KAN-based Distillation in Language Modeling

Stanford CS224N Custom Project

Nick Mecklenburg
Department of Computer Science
Stanford University
nmecklen@stanford.edu

Abstract

Kolmogorov-Arnold Networks are a newly proposed challenger to the multilayer perceptrons so pervasive throughout many architectures in modern day deep learning. Their dynamics and applicability to the domain of language modeling is as of yet unexplored, so in this paper we study their applicability to model distillation, seeing if we can replace MLPs in a pretrained transformer language model with smaller but just-as-performant KANs for a reduction in parameter count. We discover KANs are capable of learning the complex and high-dimensional distribution of natural language, though they are a poor choice for MLP-approximation compared to the simple baseline of distilling into smaller MLPs.

1 Key Information to include

- Mentor: Shikhar Murty
- External Collaborators (if you have any): None.
- Sharing project: No.

2 Introduction

The multilayer perceptron, or MLP, is a fundamental building block in many state-of-the-art neural network architectures today, used across a wide suite of modalities and use cases, with theoretical roots tracing back to the early origins of neural networks. Liu et al. (2024) recently challenged MLPs in their introduction of Kolmogorov-Arnold Networks, or KANs, framed as a more interpretable and powerful alternative that often learns the same task in orders of magnitude fewer parameters. That KANs are purportedly so parameter efficient begs the question of whether they can be used for distillation in more complex networks, used to compactly approximate equivalent MLPs. This study endeavors to explore the feasibility of KAN-based distillation in language modeling.

The original KAN paper explored many toy use cases – low-dimensional data generated from simpler functions or compositions of functions that lend themselves well to the Kolmogorov-Arnold representation theorem (Givental et al. (2009); Braun and Griebel (2009)) upon which the architecture is based. The authors note it is not immediately clear how well KANs may perform for more complex distributions common to, say, natural language or computer vision, especially at modern-day scale. They further observe that training KANs is about 10x slower than training equivalent MLPs – though they claim that since KANs need 100x fewer parameters, this is not a fatal flaw. These are our two greatest challenges in terms of KANs being an attractive drop-in replacement for MLPs, and we explore them here; this is the first in-depth study to our knowledge that applies the KAN architecture to the natural language domain.

On motivation, state-of-the-art large language models are only growing in size, with models like Llama-3 and GPT-3 reaching 70 and 175 billion parameters (Meta (2024); Brown et al. (2020)), respectively. In GPT-2 (Radford et al. (2019)), the MLPs present in each decoder block collectively make up 63.1% of the model's parameters: being able to compact these layers into orders of magnitude

fewer parameters would bring about a host of benefits, from a lower environmental impact in running the models to enabling them to fit in fewer GPUs, making them cheaper and more accessible to run. Given the increasing pervasiveness of large generative models in society across all fields, the further potential of making these layers more explainable is of large importance, especially for sensitive fields like medicine and justice.

In this work we focus on decoder-only based transformers and find KANs are unfortunately not a good candidate for distillation via MLP-approximation. We do get the promising result that KANs are powerful enough to learn the complex, high dimensional distributions inherent to large language modeling. When it comes to approximating some existing MLP by training on its input/output vectors as is typical in distillation, however, KANs are not as performant as simply retraining with MLPs when the parameter budget is controlled, at least for the shallower layers of the transformer.

3 Related Work

KANs as proposed in Liu et al. (2024) draw from the Kolmogorov-Arnold representation theorem, as opposed to the universal approximation theorem (Hornik et al. (1989)) that underlies MLPs. Specifically, Liu et al. reformulate the classic Kolmogorov-Arnold representation theorem as an $[n, 2n + 1, 1]$ dimensional KAN and propose an extension to arbitary depths and widths. A key difference between KANs and MLPs are that KANs have learnable activation functions on edges, as opposed to fixed activations on nodes as is common in MLPs; KAN nodes merely sum over all their inputs. In addition to the aforementioned limitations regarding training time, distributional complexity, and unknown applicability that partially motivate this study, many of the architectural and hyperparameter choices in the original KAN proposal are underexplored, like the choice of basis function for the activations (there, B-splines). Early works are already starting to explore the implications of these choices, such as Li (2024), both for accuracy and efficiency – and we find the selection is quite imperative to good performance, as detailed in §5.3.

The classic paradigm for knowledge distillation involves a large, performant "teacher" model trained on some task and a lightweight "student" model that we want to adapt to that task, often training the logits of the student model to be as close as possible to those of the teacher model over a wide variety of inputs (Hinton et al. (2015); Ba and Caruana (2014); Tang et al. (2019)). In the domain of natural language, a common variation of this strategy is rather than training on logits, the output text of the teacher model is used instead (e.g., Hsieh et al. (2023)). For this work, since we focus on approximating individual MLPs rather than the end-to-end language model, we adopt the style of the former setup.

4 Approach

For this work, we choose GPT-2 (XL) as our distillation target as it is open source, accessible enough at 1.5B parameters to fit in a single GPU, and relevant in its applicability to its successors, GPT-3, GPT-4 (OpenAI et al. (2024)), and beyond. GPT-2 has 48 decoder blocks, each with one shallow MLP of hidden dimension 6400 from model dimensionality 1600, meaning about 983M parameters of the 1.5B come from MLPs alone.

4.1 Dataset Generation

First we must synthesize datasets for performing our distillation on a given MLP. We run forward passes over unpadded text through the pretrained model and leverage PyTorch forward hooks to collect (input, output) tensor pairs – each of shape (sequence_length$_i$, d_model) – for each example i, for each MLP of interest. Since these MLPs are permutation invariant, transforming each token's representation irrespective of position or context, we drop the boundaries between input examples for our tensors, concatenating on the first dimension to get two large $N \times$ d_model tensors, one for MLP inputs and the other for MLP outputs. Here N is the total number of examples in our distillation dataset, and it equals the number of tokens we passed through the pretrained model. We do a 90/10 train/val split on these matrices to form our final datasets for distillation, and an 80/20 split for layer depth scaling (as introduced in section §4.3). Default fp32 precision is used and preserved throughout the study.

