Interpretable Machine Learning Regression for Nuclear Applications with Kolmogorov-Arnold Networks (KAN)

Omer Erdem [1,*], Nataly Panczyk[1], Majdi I. Radaideh[1]

[1]Department of Nuclear Engineering and Radiological Sciences, University of Michigan, Ann Arbor, MI 48109

[leave space for DOI, which will be inserted by ANS]

ABSTRACT

Artificial intelligence (AI) looms as a double-edged sword in the nuclear industry. On one side lies the promise of fast, accurate results. On the other, a black-box of calculations that is very hard to verify, validate, or regulate. This work presents an alternative to traditional, black-box AI: a Kolmogorov-Arnold Network (KAN) that produces interpretable results in the form of a symbolic equation. In this work we apply a KAN to two nuclear engineering problems using `PyKAN`, a Python package for constructing KANs with `PyTorch`. The first application predicts critical heat flux with 60 years of experimental data. The second predicts neutron flux in a high-temperature gas-cooled reactor with simulated data. We then compare the KAN model results against a traditional feed-forward neural network and conduct an explainability analysis using the Shapley game theory metrics to effectively evaluate each model's accuracy, interpretability, and explainability. While the KANs created for both problems in this analysis slightly underperformed in terms of accuracy, these results are offset by significant increases in interpretability and are on par in terms of explainability. While future work must consider further hyper parameter tuning and additional case studies, KANs stand to address the double-edged sword predicament of AI in nuclear.

Keywords: Explainable AI, Kolmogorov-Arnold Network (KAN), Feedforward Neural Network, Critical Heat Flux, HTGR

1. INTRODUCTION

While artificial intelligence (AI) poses great opportunities for the nuclear industry, like enhancing safety, performance, and efficiency, it also often comes with non-negotiable risks for regulators, utilities, and customers alike. Most AI models worth mention are "black-boxes". We provide mass amounts of data, and the algorithms "learn" a pattern that they can use to predict useful outputs with impressive accuracy. The problem in this approach is that, traditionally, the "learning" comes in the form of assigning layers upon layers of weights and biases that are, to date, impossible to detangle and extract information from. This means that we cannot reverse-engineer the conclusions an AI model draws, and therefore, we cannot provide context, understanding, or meaning beyond the probability of correct answers we found in a test set. If we could provide such metrics, the applicability of AI to sensitive industries like nuclear could skyrocket.

This work proposes a reframed approach to AI development that directly addresses the black-box problem. By implementing a recently developed machine learning model, the Kolmogorov-Arnold Network (KAN), we can create models that produce equations representing a system rather than decontextualized numerical

*oferdem@umich.edu

outputs for that system [1]. The KAN architecture is underpinned by the Kologorov-Arnold representation theorem, which states that any multivariate continuous function can be represented as a superposition of univariate continuous functions [2]. The results from a KAN model are exactly that, a superposition of univariate continuous functions that are inherently traceable because they have closed form expressions that engineers can analyze, validate, verify, and assess. In short, much more than a single number. The quantification of the uncertainties of these expressions is easier compared to the black-box AI models. Of course, such perks are not without drawbacks. This paper seeks to introduce KANs and assess their performance in nuclear engineering applications featuring regression problems to predict critical heat flux and predicting reactor power in a microreactor system. We seek to determine whether the promise of KANs really is too good to be true, or if what these networks bring in interpretability makes up for what they may lack in accuracy. For the purposes of this analysis we will consider interpretability and explainability as separate qualities, where the former evaluates *how* a model produces an output (i.e., we can reproduce the model's results) and the latter evaluates *why* (i.e., which inputs were most important in determining the output). Further, it will consider if even KAN interpretability is "good enough" for regulatory purposes, and if further metrics can supplement to ensure trustworthy results.

2. THEORY

The KAN architecture truly rethinks modern machine learning. Most machine learning algorithms today are built upon the Universal Approximation Theorem (UAT), which was proven by George Cybenko in 1989 [3]. It shows that, given that there are enough data to reasonably train the network, any continuous function can be approximated arbitrarily well by a neural network with at least one hidden layer and a finite number of weights [3]. KANs, however, rely on an entirely different mathematical foundation – the Kolomogorov-Arnold Representation Theorem (KART). This theorem states that a multivariate continuous function can be written as a finite sum of continuous univariate functions [4]. Mathematically:

$$f(\vec{x}) = f(\vec{x_1}, ..., \vec{x_n}) = \Sigma_{q=0}^{2n+1} \Phi_q \left(\Sigma_{p=1}^{n} \phi_{q,p}(\vec{x_p}) \right) \tag{1}$$

