., 2022a).

Recently a new alternative to MLP has been released called Kolmogorov-Arnold Network (Liu et al., 2024). It is based on Kolmogorov-Arnold Theorem which established that, if f is a multivariate continuous function on a bounded domain, then f can be written as a finite composition of continuous functions of a single variable and the binary operation of addition. Specifically, for each smooth function $f : [0,1]^n \to \mathbb{R}$ the theorem asserts:

$$f(x) = f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q\left(\sum_{p=1}^{n} \phi_{q,p}(x_p)\right) \tag{1}$$

where $\phi_{q,p} : [0,1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$. In a sense, they showed that the only true multivariate function is addition, since every other function can be written using univariate functions and sum. KANs try to estimate Φ_q and $\phi_{q,p}$ with splines and use nodes only to add them together. Kolmogorov-Arnold Networks have learnable activations placed on the edges, distinguishing them from MLPs, where activations are located on nodes and are not learnable. In KANs, activation functions are weighted sums of third-degree B-splines, with the weights serving as parameters. By leveraging the Kolmogorov-Arnold theorem, KANs can more accurately approximate nonlinear functions, offering greater expressiveness compared to MLPs.

In this work, we verified that KANs do not replace the MLP architecture well as an expert in any aspect of working with large language models.

2 Related work

2.1 Mixture of Experts

The Mixture of Experts (MoE) technique is first proposed by (Jacobs et al., 1991); (Jordan and Jacobs, 2001) to deal with different samples, with independent expert modules. The first use of MoE in NLP tasks was made by (Shazeer et al., 2017). Their networks were based on LSTM recurrent neural networks (Hochreiter and Schmidhuber, 1997). In Attention based solutions MoE pioneers are GShard (Lepikhin et al., 2020) and Switch Transformer (Fedus et al., 2022b). GShard has learnable top-2 routing, while Switch has top-1 learnable routing. There are many works on improving MoE, which focus on the routing mechanism i.e. (Roller et al., 2021), (Zhou et al., 2022). Unlike them we make changes in individual expert networks.

2.2 Kolmogorov-Arnold Networks

The Kolmogorov-Arnold Theorem was introduced by (Kolmogorov, 1957). This Theorem was first used in designing a neural network by (Lin and Unbehauen, 1993), but their ideas were shown to be impractical by (Poggio, 2022). (Goyal et al., 2020) has worked with learnable activation functions. Taking learnable activations further Kolmogorov-Arnold Network (KAN) has been proposed by (Liu et al., 2024). They achieved promising results, outperforming standard MLPs, using fewer parameters.

3 KAN-MoE

In this study, we employ the Switch Transformer model (Jacobs et al., 1991) with various expert architectures. Illustration of this MoE architecture is presented on Figure 1. We introduce three KAN-based experts: KAN-KAN, Lin-KAN, and Lin-KAN-Lin. The schematic representations of these experts are shown in Figure 2. The implementation of the single KAN layer is sourced from the efficient-kan GitHub repository (Blealtan, 2024).

A single KAN layer with n inputs and m outputs is represented in a matrix form as follows:

$$\mathbf{y} = \begin{pmatrix} \phi_{1,1}(\cdot) & \phi_{1,2}(\cdot) & \cdots & \phi_{1,n}(\cdot) \\ \phi_{2,1}(\cdot) & \phi_{2,2}(\cdot) & \cdots & \phi_{2,n}(\cdot) \\ \vdots & \vdots & & \vdots \\ \phi_{m,1}(\cdot) & \phi_{m,2}(\cdot) & \cdots & \phi_{m,n}(\cdot) \end{pmatrix} \mathbf{x}, \quad (2)$$

where this should be interpreted as an application of functions to arguments rather than matrix-vector multiplication:

$$y_j = \sum_{i=1}^{n} \phi_{j,i}(x_i), \qquad j = 1, \cdots, m. \quad (3)$$

Each $\phi_{i,j}$ is defined by the equation:

$$\phi(x) = w_b \operatorname{silu}(x) + w_s \operatorname{spline}(x), \quad (4)$$

where w_b and w_s are trainable scalars, and $\operatorname{spline}(x)$ is parameterized as a linear combination of B-splines such that:

$$\operatorname{spline}(x) = \sum_i c_i B_i(x), \quad (5)$$

where the c_i's are trainable parameters.

4 Experimental Set-up

In this section, we explain the purpose and meaning of the performed experiments. All our models are decoder-only, and we compare pre-trained models without fine-tuning. The models are trained on the C4 dataset (Habernal et al., 2016) and evaluated on the next-token prediction task using cross-entropy as the loss function. Due to computational resource constraints, we train on relatively small model sizes with word embedding size equal to $d_{model} = 256$ and our baselines having 3M active parameters. MoE architecture has 8 experts. In all experiments we use a batch size of 128 and sequence length of 128. We use cosine learning scheduler with 1% warm up and 10% final learning rate fraction.