4.2 Baselines

For a first global baseline, we make a copy of the pretrained MLP (MLP_ϵ) and add to each parameter element a random Gaussian perturbation $\epsilon \sim N(0, \sigma_i^2)$, where σ_i is the standard deviation over all entries in the overall weight or bias matrix that contains the parameter. Then, for each KAN experiment, we create a second baseline (MLP_r, r for retrain), instantiating a new MLP of the same architecture as the source MLP with parameter count approximately equal to that of the KAN (achieved by shrinking the hidden dimensionality as needed). We train the MLP_r weights from scratch over our tensor data. The first baseline is meant to give us a sanity check for the sensitivity of the parameters and a figure which we must get our KANs to beat. The second baseline probes the ability of the MLP architecture to (re-)approximate the source function given the same parameter budget.

4.3 Setup

We select a single layer's MLP from GPT-2 to assess KAN-based distillation feasibility as we vary the parameter count across MLP_rs and KANs from about 2.5M to 20M. The first layer's MLP is chosen as intuitively it is the most important to get right; small errors earlier in the model may cascade in subsequent layers' computation to greatly degrade performance.

For our KAN implementation, we leverage the `FastKAN` library from Li (2024). We notice this implementation is missing bias adjustments compared to the original KAN library, so we adapt the source slightly to add them in after empirically finding they provide lift, but the library is otherwise used as is. For GPT-2 inferencing and LLM evaluation (see §5.1), we leverage the transformers libraries (Wolf et al. (2020)).

After the distillation study on the first MLP, we then fix the parameter count to be at full (undistilled) scale and compare MLP_r vs. KAN performance as we vary the layer depth of the source MLP. The goal here is to understand whether it is easier or harder for KANs/MLPs to approximate the different functions over the depth of the model.

5 Experiments

5.1 Evaluation method

For the quantitative evaluation of our MLP approximation experiments, we use the MSE scores between our predicted and ground-truth output vectors. Once we have some approximate MLP_r or KAN to substitute a given source MLP in the model, we also would like to assess the impact of the substitution to the downstream model performance; for this, we select three datasets used in the original GPT-2 paper, LAMBADA (Paperno et al. (2016)), WMT-14 FR-EN (Bojar et al. (2014)), and CNN/DailyMail (Nallapati et al. (2016)). We use the default quantitative metrics for these datasets as used in the GPT-2 paper – accuracy for final-word prediction for LAMBADA, BLEU for WMT-14 FR-EN, and ROUGE score for CNN/DailyMail. Prompting strategies and preprocessing (or the lack thereof) follows the methodology laid out in the GPT-2 paper, if available. Note we use subsamples of 500, 100, and 500 from the test splits for each of the three datasets, respectively, due to compute budget reasons, though we find this is nevertheless sufficient to tease out end-to-end model degradations.

5.2 Data

For our text data, we use subsamples from BookCorpus (Zhu et al. (2015)). We observe that the distribution of BookCorpus is heavily skewed towards shorter text sequences compared to the full context length that GPT-2 supports (see appendix A), which may be cause for generalization concerns once we plug these modules back into GPT-2 – even though the MLPs are permutation invariant, there should still be some signal in the data from the positional embeddings. Empirically we find no such universal catastrophic drop however between the base model and our modified models even on longer context datasets like CNN/DailyMail, where ROUGE score is on par with the base model after making some layer substitution (e.g., a KAN for the final layer's MLP, see section §6).

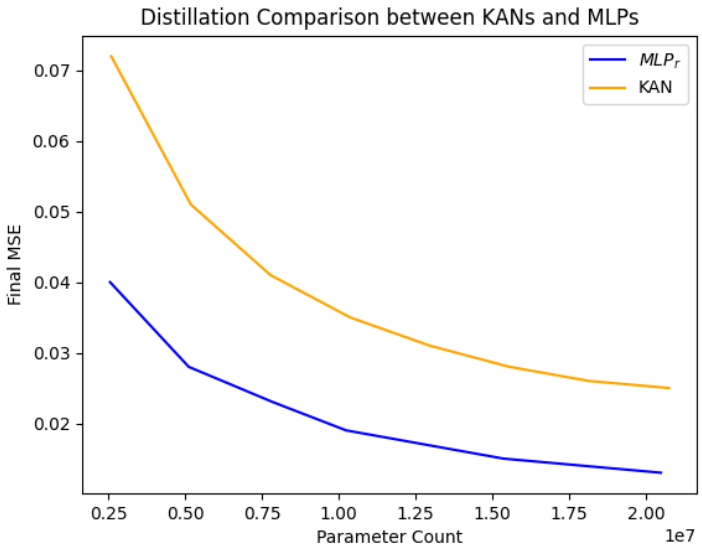

Figure 1: Comparison of MSE loss in approximating source MLP function between KANs of varying parameter counts and MLPs given equivalent parameter budgets, both trained up from scratch.

With respect to dataset size, we observed that while final MSE consistently plateaus after about 50 epochs (see appendix B) given a fixed number of examples, we do see further performance gains if we increase the subsample size (see appendix C for an example). Nevertheless, we find 50,000 examples is more than enough for assessing distillation capacity between MLPs and KANs for the first transformer layer's source MLP. For the layer depth scaling in §6.2, we use a smaller subsample at 25,000 examples to speed up iteration time.

5.3 Experimental details

We arrive at using the `FastKAN` library for our KAN implementation by way of iteration. We start with the original `PyKAN` code from Liu et al. (2024) and quickly find it too slow and inaccurate for our high-dimensionality and larger dataset-size use case. This motivates us to try the `efficient-kan` library from Blealton (2024) which achieves many orders of magnitude speed-up by avoiding the expansion of intermediary variables in calculating each activation function. After exhaustive hyperparameter exploration on GPT-2 (small), our KAN MSE losses are off by an order of magnitude even when given the same parameter budget – until we try using the `FastKAN` implementation with its choice of Gaussian radial basis functions which bring scores up to par at least for the full parameter budget case.