where $\vec{x_p} \in [0,1]$, $\phi_{q,p}(\vec{x_p}) : [0,1] \to \mathbb{R}$, and $\Phi_q : \mathbb{R} \to \mathbb{R}$. This theorem on its own does not promise the success of KAN networks. In fact, a limitation of Equation (1) is that the one-dimensional functions are not required to be smooth, yet smoothness is essential for a neural network to learn using gradient descent algorithms. The scientific community used this smoothness obstacle to condemn the KART from becoming a feasible neural network for decades [5]. Over 30 years later, the authors of [6] gave the KART a second chance, relying on the knowledge that most functions (in science and elsewhere) *are* smooth. Their goal for KANs is therefore to accurately capture the majority of cases, with indifference to outliers.

We can intuitively convert the KART, shown by Equation (1) in its mathematical form, to a visual representation of a neural network with $2n + 1$ edges and n layers. We show this in Figure 1a. The authors of [6] then created a network that expanded beyond the limits of Equation (1) by going "wider and deeper," i.e., going beyond $2n + 1$ in number of edges and beyond n in the number of layers. This sought to mimic the "stacked" structure of the popular feedforward (fully-connected) neural networks (FNNs) and allow for more flexible, accurate learning. From a network-architecture perspective, this means that instead of placing activation functions on the nodes of a neural network, which is how FNNs are built, the KAN places activation functions on the edges [6]. We show a visualization of this extension in Figure 1b.

The expansion of the KART into a neural network with L layers can therefore be written as:

$$\text{KAN}(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_1 \circ \Phi_0)(x) \tag{2}$$

where Φ_L is a matrix of trainable activation functions $\phi_{j,i}$ where $j = 1, \cdots, n_{L+1}$ and $i = 1, \cdots, n_L$, $\circ$ is the composition of functions, meaning applying one function to the result of another function. For comparision,

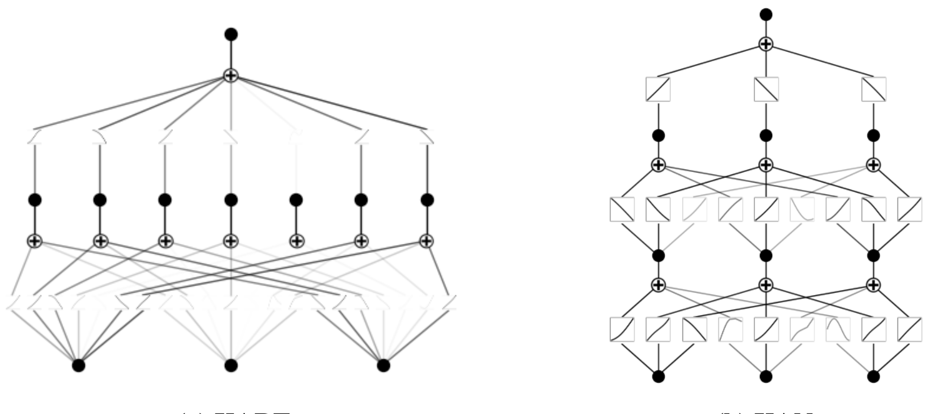

(a) KART (b) KAN

Figure 1. Original KART depicted as a neural network (left) vs. the expanded KAN network structure (right). Opacity of the nodes and edges indicate their weights.

a traditional FNN can be written as:

$$\mathrm{FNN}(x) = (W_{L-1} \circ \sigma \circ W_{L-2} \circ \sigma \circ \cdots \circ W_1 \circ \sigma \circ W_0)(x) \tag{3}$$

where W_L represents the transformation matrix for the L_{th} layer (linear weights and biases) and σ is the (non-linear) activation function. A clear distinction between Equation (2) and (3) is the the FNN separates its linear transformations (weights and biases) from its nonlinear transformations (activation functions), whereas the KAN combines all transformations in a single function, Φ. We will further consider how Equation (2) gets implemented in Section 4.

3. DATASETS

The following section will describe the datasets used to assess and compare KANs against a traditional feedforward neural network (FNN) in this paper. Due to the space limit in this paper, two benchmark datasets were selected from our AutoML framework, pyMAISE (Michigan Artificial Intelligence Standard Environment) [7]. The authors are exploring extending this work to more datasets available in the pyMAISE stack in future work.