In the initial phase of our research, we decided to compare the performance of vanilla transformer with MLP and KAN as feed forward in their basic forms. This decision was driven by the need to establish a robust comparative baseline for further analyses. Understanding the fundamental properties of these models also enabled us to better tune their parameters and optimize their performance. Additionally, this comparison served as a sanity check, allowing us to evaluate how much better KANs perform compared to MLPs in standard applications. Thus, the analysis of vanilla transformer with MLP and KAN as a feed forward constituted a fundamental step in our research methodology, facilitating a more precise and reliable evaluation of the implemented solutions.

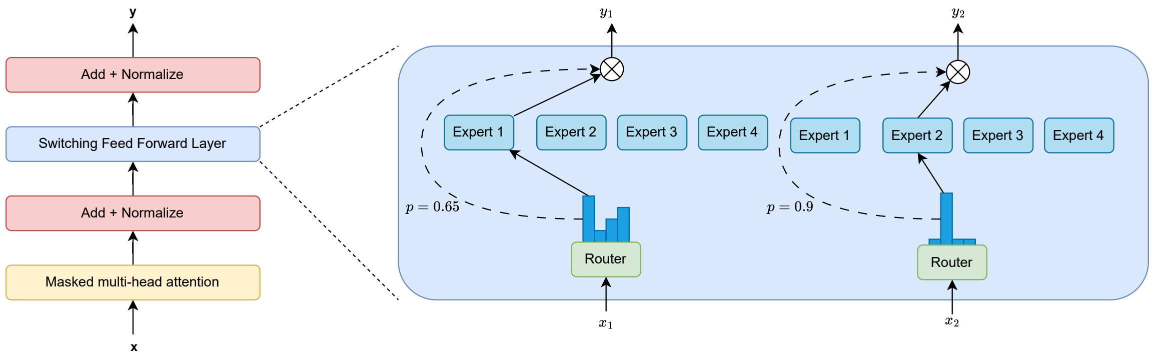


Figure 1: Illustration of the Switch Transformer architecture.

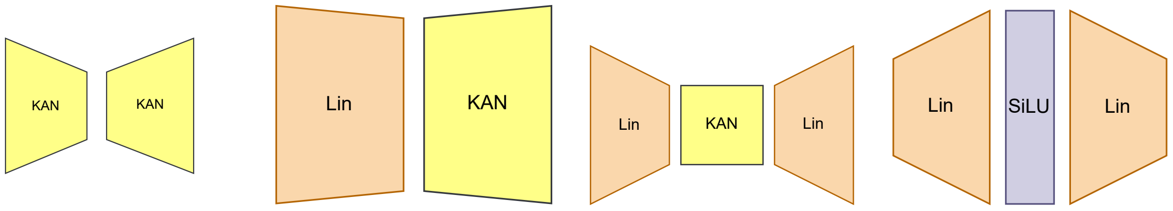


Figure 2: Visual representation of various KAN-based expert architectures. From the left: KAN-KAN, Lin-KAN, Lin-KAN-Lin. Last one is standard MLP.

4.1 Parameters Matched

The parameters matched approach is to maintain the total number of trainable parameters in the model. This ensures that differences in performance are due to architectural changes rather than variations in model size. According to (Liu et al., 2024), KAN architecture offers faster matching of the activation function to the modeled problem.

This setup gives the opportunity to fix the size of the model placed in VRAM. One of the problems of MoE architectures is that experts take up much more space than a vanilla transformer with one MLP layer, so it has to be distributed on more units. Placing the entire LLM on a small number of computing units brings many benefits - it reduces data access time, reduces the risk of parameter inconsistency and eliminates the computational costs of distributing variables between units.

This experiment allows to check whether, apart from efficiency, the KAN-MoE architecture can compete with current solutions in terms of memory consumption and latency.

It is also noteworthy that parameter matched KANs are FLOPs matched as well, meaning that during their training they utilized the same number of floating point operations per second (FLOPS) as a baseline MLP model. For the same input and output dimensions, a KAN network has 10 times more parameters and uses 10 times more FLOPs compared to MLP. A detailed estimation of the number of FLOPs for each network type is provided in the appendix A. This matching ensures a fair comparison in terms of computational complexity and resource utilization, further validating the robustness of our comparative analysis.

Expert type	**Layers sizes**
MLP	$256 \times 1024 \times 256$
KAN-KAN	$256 \times 102 \times 256$
Lin-KAN	$256 \times 186 \times 256$
Lin-KAN-Lin	$256 \times 205 \times 205 \times 256$

Table 1: Layer sizes for different expert types, such that each one is parameter matched.

In order to have parameter matched architectures, for every KAN-based expert we set appropriate layer sizes. We present them in Table 1.

4.2 Other FLOPS Matched

KAN in test tasks such as Feynman datasets provides faster and more accurate matching in the initial phases of training, thanks to a more flexible non-linearity modeling formula (Liu et al., 2024). We want to use this test in a larger architecture setting that will not require long training to obtain relevant performance.