With respect to hyperparameters, for both baselines, we have no modeling hyperparameters to tweak as there is no training in the perturbation case and the architecture is fixed in the retrain case. For KAN experimentation, we face a wide variety of modeling hyperparameters and use GPT-2 (small) to inform our selection; ultimately we constrain ourselves to single-hidden-layer KANs of varying widths as we find these perform the best in this setting. Grid size, grid range, base activation, and other choices are kept at library defaults.

For training, we find MSE loss, no regularization, 512 batch size, 1×10^{-3} learning rate, 50 epochs, AdamW optimization, and an exponential learning rate schedule with gamma=0.95 works best for both KANs and MLPs.

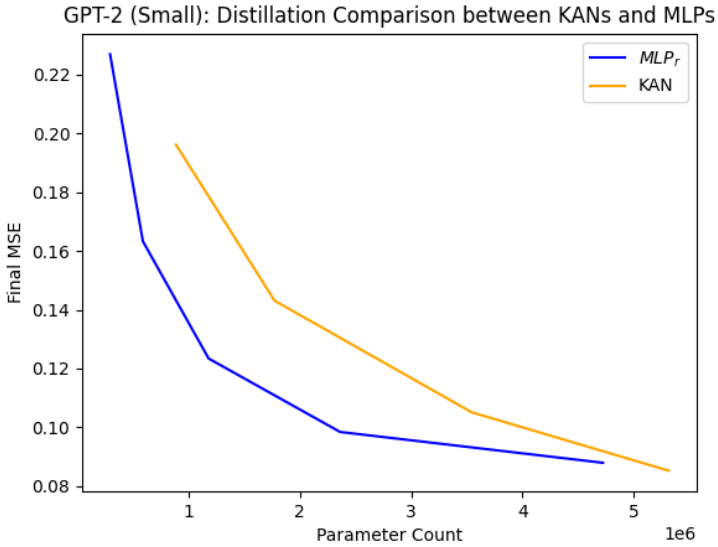

Figure 2: Repeat of distillation experiment, comparing MSE loss in approximating source MLP function between KANs and MLPs of varying parameter budgets, but for GPT-2 small which has a lower model dimensionality of 768.

6 Results

6.1 KAN-based Distillation

Figure 1 shows the results of our distillation experiment. We observe that KANs are able to approximate the source MLP to a great degree – all MSEs are on the order of 1×10^{-2} for both MLPs and KANs for all parameter counts, meaning the distribution is not so complex or so poorly suited for Kolmogorov-Arnold representation theorem that the KAN cannot pick it up in its entirety, which was a question we aimed to answer in this study. Our perturbation baseline, MLP_ϵ, averaged 0.381 MSE, and we score a completely random MLP as well (with zeroed out biases and zero-mean Gaussian initialization with $\sigma = 0.02$ for linear weights, per the GPT-2 defaults) to see an average of 0.524 MSE. Both of these figures are much larger than our trained results (hence why they are omitted from figure 1). The KANs are, however, consistently underperforming MLPs given the same parameter budget, meaning if our true goal was distillation at the end of the day, we would be better off using a smaller MLP to do the job rather than a KAN.

Given the disparity between this result and the parameter efficiency claims of Liu et al. (2024), we wonder if during the pretraining of GPT-2, the MLPs learned some set of functions that they were a more natural fit for compared to KANs; that is, it is unclear if had we pretrained with smaller KANs from the beginning, the KANs could have learned some different, better-suited set of function that jointly yield the same end-to-end language model performance as the larger MLP-based model. But that would no longer be distillation.

Figure 2 hints that instead the performance delta may be attributed to KANs struggling with the high dimensionality of the space, since for the lower dimensional GPT-2 small, the difference between MLPs and KANs is less pronounced, especially in the higher parameter budget regime where KANs approach performance parity. This is an interesting result if only for motivating an interpretability study for how the KAN is making its language predictions after indeed achieving MLP parity.

6.2 Layer Depth Scaling

The inability of KANs to beat MLPs for the first transformer layer over all parameter scales raises the question of how they might compare for the other layers, too. In figure 3, we fix the parameter budget to be full scale and train both a KAN and an MLP to approximate the source MLP at some decoder

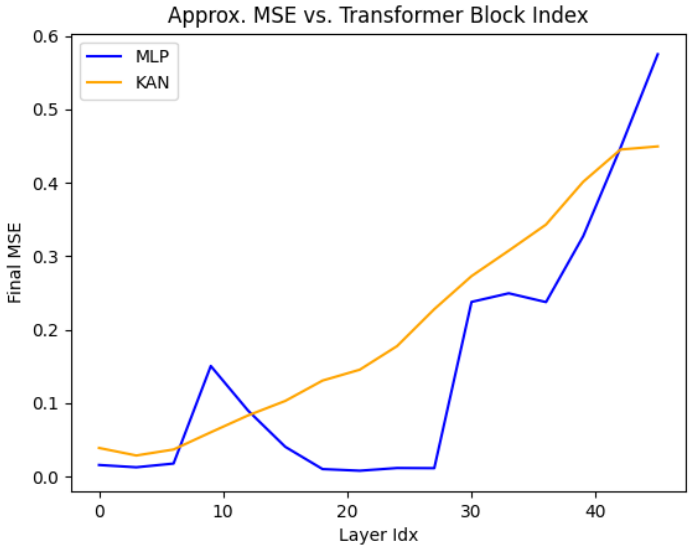

Figure 3: MSE for source MLP approx. vs. layer index of that MLP in the transformer. We see an increasing trend for both MLPs and KANs, though MLPs exhibit some interesting behavior in the middle layers. We measure every three layers for a total of 16 results for each of our two architectures.

block depth, where depth is varied. We observe increasing trends for both KANs and MLPs, though we see patches where KANs outperform MLPs and vice versa. Of particular interest is the middle section of the MLP trend; the KANs smoothly increase in final loss with depth, but for the middle layers, MLPs reach the same low loss regime that they achieved for the early layers, on the order of 1×10^{-2}. We include the loss curves for each of the runs used to build figure 3 in appendices E and F. We notice the KAN training dynamics are consistent across layers – a steep drop in the early epochs then decreasing returns until we eventually plateau, where the plateau point increases gradually with layer depth. MLP layers 30 through 42 see a large loss spike early in training, likely falling in some unlucky loss regime in the optimization landscape. That this happens for this contiguous chunk of layers only might suggest they're learning similar types of functions/features which are more difficult to approximate; the behavior altogether might be mitigated in future experiments through better hyperparameter tuning.