The first dataset we have chosen to use as a demonstration in this paper is a simulated dataset representing the HolosQuad high-temperature gas-cooled reactor (HTGR), presented originally in [8] and now housed in pyMAISE [7]. The simulated data was generated using a Serpent model of the 22 MWth reactor. The dataset contains 751 samples and eight input features, which correspond to eight unique control drum angles. Applying rotational symmetry for the 4 quadrants in the reactor can multiply the sample size by four, resulting in a total of 3004 samples. The asymmetrical control drums in the HolosQuad HTGR manipulate reactivity in the reactor by adjusting how close their boron-carbide edge (roughly semi-circular) is to the fueled region of the core. Adjusting the angle of rotation of these drums between 180°and -180°significantly perturbs the flux in the core. The output features in this dataset therefore correspond to the neutron flux in each of the four quadrants of the HTGR as determined by these alterations [8]. For our analysis, we will only consider the output in the first quadrant (Q1). Though, both FNN and KAN support multi-outputs, we limited our results for brevity and space requirements.

The second dataset we chose as a demonstration is a critical heat flux (CHF) dataset from the Nuclear Regulatory Commission (NRC). It contains over 60 years worth of CHF values in over 21,000 samples. The critical heat flux of the systems are reported with respect to varied parameters. These parameters (input features) include: heated channel length (L), pipe diameter (D), mass flux (G), inlet temperature (T_{in}), pressure (P), and outlet equilibrium quality (X_e) [9, 10]. Unlike the HTGR dataset, the CHF dataset only contains one output feature: CHF.

4. METHODOLOGY

4.1. Previous Modeling Work

We generated the FNN model results used in this analysis using pyMAISE, which is an automated machine learning tool for nuclear applications developed and maintained by our team at University of Michigan [7]. pyMAISE uses each of the datasets shown in this analysis as a benchmark model. Thus, we have used the hyperparameter tuning results found in each pyMAISE benchmark to reproduce an optimized FNN model for each dataset. To conduct its hyperparameter tuning, pyMAISE uses grid, random, and Bayesian search algorithms [7]. The results of each FNN as modeled by pyMAISE are shown in Section 5 via the diagonal validation plots and in Table I. The focus of this paper is on KAN, therefore, FNN results are reported for benchmarking purposes.

4.2. KAN Implementation

In this section we will outline the general steps we used to implement a KAN for our analyses, though many of these are automatic features of the package `PyKAN`, which we used to create our models [6]. Full details on how the KART becomes an KAN can be found in [6] and is beyond the scope of this work. First, like the residual connections in an FNN, the KAN requires residual activation functions (ϕ in Equation (2)). These are written as

$$\phi(x) = w_b b(x) + w_s \text{spline}(x) \tag{4}$$

where

$$b(x) = \text{silu}(x) = \frac{x}{1 + e^{-x}} \tag{5}$$

and

$$\text{spline}(x) = \Sigma_i c_i B_i(x) \tag{6}$$

where c_i are trainable parameters and $B_i(x)$ are B-splines. w_b is initialized using Xavier initialization and w_s is initialized to one [6]. The `PyKAN` package approximates each linear combination of B-splines using a grid parametrization. This parametrization is flexible, such that activation functions can cross boundaries during training, and `PyKAN` updates its grid parametrization accordingly.

Much like an FNN, KAN models require the tuning of several hyper parameters to configure the B-spline components. The most critical B-spline parameters are the number of grid intervals at each node and the order of the spline. Typically, a constant spline order of three is used across most KAN applications. To increase the complexity of the B-splines at the nodes and thereby refine model results, we increase the grid intervals from a baseline of three after each training iteration. By continuing this iterative process of training and refinement, we can construct a model with increasingly intricate activation functions.

The process of optimizing the KAN model begins with a larger initial configuration. During training, we reduce the weights associated with redundant nodes, which we then ”prune” off after the first training phase. `PyKAN` defaults pruning thresholds to 0.03 for edges and 0.01 for nodes. By removing nodes and edges with weights below these thresholds, we reduce the model’s complexity and subsequent training time. After the pruning phase, we continue training the model for a specific number of epochs to minimize the training loss. This refinement ensures that the smaller model achieves optimal performance. An example illustrating the model before and after pruning is shown in Figure 2.

