EXPLAINABLE ARTIFICIAL INTELLIGENCE MODELLING OF A COMBINED CYCLE POWER PLANT USING KOLMOGOROV-ARNOLD NETWORKS

Paul-Vasile VEZETEU [1], Dumitru-Iulian NĂSTAC [2,*]

Artificial Intelligence Explicability (XAI) is an essential research field which aims at developing methods and models which can explain the decision process of machine learning (ML). Interpretable ML is a subfield concerned with developing algorithms which are intrinsically explainable. The current paper developed a state-of-the-art Kolmogorov-Arnold Network (KAN) which was recently published as a potential method of replacing ANNs in applications which require high interpretability. Our developed KAN system is designed to model a combined cycle power plant (CCPP) to test the capabilities of KAN to be implemented for real life applications. Our work will describe the methods of implementation, discuss the results obtained and conclude on best practices and pitfalls of such systems.

Keywords: Kolmogorov-Arnold networks, explainable AI, interpretable machine learning, combined cycle power plant modeling

1. Introduction

Explainable Artificial Intelligence (XAI) is a rapidly growing field of research [1][2][3][4] aiming to shed some light into the black box principle which currently governs the field of AI. Considering the wide spread of intelligent systems, and their involvement in high stake fields, such as healthcare [5] [6], finance and banking [7], law and criminal justice [8], manufacturing and energy [9] [10], the question of how such systems make decisions becomes crucial from a societal perspective. A diagnosis tool for cancer detection may mislead doctors into confirming an incorrect diagnosis, potentially leading to death, an energy forecasting system which incorrectly estimates future consumption may lead to financial losses, and an AI legal advisor or co-pilot can misguide lawyers, resulting in reputation loss and wrong sentencing. Additionally, apart from experts benefiting from this technology, AI integration into consumer products and services requires people to learn how to interact and communicate with the intelligent systems, reframing competency requirements [11], as well as trusting their reliability and

[1] PhD Candidate, Eng., Faculty of Electronics, Telecommunications and Information Technology, National University of Science and Technology POLITEHNICA of Bucharest, e-mail: paul-vasile.vezeteu@upb.ro.

[2] * Prof., Faculty of Electronics, Telecommunications and Information Technology, National University of Science and Technology POLITEHNICA of Bucharest, e-mail: iulian.nastac@upb.ro (corresponding author).

reasoning. This has proven to be significantly challenging, leading to the development of new regulatory and technical frameworks for developing, integrating and using AI [12] [13]. In this regard, questions of ethics and trust remain central to the field of artificial intelligence, creating boundaries on how one can develop and innovate such systems.

From a technical perspective, trustworthy AI, and in turn trustworthiness [1] [4] [14], becomes a crucial development requirement for future intelligent systems. Explainable AI covers three main characteristics which support trust and accountability, bias detection, collaboration and scientific advancements [14]. Transparency is the first step and should be made available, in general, for all scientific research. Providing documentation about the system, its architecture, parameters and optimization procedure, ensures compliance and accountability. Interpretability [1] [15] refers to models which are intrinsically explainable, and which provide visibility into its reasoning. This means one can draw conclusions with respect to the input-output relationship (e.g., feature importance, mapped patterns) by analyzing its internal parameters (e.g., weights, functions, connections). Explainability, on the other hand, describes a methodology which aims at explaining any type of model, including ANNs and deep learning models, by developing techniques to analyze the output of the black box.

While still in its early stages, there is a significant amount of scientific literature which explores and proposes new methods and algorithms, tests assumptions and integrates XAI into the broader AI framework [1] [2] [3] [4]. Nevertheless, many methods still lack maturity, are difficult to apply to state-of-the-art AI algorithms or are not sufficiently tested on real-life use cases and datasets. For this reason, the scientific community encourages research and exploration to fill in the knowledge gaps and bring the field of XAI closer to the demand of such systems.

In this regard, the field of XAI either studies or contributes to the development of inherently interpretable machine learning models or provides new techniques for post-hoc explicability of the developed intelligent systems [1]. Between the two approaches, interpretability is preferred [15] as it encourages the development of AI systems which, through their own architecture and inner workings (explicable by design), allow experts to determine how the output was formed and what are the captured patterns that describe the relationship between features and output. Examples of interpretable linear models include linear regression, sparse models and generalized models. Linear regression is highly interpretable considering that the output is computed as a sum of weighted inputs. On the other hand, sparse models take advantage of regularization principles in order to drive irrelevant features to zero which simplifies the model and selects the core features. Generalized models, such as additive and linear models, allow either the modeling of non-linear relationships between features and output, or can map

distributions other than the normal distribution (e.g. Poisson). What makes these models interpretable is the fact that their coefficients, adjusted after training, are interpretable. Another interesting class of interpretable models are decision trees and rule-based modeling as the chain of decisions can be easily followed.

Artificial neural networks (ANNs) are constructs of artificial neurons which are used to model highly non-linear relationships between inputs and outputs of a real system. Compared to regression models that we have previously discussed, each artificial neuron will compute the sum of the weighted inputs; however, it will also apply a non-linear function (e.g. Sigmoid, Tanh and ReLU) which supports the approximation of the original system. In this way, a fully connected neural network will be able to represent the non-linear space of features. Nevertheless, this also means that the data was now transformed into a new representation model which is not understandable by humans. While one can extract the adjusted weights of the model and know its architectural elements, one cannot interpret or draw relevant conclusions by looking solely at internal parameters. This is a consequence of how the information is mapped or captured by the weights. For example, neurons are not a one-to-one relationship with the input features, meaning that one neuron can map one or more features contained in the training data, characteristic which is called polysemanticity.