All runs here adopt the same hyperparameter suite we optimized for the first layer. The low MLP losses of layers 18 through 27 in figure 3 make us question if better hyperparameter tuning might be the answer for all layers to perform just as well – but we see no success with hyperparameter tuning over our highest loss layer, depth 45.

Intuitively, as transformer depth increases, we would expect the layers to be learning increasingly nuanced and sensitive features, with represented functions growing more complex. It would make sense, then, for the difficulty of the approximation task we tackle here to increase in turn, explaining the upward trend in MSE loss that both architectures exhibit with depth. As a loose sanity check, we plot the standard deviations of the pretrained GPT-2 MLP weights as a function of depth in appendix D, observing an increasing trend for the projection matrix of deeper MLPs, though we note this is nothing conclusive.

Finally, we wish to get a sense for what 1×10^{-2} (or worse) MSE means in terms of the end-to-end language model performance. In table 1 we show the results of substituting the MLP or KAN model we trained to approximate a source MLP at some depth into pretrained GPT-2 and evaluate on some of the datasets explored in the original GPT-2 paper. The base model results are below those originally listed for all three datasets – as a reminder, we only evaluate on subsets of the test datasets, and the authors do not list all the preprocessing steps they undertook for their evaluation – but we still get enough of a signal to understand the language model's sensitivity to our approximation error.

	WMT14 FR-EN	CNN/DailyMail	LAMBADA
	BLEU	ROUGE-AVG	ACC
BASE	0.0842	0.0694	0.322
MLP0	0.0000	0.0166	0.320
KAN0	0.0000	0.0413	0.334
MLP47	0.0776	0.0676	0.318
KAN47	0.0818	0.0698	0.324

Table 1: Language metrics post MLP substitution.

We observe the final word prediction of LAMBADA is robust to the substitution, with base model performance preserved for both KAN- and MLP-based subs. The stronger performance of this dataset is not surprising given it is close to the next token prediction that these models were trained on, LAMBADA was formed using BookCorpus as a source, and for highly confident predictions in the base model, small perturbations due to approximation loss are not enough to offset the confidence in the next token. CNN/DailyMail and WMT14 FR-EN are more complex, requiring longer sequences of concentrated generations. We see that when we substitute the first layer MLP – which perhaps tracks critical, lower-level features in the text, and where approximation loss is more costly, cascading throughout the rest of the computation – the ROUGE-AVG (average of ROUGE-1,2,L) and BLEU scores see significant degradation. Qualitatively, we see repeated sequences pop up more frequently (e.g., "the the the ...") and a decrease in relevance even for less repetitive generations. Clearly the accuracy of our source MLP approximations for early transformer layers is insufficient for maintaining good performance; on the flip side, despite the order of magnitude higher loss observed in the approximation of the deepest transformer layers (6×10^{-1} vs. 1×10^{-2} for early layers), performance is preserved.

7 Conclusion

Overall we find that KANs are a poor candidate for MLP distillation in transformer-based language models compared to simply smaller MLPs. They are, however, capable of learning the complex distributions of natural language at scale, especially for lower model dimensionality. KANs struggle more with approximating later layers in the transformer, though we find MLPs may share this weakness – the more nuanced, sensitive features deeper in the model are just harder to learn. This work is limited in terms of the scales of data and model sizes explored; future work could validate KANs on much larger models, like Llama-2, which has a higher model dimensionality, and more diverse datasets. Future work could also experiment with pretraining language models from scratch, ablating between KANs and MLPs in the transformer layers.

8 Ethics Statement

The uber goal of this project was reducing the number of parameters needed for language modeling through the introduction of KAN-based distillation. Had the results been favorable and we achieved such a reduction, this would have posed ethical concerns in making advanced models more freely available; they could be leveraged more easily by bad actors for malicious use cases such as generating spam or facilitating scams. A mitigation strategy to reduce the availability might be deploying any distilled models behind closed endpoints in walled garden ecosystems, where we could control and filter model outputs with techniques like prompt rewriting and post-generation safety classifiers. A second concern would be if any distillation, even if done through MLPs as concluded in this work, undoes any safety alignment previously executed on the model. This safety reversal could allow toxic, biased, sexist, or otherwise unsafe responses to be generated by the model to the detriment of the downstream user or application. To fix this, we can simply reapply model alignment techniques, such as the post-training model interventions of PPO-based RLHF and/or DPO on any models after distillation.

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Blealton. 2024. efficient-kan. `https://github.com/Blealtan/efficient-kan`.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jürgen Braun and Michael Griebel. 2009. On a constructive proof of kolmogorov's superposition theorem. *Constructive Approximation*, 30:653–675.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Alexander B. Givental, Boris A. Khesin, Jerrold E. Marsden, Alexander N. Varchenko, Victor A. Vassiliev, Oleg Ya. Viro, and Vladimir M. Zakalyukin, editors. 2009. *On the representation of functions of several variables as a superposition of functions of a smaller number of variables*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes.

Ziyao Li. 2024. Kolmogorov-arnold networks are radial basis function networks.