In addition to B-spline parameters, KAN models require other hyperparameters, such as the number of training epochs per step, the number of layers, and the number of neurons in each layer. For each model, we

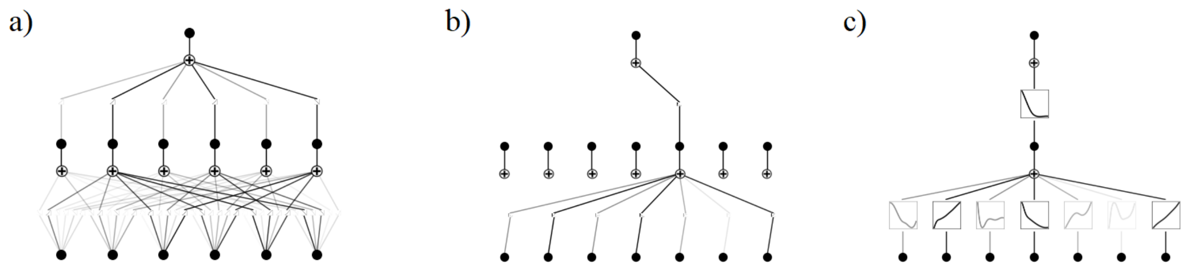


Figure 2. (a) A diagram of the initialized KAN model for CHF prediction, (b) KAN model after the first training, (c) KAN model after pruning of nodes and edges. Opacity of the nodes and edges indicate their weights.

set the number of neurons in each hidden layer equal to the number of input dimensions in the problem. We varied the number of hidden layers between one, two, and three to identify the optimal model configuration for each problem. We set the training epochs for each step to 100. Specifically, we trained the initial (unpruned) model for 100 epochs. Then, we trained the pruned model for another 100 epochs. Finally, we trained the refined models iteratively using configurations with 5, 10, and 25 grid intervals. This process totaled 500 training epochs for the KAN models. For fair comparison, we trained the FNNs for the same total number of epochs.

For as-advertised "interpretability," we convert the B-spline activation functions in the KAN models into symbolic functions using the SYMBOLIC_LIB package. The automatic fitting option in `PyKAN` evaluates every function within the symbolic equations package over the specified interval to identify the best match. To reduce conversion time, users can provide a smaller subset of functions in lieu of the full SYMBOLIC_LIB package.

4.3. Explainability Analysis

To provide both explainability *and* interpretability to KANs, we have created an ad hoc feature importance algorithm based on the popular SHAP (SHapley Additive exPlanations) framework that seeks to determine not just *how* the KAN came to its conclusion (we can see this in the symbolic equation it produces), but *why*. In other words, what input feature held the most value in determining the outcome for a given scenario. To do this, we use the concepts from game theory where we connect optimal credit allocation with local explanations using the classic Shapley values and their related extensions. We expect that the most important feature with the highest SHAP value would yield the greatest change in output when modified. *This framework is consistently applied to both FNN and KAN in this work.*

5. RESULTS

5.1. Model Performance

This analysis considers two separate case studies. For the high-temperature gas-cooled reactor (HTGR) control problem, we train Kolmogorov-Arnold Network (KAN) models with varying numbers of hidden layers to evaluate their effectiveness. Since the input-output relationship of the problem is unknown and may involve complex dynamics, we tested KAN configurations with layer structures [8,4], [8,8,4], and [8,8,8,4] with pruning of the low-importance nodes and edges after first training. We achieved the best test metrics with the [8,4] configuration and while using five final grid intervals.

In the case of critical heat flux (CHF) prediction, we tested KAN configurations with layer structures [6,1], [6,6,1], and [6,6,6,1] with pruning of the low-importance nodes and edges after first training. The best test metrics were achieved with the [6,1,1] configuration and while using 25 final grid intervals.

Table I. Model metrics comparison for FNN and KAN on two test cases.

	Case #1: HTGR		**Case #2: CHF**	
Model	FNN	KAN	FNN	KAN
Single CPU Training Time [min.]	4.64	2.97	192.08	23.35
R^2	0.9808	0.9634	0.9972	0.9822
MAE	$5.57 \text{x} 10^{16}$	$8.04 \text{x} 10^{16}$	51.99	120.05
MAPE [%]	0.18	0.26	3.28	8.32
RMSE	$5.33 \text{x} 10^{16}$	$6.74 \text{x} 10^{16}$	69.49	175.90
RMSPE [%]	0.25	0.33	4.35	11.79
Symbolic Conversion Time [min.]	-	0.93	-	9.94
Symbolic Equation R^2	-	0.9643	-	0.9757

The KAN test metrics in Table I are given for the single output of the CHF prediction problem and the first output (first quadrant flux) of the HTGR control problem. We extract the KAN test metrics before converting the models to the symbolic forms, but we report the R^2 values of the created symbolic equations after conversion to show the accuracy change of converting the models to symbolic equations. The symbolic equation form of the created KAN model for the HTGR control problem and CHF prediction problem are given in Equations (7) and (8). The resulting equations are much longer than the ones in the KAN documentation, since these problems have higher numbers of input dimensions and higher complexities.