The core idea is to replicate the size and structure of the linear layers in the KAN to match those in the MLP, while leveraging the superior adaptability of KAN. In this approach, KANs are designed to use ten times the number of trainable parameters compared to MLPs. However, to maintain computational efficiency, the KANs are trained for one-tenth the duration, using one-tenth the number of tokens. This strategy is grounded in the hypothesis that KAN parameters can adapt more rapidly to the problem at hand than MLP parameters, thereby potentially enhancing performance with significantly shorter training periods.

The premise of this approach aligns with given compute resources, the model can be equivalently optimized by training a larger model for a shorter period or a smaller model for a longer period. By increasing the number of parameters in the KAN by a factor of ten and reducing the training time proportionally, this method aims to achieve a balance where the computational cost remains consistent with traditional training methods.

4.3 Overparameterised architecture

Conducting an upper bound experiment is a crucial step in evaluating the potential performance limits of architectures as KAN-MoE. This type of experiment involves training the model under idealized conditions to understand the theoretical maximum performance achievable with the given architecture within the capabilities of the compute resources we worked on.

To achieve the highest performance, we conducted experiments using overparameterised models, training on the same number of tokens as the MLP architecture across two settings. First, we trained models around the compute-optimal configuration as described by (Kaplan et al., 2020) for state-of-the-art MLP. In the second setting, after failing to achieve MLP performance, we overtrained 10 times to further explore the potential advantages of the models.

These approaches, however, present a biased comparison of which architecture is superior. In this setup, the KAN-MoE model has ten times the number of parameters allocated to each expert. This results in an architecture that utilizes significantly more computational resources and memory.

5 Results and Discussion

5.1 Hyperparameters fine-tuning

The learning rate significantly affects how quickly the model reaches the optimal solution. A well-chosen learning rate can dramatically reduce training time, allowing the model to achieve optimal parameters more efficiently. The learning rate can affect the model's ability to generalize. Proper tuning helps balance fitting the training data and maintaining robustness to invisible tokens, thus reducing over-fitting (Jin et al., 2023). For these reasons, we have prepared a grid of potential learning rates to find the best one for flop matching experiments. The vanilla transformers and MoE models we found learning rates based on the lowest final loss function. Our experiments are presented in Figures 3 and 4.

5.2 FLOPS matched

Figures 5 and 6 contain plots for runs in which respectively vanilla and Mixture of Experts (MoE) models were FLOPS matched. Architectures Lin-KAN, Lin-KAN-Lin, and KAN-KAN are not only FLOPS matched but also parameter matched according to Appendix A. Runs with names Lin-KAN-short_train and KAN-KAN-short_train have the experts that have more parameters than the MLP baseline but were trained on fewer tokens so that the whole training process is FLOPS matched.

When considering vanilla models, we examined Lin-KAN-fat and KAN-KAN-fat configurations, characterized by fewer transformer blocks with a higher number of parameters per feed forward layer, thereby maintaining the overall parameter count of the model. These configurations did not yield favorable results, hence they were not used in our MoE setup.

From the results presented in Figure 6, it can be observed that none of the KAN-based experts outperform the baseline MLP. Among the KAN-based architectures, Lin-KAN shows the best performance. The two runs with a greater number of parameters but shorter training time perform worse than the others.

Our experiments provide compelling evidence that the resemblance of a block to a Multi-Layer Perceptron (MLP) correlates positively with improved performance. Specifically, we observe that blocks containing a higher number of linear operations consistently achieve better results across various evaluation metrics.

5.3 Overparamterised and Overtrained

Figures 7 and 9 depict results from experiments where vanilla and Mixture of Experts (MoE) models were overparameterised. Specifically, architectures KAN-KAN and Linear-KAN have approximately 10 and 5 times more parameters, respectively, compared to MLP.

Figure 8 presents results from overtrained models, indicating that all models were trained for 10 times longer than usual. The experiment includes KAN-KAN and Lin-KAN architectures across default, Overparameterised, and fat scenarios. As with normal training length, fat-KAN is the worst architecture in overtrain. With that we decided not to consider it in MoE setting. Overparameterised Vanilla Transformer doesn't achieve SOTA performance. When trained for 10 times longer Linear-KAN architecture achieves baseline's loss, beating overparameterised KAN-KAN. Longer training doesn't close the gap to the baseline for any parameter-matched architecture.

In MoE setting overparameterised networks achieve baselines performance without overtrain. This suggests that KAN-based networks are more compatible with MoE, than Vanilla setting. With overtraining (Figure 10) overparameterised Lin-KAN-MoE has high chances for achieving higher performance than MoE with MLP experts. Parameter-Matched architectures don't improve, with longer training, when compared to the baseline.