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

Meta. 2024. Introducing meta llama 3. `https://ai.meta.com/blog/meta-llama-3/`. Accessed: 2024-06-06.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon

Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

A BookCorpus Distribution

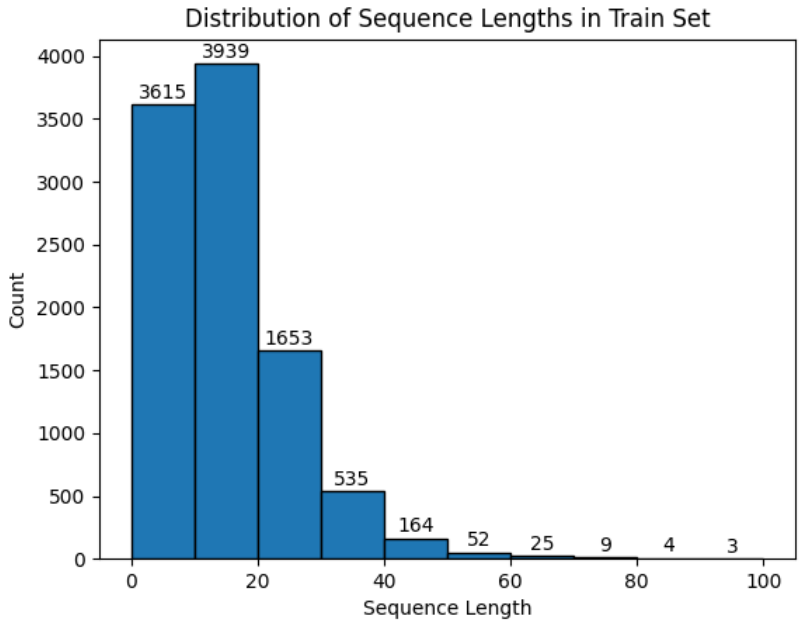

Figure 4: Distribution of BookCorpus example sequence lengths over random 10k subsample.

B Epoch horizon for sample MLP, KAN runs

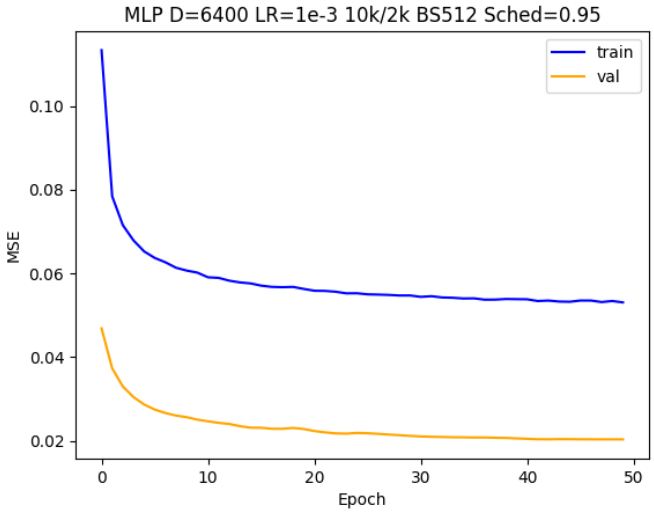

Figure 5: Sample MLP training run metrics over 50 epoch horizon. MLP has dropout p=0.1, which is responsible for the delta between train and val metrics. We can see that by the 50 epoch mark we have plateaued.

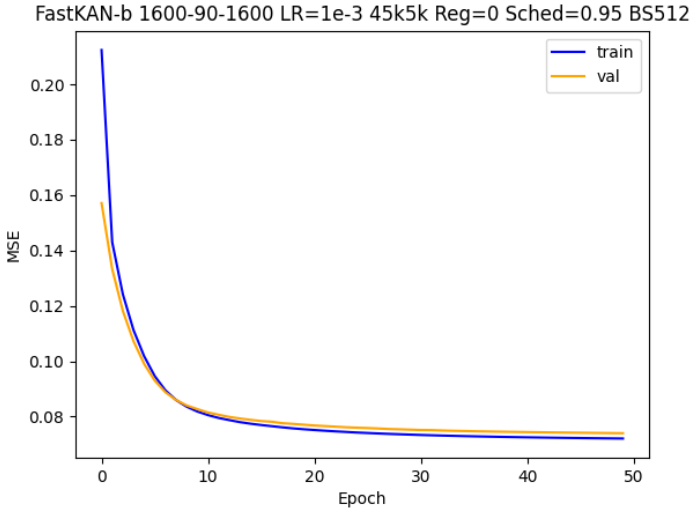

Figure 6: Sample KAN training run metrics over 50 epoch horizon. We do each validation pass and corresponding metric calculation after training on all minibatches of data for the given epoch, whereas the training metric is accumulated over each minibatch in the epoch and averaged at the end. This is why the validation set appears to do better in the early epochs, though the benefit disappears as we train for longer and start to slightly overfit. Here too we see a plateau by the 50 epoch mark.

C Impacts of Subsample Size

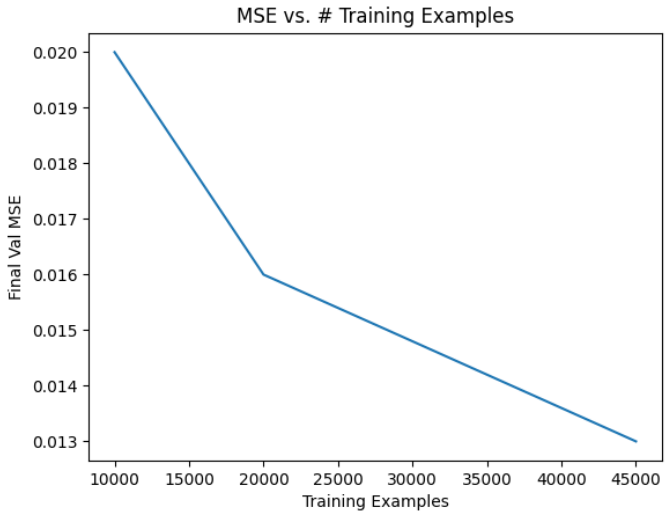

Figure 7: As we increase the example count, our final loss continues to decrease – given BookCorpus has more than 70,000,000 examples, it is likely we could have pushed the MSE down further had we taken a larger subsample, but we found this order of magnitude to be sufficient for our exploration.