The input parameters in Equation (7) correspond to control drum 1-8 angles (radians) in order from x_1 to x_8. The output parameter is the flux of quadrant 1 (neutrons/cm^2s). The input parameters in Equation (8) correspond to diameter (m), length (m), pressure (kPa), mass flux (kg m^{-2}s^{-1}, input temperature (°C), output quality (-), critical heat flux (W/m^2). The output parameter is the critical heat flux (W/m^2). Note that the symbolic equations correspond to the KAN models, which are trained with minimum-maximum scaled data. The equation results must be inverse scaled from 0 - 1 range to 2.45 $neutrons/cm^2s$ - 2.73 $neutrons/cm^2s$ range for HTGR quadrant flux and 130 kW/m^2 - 13345 kW/m^2 range in order to be in a physical range.

$$
\begin{aligned}
Y_{HTGR} = & -0.0006\cos(6.7502x_8 + 2.2243) + 0.2494 \\
& - 0.0013e^{-12.96(0.5389-x_3)^2} - 0.0041e^{-12.96(0.5211-x_5)^2} \\
& + 0.0073e^{-13.4771(0.5074-x_2)^2} - 0.0023e^{-13.0714(0.5048-x_7)^2} \\
& - 0.0037e^{-13.8063(0.4958-x_6)^2} - 0.0023e^{-12.2786(0.4898-x_4)^2} \\
& + 0.0076e^{-12.5821(0.4692-x_1)^2}
\end{aligned}
\tag{7}
$$

$$
\begin{aligned}
Y_{CHF} = 0.6975 - 0.6996\tanh\Bigg(& 0.6936\cos\left(1.5813X_1 - 4.589\right) \\
& - 0.0589\tan\left(2.7642X_3 - 4.5831\right) - 0.3475\tan\left(2.212X_4 - 1.4029\right) \\
& + 0.6529\tan\left(0.6X_5 + 9.9968\right) + 0.9714\text{atanh}\left(1.0X_2 - 0.992\right) \\
& + 1.3965 + 0.4367\exp\left(-6.1413\left(0.2107 - X_6\right)^2\right)\Bigg)
\end{aligned}
\tag{8}
$$

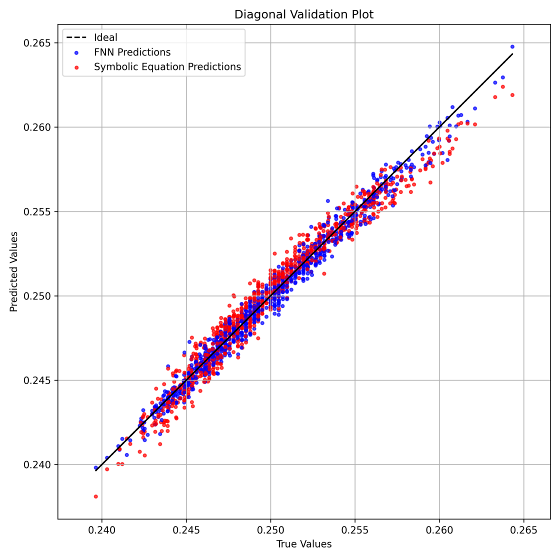
When comparing the training times of KAN models, a clear correlation between dataset size and training time emerges. The most time-intensive step in KAN training is the initial model training before pruning, driven primarily by the size and complexity of the initial model. For instance, considering dataset sizes of 3004 (HTGR) and 21453 (CHF) samples, the respective training times of 2.97 and 23.35 minutes, respectively, are consistent with expectations.

FNN training times significantly exceed those of KAN models. In the CHF prediction problem, the selected KAN configuration required training time that was 8 times shorter than that of an FNN. For problems with small datasets, KAN training times were 5 times shorter than FNN training times, highlighting the efficiency of KANs for the same number of epochs and batch sizes. The main reason for this difference is that KANs have smaller model sizes compared to FNNs after pruning. Even after including the symbolic equation conversion times, the training time of the KAN models still outperform FNN models.