6 Conclusions

The use of Kolmogorov Arnold Networks (KAN) as experts in the Mixture of Experts (MoE) architecture resulted in worse performance (measured by loss function decrease) compared to the standard approach employing fully connected layers. This was consistently observed across all tests, including overparameterised scenarios that favored KAN-MoE in terms of computational resources. In previous studies, KAN has been evaluated on science-related tasks, such as solving partial differential equations or modeling mathematically complex functions within small networks. These tasks differ significantly from those typically addressed by large language models (LLMs), which may contribute to the inferior performance of KAN-MoE in our experiments.

It is important to note that our experiments were conducted on small-scale language models. It remains possible that architectures with larger model dimensions (e.g., $d_{model} = 512$ and above) might exhibit improved performance when utilizing KAN. Further research is required to explore this potential and to better understand the applicability of KAN in the context of LLMs.

References

Blealtan. 2024. An efficient implementation of kolmogorov-arnold network. https://github.com/Blealtan/efficient-kan.

William Fedus, Jeff Dean, and Barret Zoph. 2022a. A review of sparse expert models in deep learning. *Preprint*, arXiv:2209.01667.

William Fedus, Barret Zoph, and Noam Shazeer. 2022b. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Preprint*, arXiv:2101.03961.

Mohit Goyal, Rajan Goyal, and Brejesh Lall. 2020. Learning activation functions: A new paradigm for understanding neural networks. *Preprint*, arXiv:1906.09529.

Ivan Habernal, Omnia Zayed, and Iryna Gurevych. 2016. C4Corpus: Multilingual web-size corpus with free license. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 914–922, Portorož, Slovenia. European Language Resources Association (ELRA).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

Robert Jacobs, Michael Jordan, Steven Nowlan, and Geoffrey Hinton. 1991. Adaptive mixture of local expert. *Neural Computation*, 3:78–88.

Hongpeng Jin, Wenqi Wei, Xuyu Wang, Wenbin Zhang, and Yanzhao Wu. 2023. Rethinking learning rate tuning in the era of large language models. *Preprint*, arXiv:2309.08859.

Michael Jordan and Robert Jacobs. 2001. Hierarchical mixtures of expert and the em algorithm. *Neural Computation*, 6.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Andrei Nikolaevich Kolmogorov. 1957. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In Doklady Akademii Nauk, volume 114, pages 953–956.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *Preprint*, arXiv:2006.16668.

Ji-Nan Lin and Rolf Unbehauen. 1993. On the Realization of a Kolmogorov Network. *Neural Computation*, 5(1):18–20.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. 2024. Kan: Kolmogorov-arnold networks. *Preprint*, arXiv:2404.19756.

Tomaso A. Poggio. 2022. How deep sparse networks avoid the curse of dimensionality: Efficiently computable functions are compositionally sparse.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *Preprint*, arXiv:1701.06538.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Preprint*, arXiv:1706.03762.

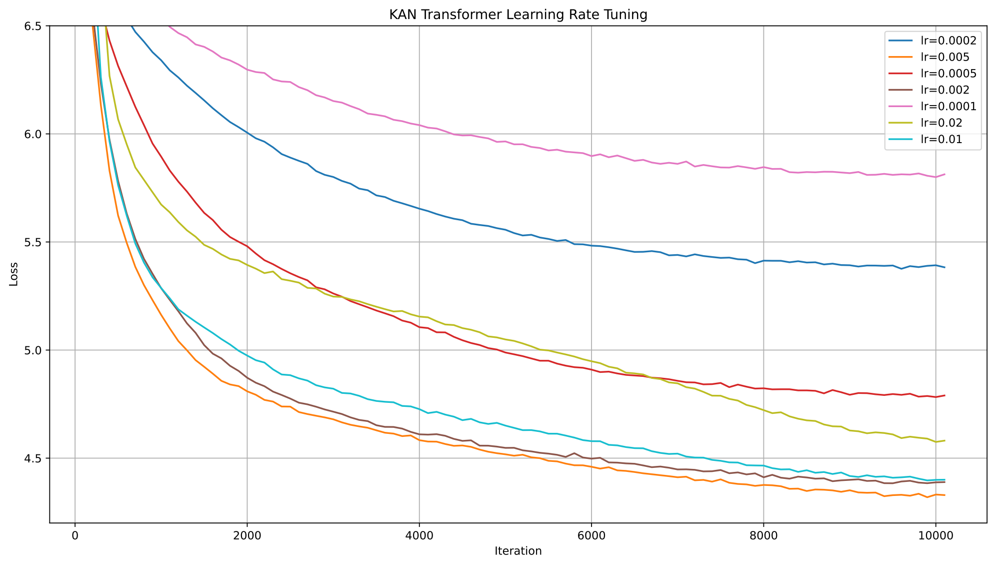
Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing. *Preprint*, arXiv:2202.09368.