D MLP Weight Layer-wise Stats

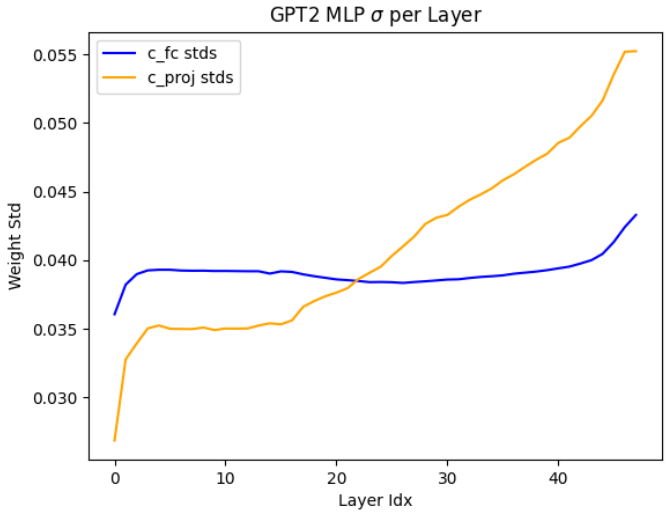

Figure 8: We see the standard deviation of the c_fc (1600, 6400)-shape expansion matrix is mostly consistent across layers, save for a slight uptick towards the end. The standard deviation of the c_proj (6400, 1600)-shape projection matrix however does increase with depth, perhaps indicative of the increasingly nuanced and higher level features we may be learning deeper in the model.

E MLP approx. by MLP – Depth Scaling Curves

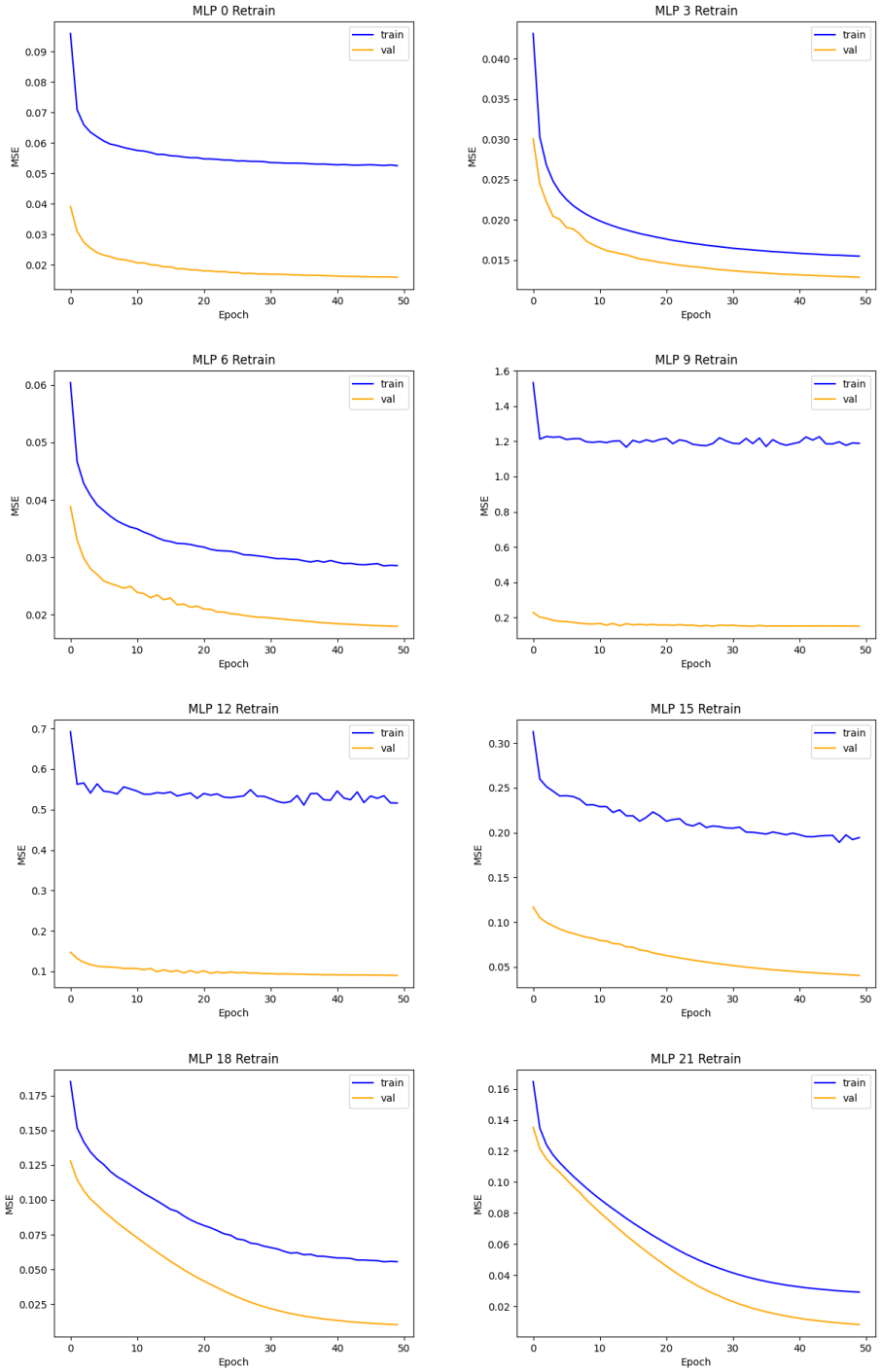

Figure 9: Training curves for MLP-based MLP approximation over varied depths.