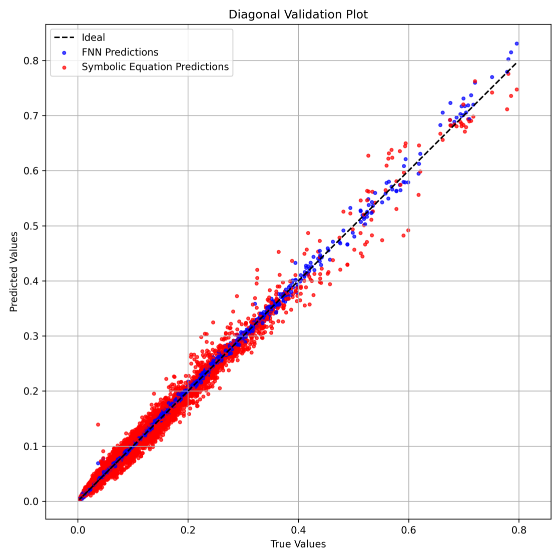
The accuracy of KANs is comparable to that of FNNs when the KAN model size is appropriately selected. However, KAN models with poorly chosen sizes exhibit significantly lower performance than FNNs. Using the KAN models require an understanding of the problem at hand. KANs perform well when the input-output relationship can be effectively represented by B-splines of the selected grid intervals and functions available in the symbolic library.

In the selected nuclear engineering problems (HTGR control and CHF prediction), FNNs consistently outperformed KANs in terms of accuracy. In both cases, the best KAN models exhibited approximately twice the MAE and RMSE of the best FNN models. Comparing our results to promising metrics in the literature [6] indicates that the suitability of KAN models and symbolic equations depends on the nature of the problem. When the KAN structure aligns well with the problem, KANs can achieve accuracy on par with FNNs. To demonstrate the accuracies of the created FNN models and KAN models, the diagonal validation plot of the test predictions are given in Figure 3.

Despite the reduced accuracy in some cases, KANs offer advantages in terms of shorter training times and simpler, interpretable input-output relationships, making them valuable for problems where understanding model results is a priority.



(a) Diagonal validation plot for HTGR dataset.



(b) Diagonal validation plot for CHF dataset.

Figure 3. Comparison of diagonal validation plots for HTGR and CHF datasets.

An important difference between the two approaches is training time. KANs can achieve shorter training times than FNNs when subjected to the same number of epochs. This is primarily attributed to the smaller model sizes inherent in KANs, which require fewer parameters to be learned. The reduced complexity of KANs not only accelerates the training process but also improves computational efficiency, making them attractive for scenarios where time and computational resources are constrained.

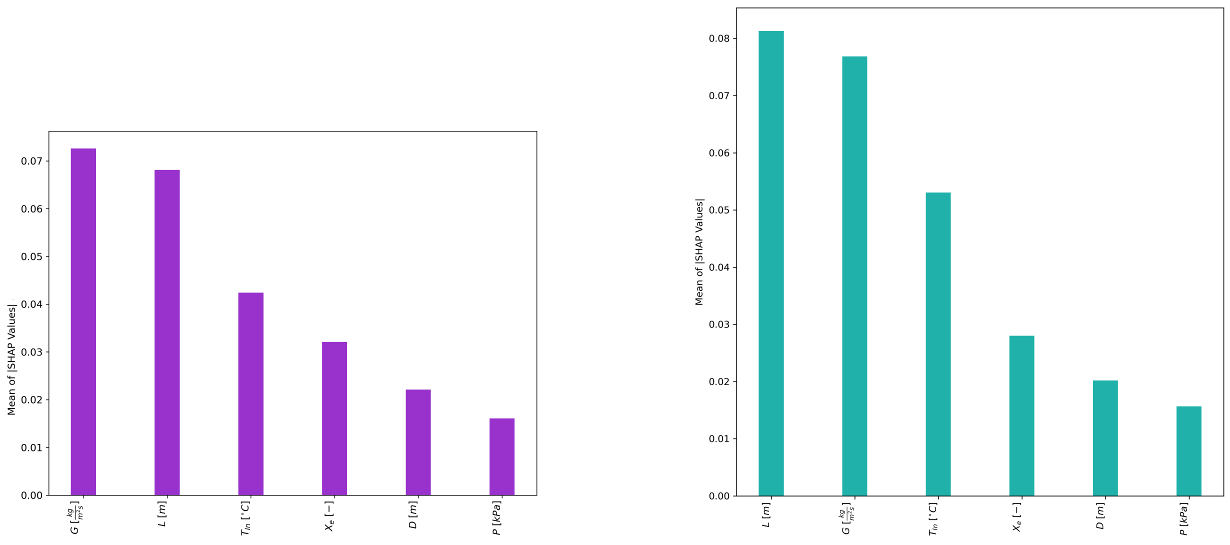
Still, FNNs significantly outperform KANs on accuracy. The increased number of degrees of freedom in FNNs allows them to better capture complex patterns and fit training data with higher precision. This increased flexibility is particularly beneficial in applications where a higher level of detail is required to model intricate nuclear phenomena. KANs face limitations in their ability to fit such detailed data due to their reliance on symbolic equations. The process of converting the model to symbolic form requires that the functions used in the model exist within the predefined symbolic library, which restricts the level of detail that can be captured. Further, conversion accuracy for KANs decreases with model complexity, which limits their applicability. On the other hand, we should highlight that our conclusions here hold for the two benchmark problems reported and further investigation of more nuclear problems analyzed with KAN is needed, which is the topic of our future work.