A KAN FLOPs estimation

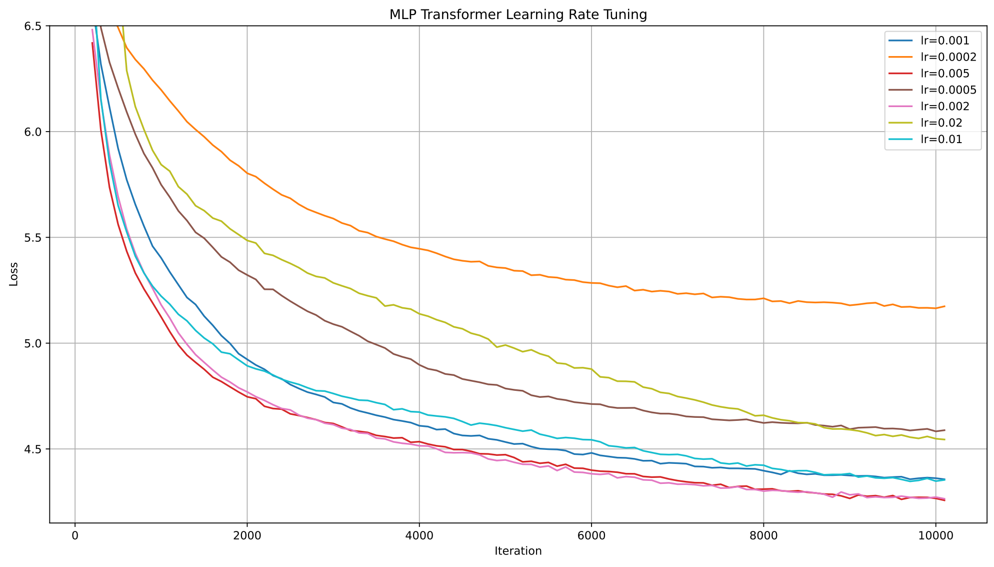
[appA] The most computationally intensive component of KANs involves calculating

$$\phi(x) = w_b b(x) + w_s spline(x) \tag{6}$$

where $b(x)$ and $spline(x)$ functions each require $O(n_{in})$ FLOPs. The weight matrix w_b has dimensions $n_{in} \times n_{out}$ and w_s has dimensions $n_{in} \times n_{out} \times (grid\_size + spline\_order)$. For our specific case $grid\_size = 5$ and $spline\_order = 3$ making the linear operations collectively require $9 \times n_{in} \times n_{out}$. This is nine times more than a traditional MLP. Additionally, KANs involve several operations with $O(n_{in})$ FLOPs. Thus, we approximate that for the same n_{in} and n_{out}, KANs are approximately ten times slower than classical MLPs. This approximation is consistent with the original KAN paper (Liu et al., 2024).

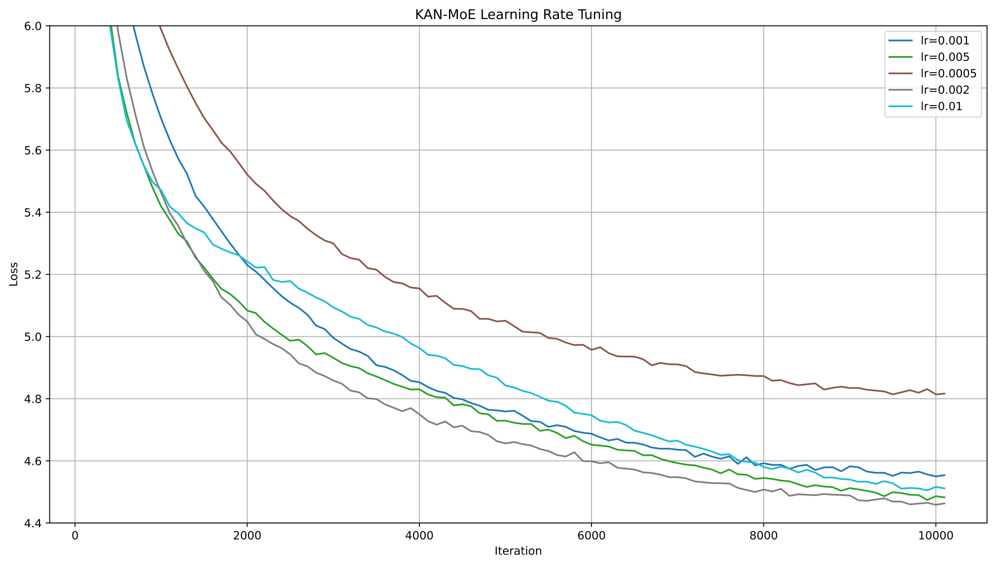


(a) Grid of learning rates for KAN layer instead of feed forward layer.