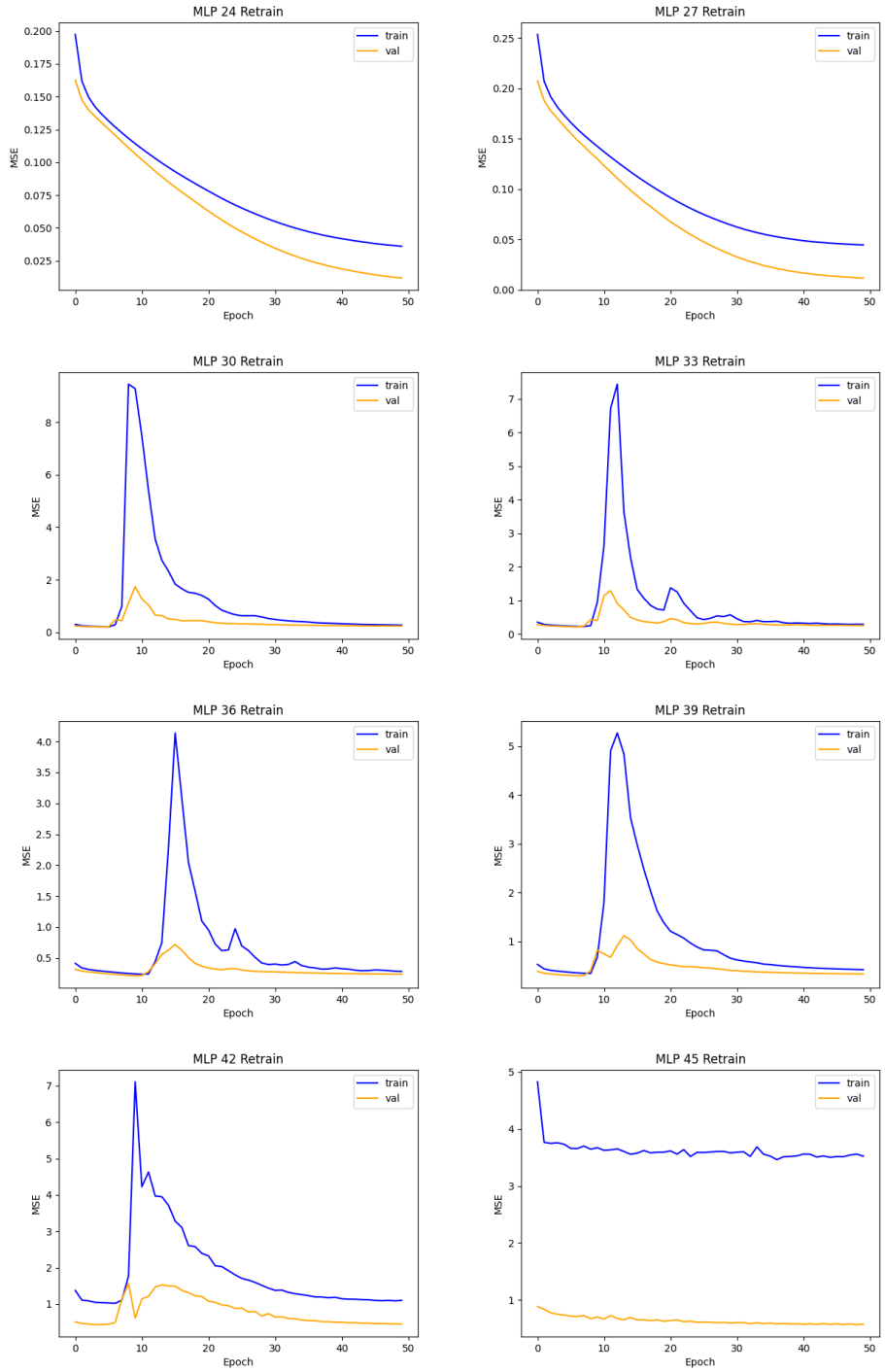

Figure 10: Training curves for MLP-based MLP approximation over varied depths.