5.2. Explainability

Though model interpretability is inherent to KAN outputs in the form of a symbolic equation, we still observe a gap in model explainability for both KANs and FNN. Figures 4 and 5 show feature importance rankings for both the CHF application and HTGR application to address this gap. For the CHF explainability results, shown in Figures 4a and 4b, we find similar rankings for the feature importances for both FNN and KAN, respectively. The only difference between these rankings is mass flux (G) outranking channel length (L) as the most important feature in the FNN (L is most important in the KAN). This difference could be attributed to a difference in SHAP methods, as we used DeepLIFT (better and designed for neural networks) on the FNN and KernelSHAP on the KAN. Still, considering the closeness in magnitude of the G and L rankings for both models, it is a very marginal discrepancy. The similarity between these results suggests general agreement between the models on overall feature importance for predicting CHF in the fact that channel length, mass flux, and inlet temperature are the leading features.

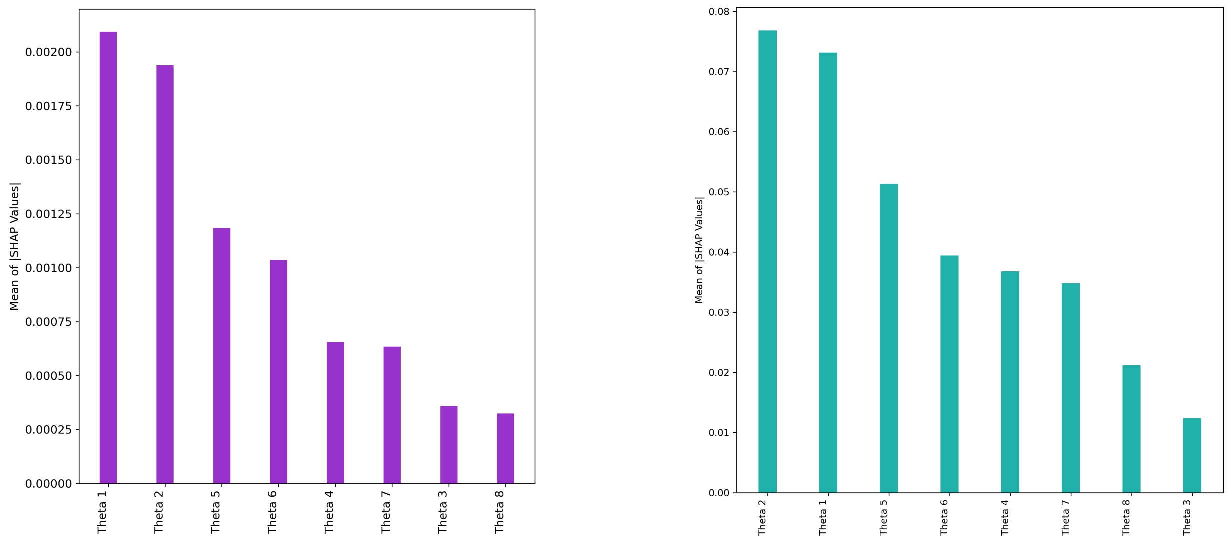
Figures 5a and 5b show the feature importance ranking for the FNN and the KAN for the HTGR application, respectively. Like the CHF application, both the FNN and KAN feature importance rankings show general agreement, with a few discrepancies. Notably, the first two features (Theta 1 and Theta 2) and the last two features (Theta 3 and Theta 8) are flipped for the two models, with the FNN ranking Theta 1 as the most important feature and Theta 8 as the least important feature. Again, this discrepancy could be due to both a difference in SHAP methods, as we used DeepLIFT for the FNN analysis and KernelSHAP for the KAN analysis, or due to model differences themselves. Nonetheless, these differences are very minor and overall show agreement between the feature importances for both models. The ranking is expected to have drums 1 and 2 worths, which are located in the first quadrant to be the most important to predict the neutron flux in that quadrant.