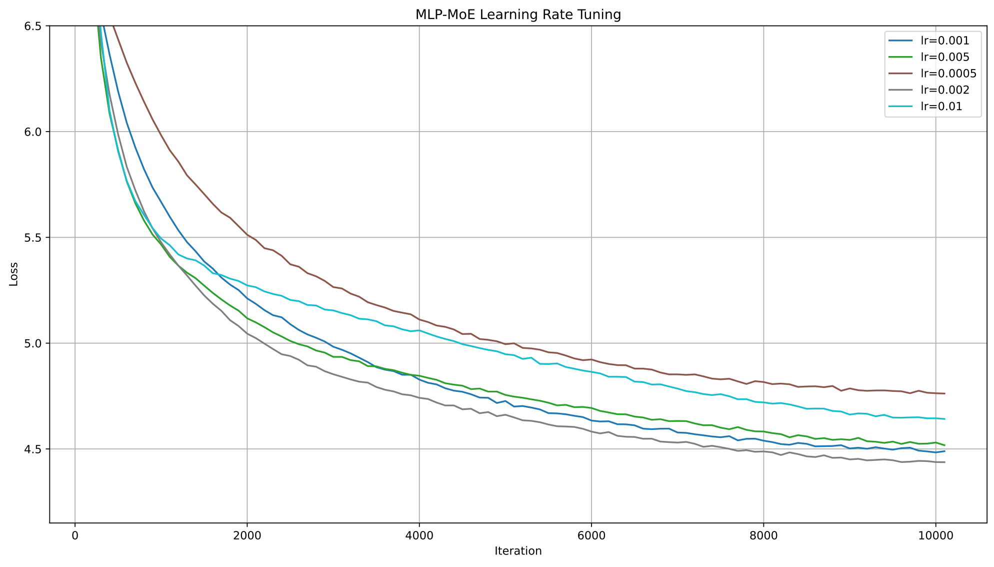


(b) Grid of learning rates for standard feed forward layer.

Figure 3: Potential learning rates for vanilla transformer in our setup.



(a) Grid of learning rates for KAN experts instead of MLP experts.



(b) Grid of learning rates for standard Mixture of Experts architecture.

Figure 4: Potential learning rates for MoE models in our setup.

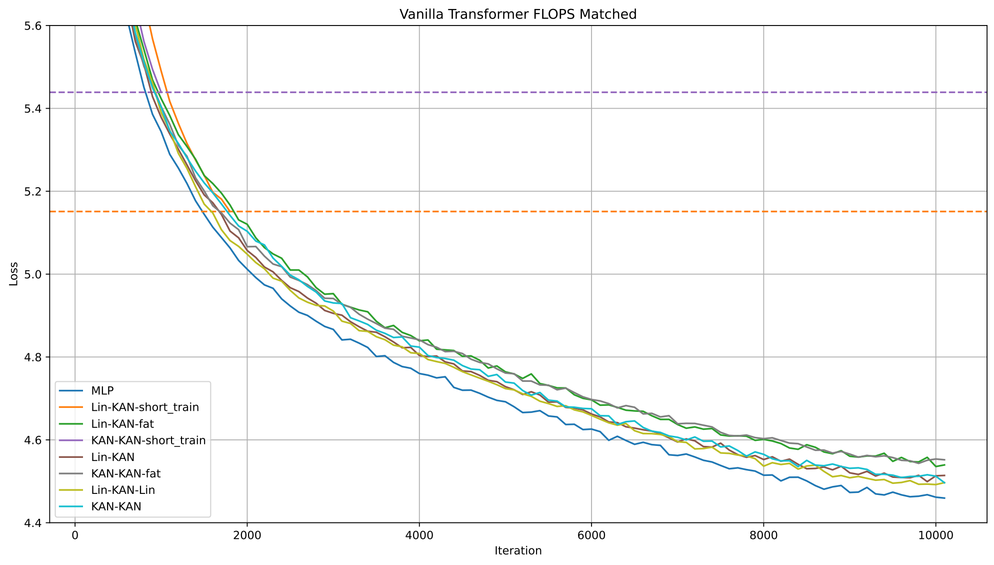


Figure 5: Training loss over iterations for vanilla architectures with FLOPS matched configuration. The compared feed forward layers include MLP, Lin-KAN-short_train, KAN-KAN-short_train, Lin-KAN, Lin-KAN-Lin, and KAN-KAN.