F MLP approx. by KAN – Depth Scaling Curves

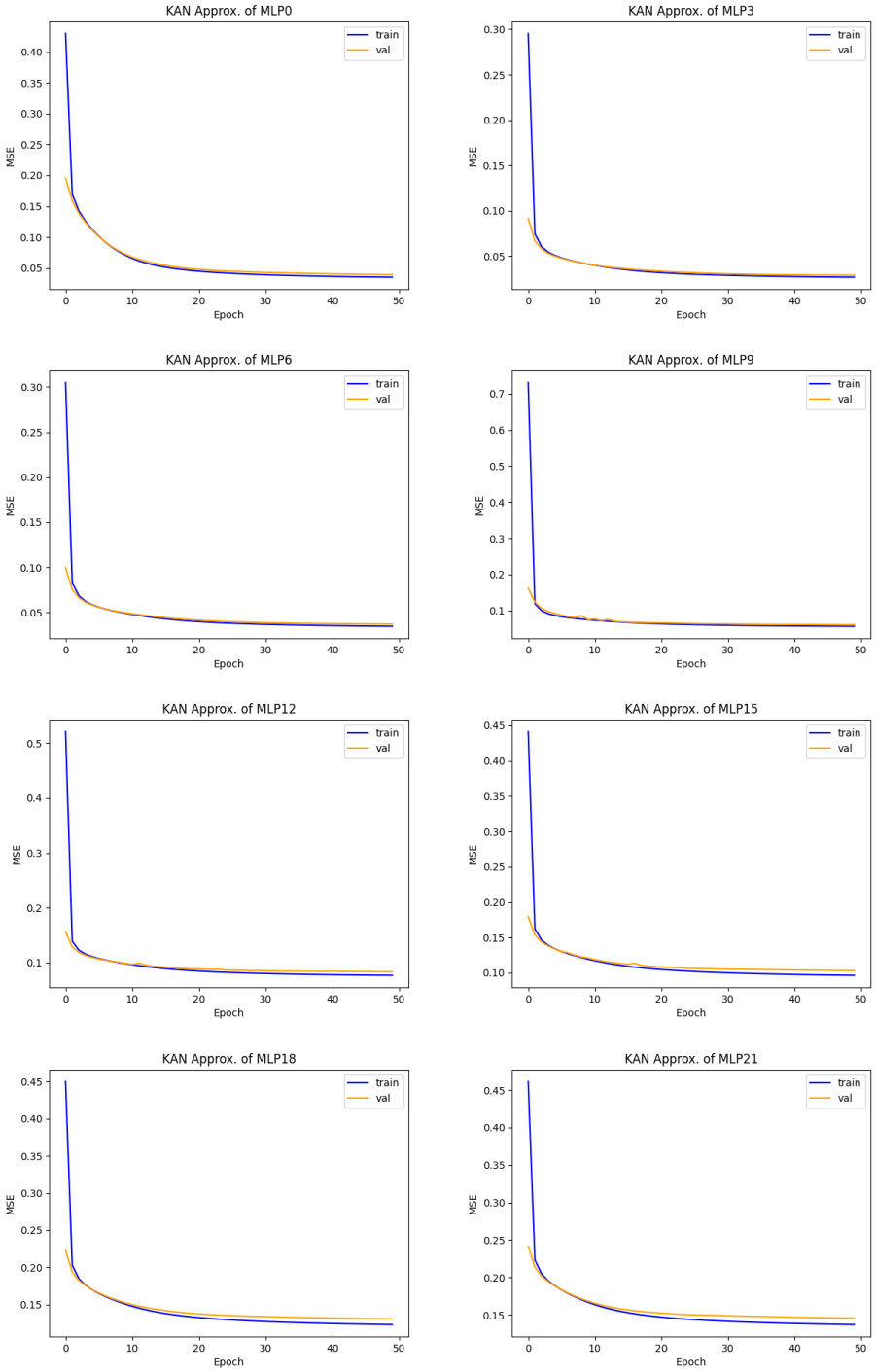

Figure 11: Training curves for KAN-based MLP approximation over varied depths.

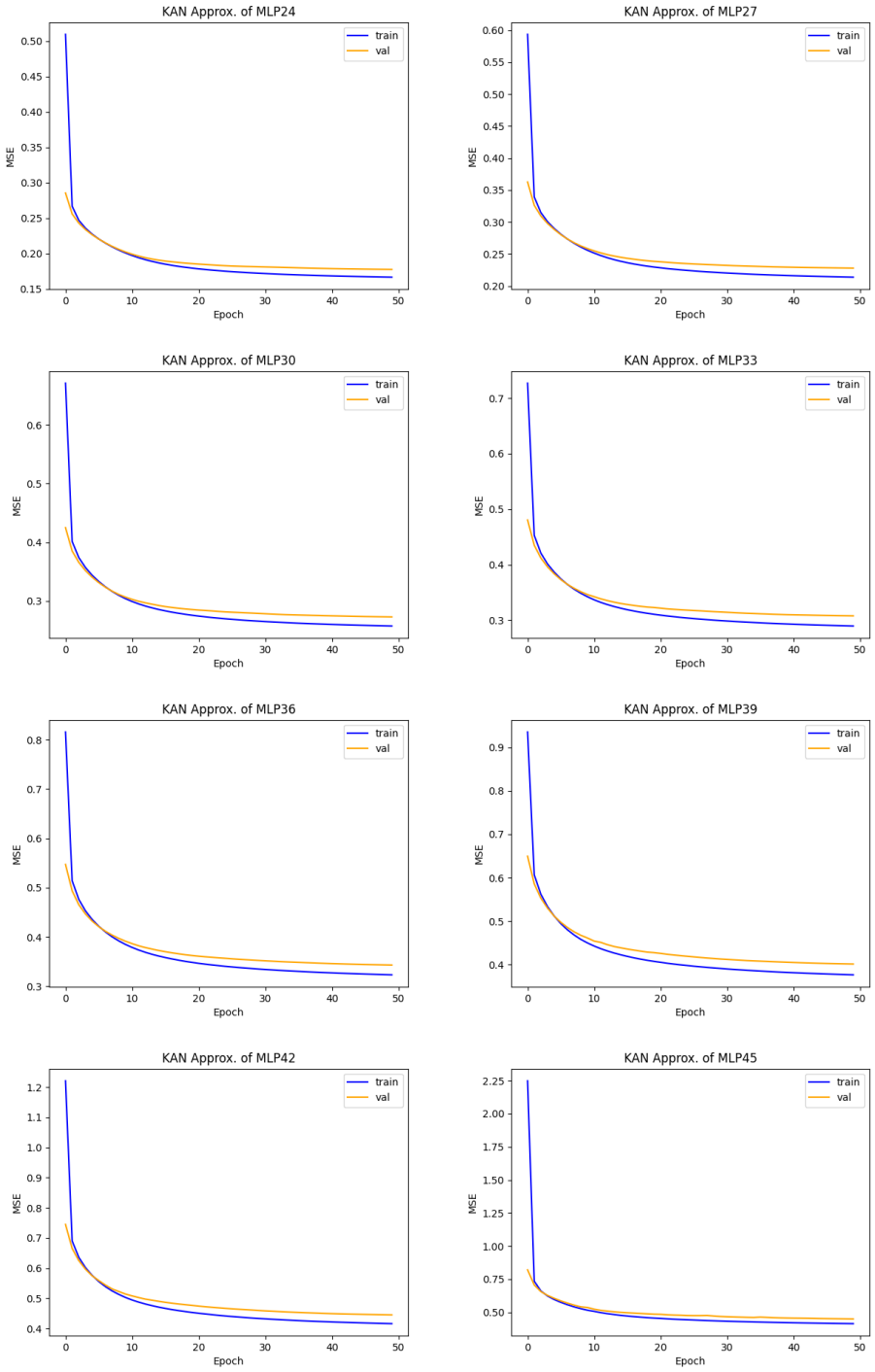

Figure 12: Training curves for KAN-based MLP approximation over varied depths.