One of the key strengths of KANs is their ability to generate symbolic equations that illuminate the relationship between inputs and outputs. These equations are particularly helpful for applications that demand more trustworthy models. FNNs do not provide such explicit representations. Still, with ad hoc explainability methods, we can evaluate the importance each input has the model output, providing some transparency for FNNs and enhanced transparency for KANs.



(a) FNN (b) KAN

Figure 4. Feature importances for CHF prediction from the CHF dataset using an FNN vs a KAN.



(a) FNN (b) KAN

Figure 5. Feature importances for neutron flux in the first quadrant prediction from HTGR dataset using an FNN vs a KAN.

6. CONCLUSION

The comparative analysis of FNNs and KANs for two distinct nuclear applications underscores a clear trade-off between training efficiency, model accuracy, explainability, and interpretability, with each network type offering distinct advantages and limitations suited to different aspects of nuclear modeling. KANs perform better in problems where computational efficiency and model transparency are crucial, while FNNs are better suited for applications demanding higher accuracy and flexibility. Both approaches present opportunities for addressing complex challenges in nuclear engineering, with KANs demonstrating significant potential to reduce training times and enhance interpretability, along with some tradeoff in accuracy. The choice between FNNs and KANs depends on the specific needs and priorities of the researcher, balancing speed, precision, and interpretability. Future work could focus on hybrid approaches that combine the strengths of both networks or explore innovative methodologies to further optimize this balance, advancing the field of nuclear modeling in both accuracy and interpretability. Future work could also consider automating a process for hyperparameter tuning KANs to improve speed in development and model accuracy.

ACKNOWLEDGEMENTS

This work is funded by the U.S. Nuclear Regulatory Commission's University Nuclear Leadership Program for Research and Development (Award: 31310024M0013). Additionally, the second author (N. Panczyk) received sponsorship through the National Science Foundation's Graduate Research Fellowship Program (Grant Number: DGE 2241144).

REFERENCES

[1] Z. Liu, Y. Zhang, and H. Wang. "A Comprehensive Survey on Kolmogorov Arnold Networks (KAN)." *arXiv preprint arXiv:240711075* (2023). Available at https://arxiv.org/abs/2407.11075.

[2] A. N. Kolmogorov. "On the Representation of Continuous Functions of Several Variables as Superpositions of Continuous Functions of One Variable and Addition." *Doklady Akademii Nauk SSSR* (1957).

[3] G. Cybenko. "Approximation by superpositions of a sigmoidal function." *Mathematics of Control, Signals, and Systems* (1989).

[4] A. Kolmogorov. "On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables." *Proceedings of the USSR Academy of Sciences* (1956).

[5] F. Girosi and T. Poggio. "Representation Properties of Networks: Kolmogorov's Theorem Is Irrelevant." *Neural Computation*, **volume 1**(4), pp. 465–469 (1989).

[6] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark. "KAN: Kolmogorov-Arnold Networks." (2024). URL https://arxiv.org/abs/2404.19756.

[7] P. A. Myers, C. Craig, J. Cooper, V. Joynt, and M. I. Radaideh. "Pymaise: A Python Platform for Automatic Machine Learning Development and Benchmarking for Nuclear Engineering Applications." *Available at SSRN 4924326*.

[8] D. Price, M. I. Radaideh, and B. Kochunas. "Multiobjective optimization of nuclear microreactor reactivity control system operation with swarm and evolutionary algorithms." *Nuclear Engineering and Design*, **volume 393**, p. 111776 (2022).

[9] D. Groeneveld. "Critical Heat Flux Data Used to Generate the 2006 Groeneveld Lookup Tables." Technical Report NUREG/KM-0011, U.S. Nuclear Regulatory Commission (2019). URL https://www.nrc.gov/reading-rm/doc-collections/nuregs/knowledge/km0011/index.html.

[10] J.-M. L. Corre, G. Delipei, X. Wu, and X. Zhao. "Benchmark on Artificial Intelligence and Machine Learning for Scientific Computing in Nuclear Engineering. Phase 1: Critical Heat Flux Exercise Specifications." *NEA Working Papers* (2024).