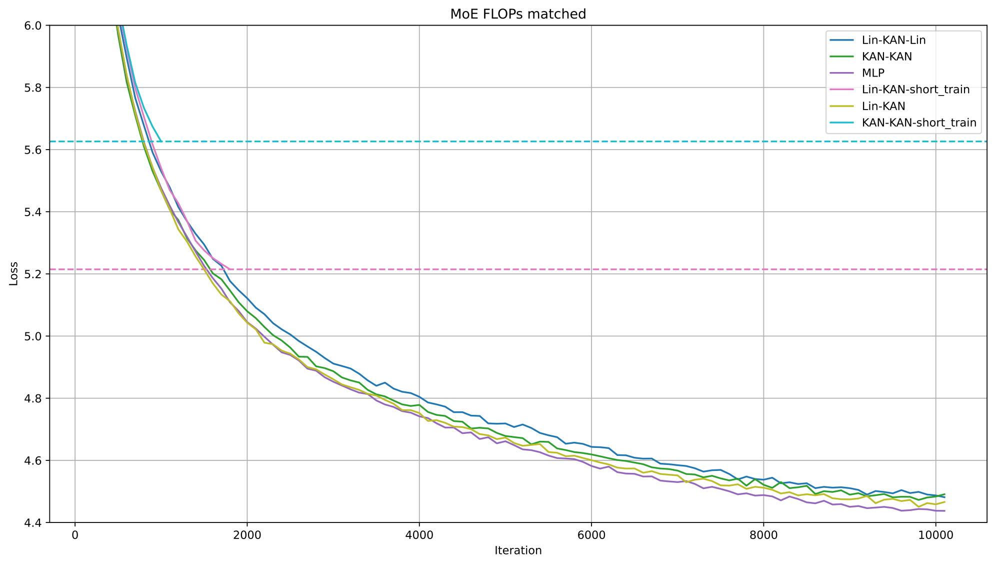


Figure 6: Training loss over iterations for various expert architectures with FLOPS matched configuration. The experts compared include MLP, Lin-KAN-short_train, KAN-KAN-short_train, Lin-KAN, Lin-KAN-Lin, and KAN-KAN.

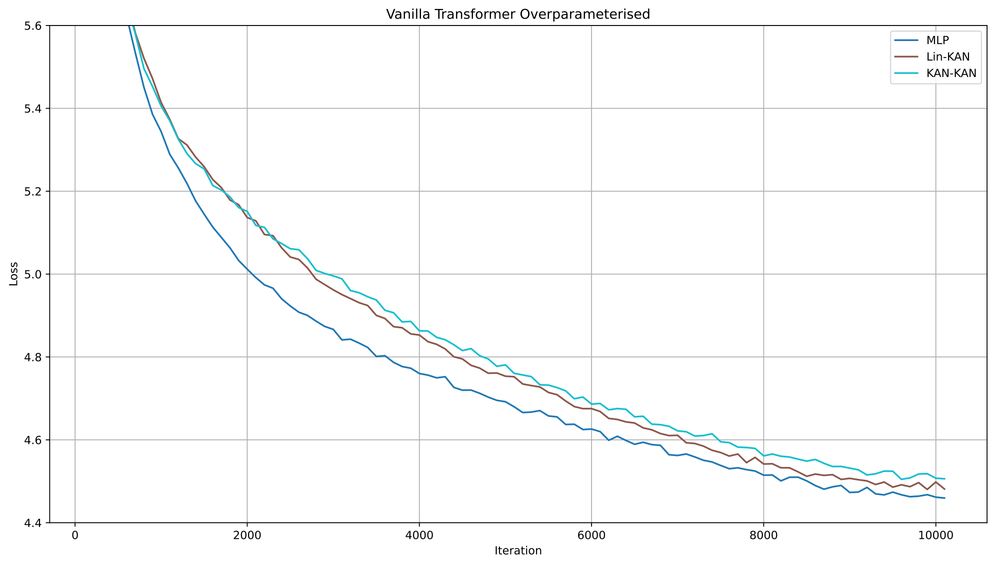


Figure 7: Training loss over iterations for vanilla architectures with overparameterised configuration. The compared feed forward layers include MLP, Lin-KAN, Lin-KAN-overparameterised, Lin-KAN-fat, KAN-KAN, KAN-KAN-overparameterised, KAN-KAN-parametrized.

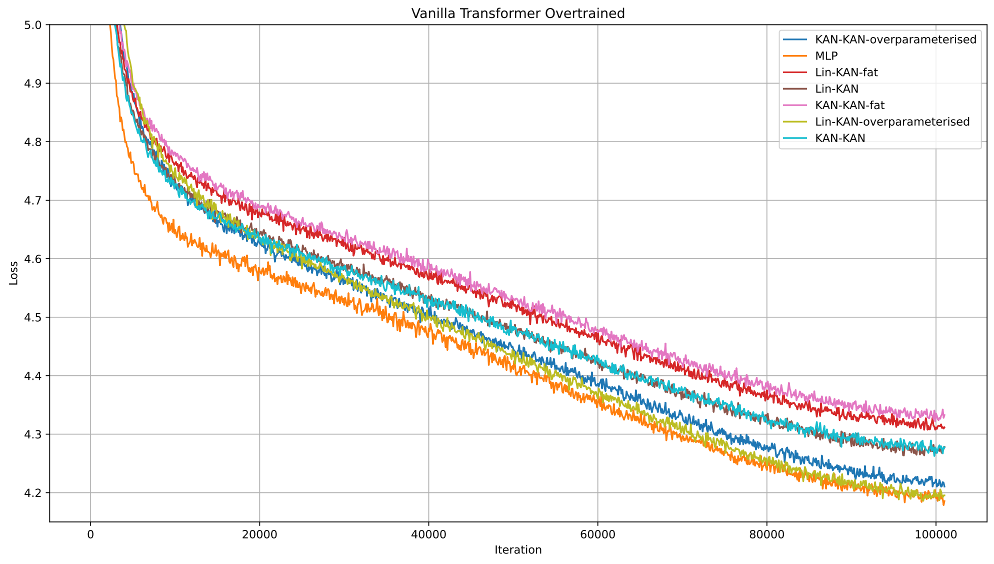


Figure 8: Training loss over iterations for vanilla architectures with overtrain configuration. The compared feed forward layers include MLP, Lin-KAN, and KAN-KAN.

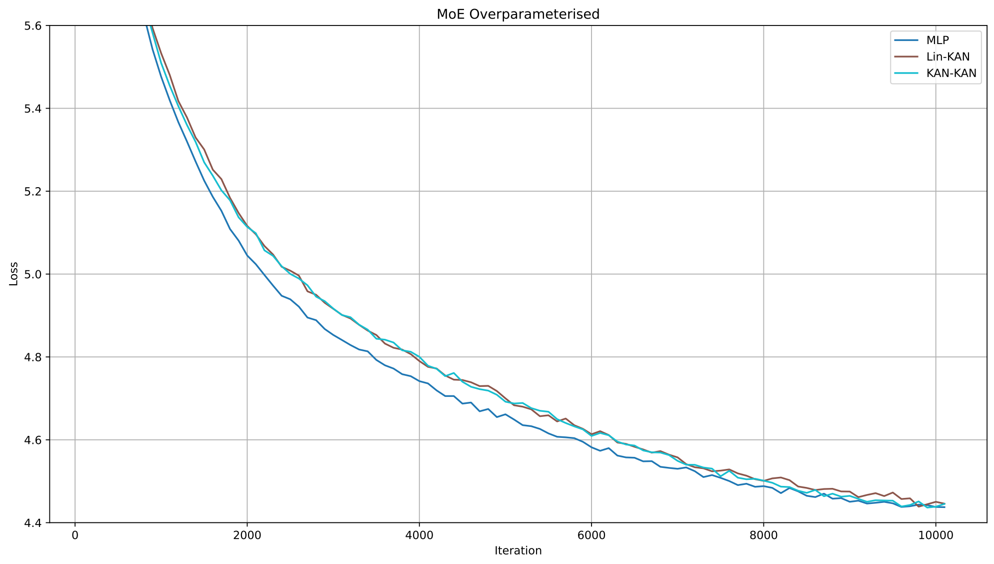


Figure 9: Training loss over iterations for various overparameterised expert architectures. The experts compared include MLP, Lin-KAN, and KAN-KAN.

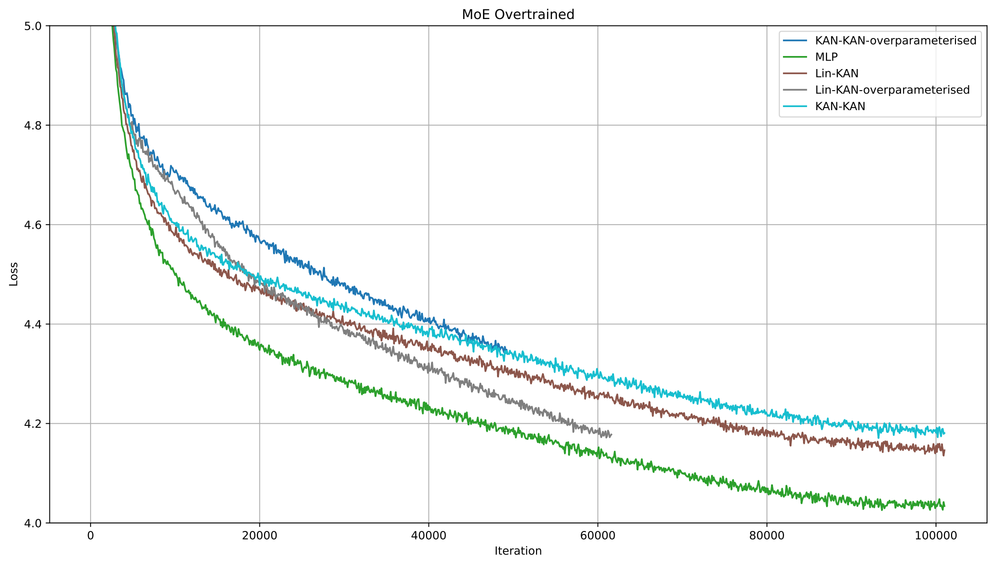


Figure 10: Training loss over iterations for MoE architectures in overtrain setup. The experts compared include MLP, Lin-KAN, and KAN-KAN.