Kolmogorov-Arnold Networks (KANs) are a state-of-the-art approach to neural networks which replace the structure of adjustable weights and fixed activation functions, characteristic of the multilayer perceptron (MLP), with learnable one-dimensional functions and fixed nodes, whose role is to aggregate the signals coming from the edges [16]. As the name suggests, this network is based on the Kolmogorov-Arnold representation theorem [17] (or superposition theorem) which states that every multivariate continuous function denoted as

$$f: [0,1]^n \rightarrow \mathbb{R} \tag{1}$$

can be represented as a superposition of continuous single-variable functions. This can be mathematically represented through equation (2) which represents the base for KAN implementation.

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

While the idea of implementing this theorem for artificial intelligence algorithms was previously explored, it was Liu et al. [16] who proposed in 2024 a generalized version of KANs which can be implemented and trained under today's deep learning context. Without going too much into detail about the mathematical instruments which create the foundation of this technique, as it falls outside the

scope of this paper, KANs aim at approximating the edge functions of the network during training, whose aggregation results in the approximation of the transfer function of the target system [16]. In other words, the training process aims at finding the one-dimensional functions which together approximate the function modeling the input-output relationship. In this regard, the edge functions are approximated using B-spline functions whose coefficients serve as the model's learnable parameters.

From an interpretability standpoint, KANs are a form of interpretable neural networks, as the intrinsic parameters of the model characterize functions which can model the system [16]. In other words, by looking at the edges of the network one can determine relationships between inputs and outputs, as well as to determine why the model behaves in a certain way during training, allowing for better optimization. While this type of network is promising and a good competitor to the MLP there are still research gaps which need to be studied. For example, the practical implementation is still nascent and requires work to grow into a clear development framework. Additionally, the original paper [16] presents a proof of concept which is tested on synthetic datasets leaving the KANs implementation open for more complex real-life datasets.

The current paper focuses on implementing a state-of-the-art Kolmogorov-Arnold Network which can model the transfer function of a Combined Cycle Power Plant (CCPP) [18] [19]. Such systems are an essential part of the power grid ensuring that the power levels remain balanced. Due to their fast start time and high yield, CCPPs are used to compensate for energy fluctuations caused by less reliable energy sources such as wind, solar and hydroelectric. Additionally, because they are a cleaner energy source than other combustion-based power plants, they are mounted closer to the population centers, reducing the cost and loss of energy transportation. To train, test and optimize our model we use four types of environmental factors, measured over a period of 6 years [20]. In this way, the KAN can provide information with respect to the net hourly electrical energy output. This allows operators to monitor CCPP's performance in real-time, supporting them in optimizing the power plant's energy production. Other applications include anomaly detection, helpful for maintenance planning or predictive maintenance, or simulation of scenarios and grid integration planning. From a methodology perspective, the data is pre-processed and fed into the network, whose architecture is optimized gradually based on the model's interpretability characteristics and computed error metrics.

In terms of results, the optimized intelligent system was able to approximate the original CCPP with high confidence and to achieve performance similar to the one of artificial neural networks. During training the model parameters had to be adjusted multiple times to ensure the correct modeling of the original system. All the methods, experiments and results will be discussed in detail in the next sections;

however, the test metrics show that KANs have great potential to challenge the classical machine learning perceptron (MLP) when interpretability is a key requirement for the application. Nevertheless, drawbacks such as increased training time and difficulty in scaling were also identified during system development and will be discussed to offer the reader a comprehensive perspective over KAN implementation and evaluation.

2. Methods

This section focuses on describing the methods used for the practical implementation of the Kolmogorov-Arnold network for the modelling of a Combined Cycle Power Plant [18] [19]. The stages of data preparation and preprocessing, as well as the parametrization of the intelligent system for training purposes will be presented.

2.1 Combined Cycle Power Plant dataset

Considering that the authors of the original paper [16], which introduced generalized KANs, used a synthetic dataset, we decided to train and optimize this model for a real-life application. We chose the Energy sector, considering our previous experience with AI in this field [21], as well as the importance of CCPPs for the stability of the power grid. Additionally, data needed to be reliable and suited for a continuous and nonlinear multi-variable function fitting application. Considering that KANs are more computationally expensive than the MLP we were interested in a smaller dataset which would be able to train in an acceptable amount of time on CPU. In this regard, the dataset is made available by UC Irvine Machine Learning Repository [20]. Composition wise, the dataset contains 9568 instances of measurements having four different features, such as temperature, ambient pressure, relative humidity and exhaust vacuum to predict the net hourly electrical energy output, characteristics which fit our system design requirements.

2.2 General Architecture of the KAN System

For the practical implementation of the model, a series of stages were defined to propagate the data, train and optimize the model. The overview of the system can be seen in Fig. 1 and will be presented in this sub-section of the paper. After importing the CCPP dataset into the Python workspace, data had to be split into a training, validation and testing set. An important mention is that the pykan library [16], referenced by the authors in their paper, does not support a validation set for the training of the model. Considering that a validation set is crucial for overfitting prevention, and model evaluation, we initially split the data into a training set and a temporary set based on the 80% – 20% rule. After that, the temporary set was divided again with a ratio of 0.25 into a testing set and validation set. This gave us 7654 datapoints for training, 1435 for testing and 479 for

validation. Also, we have used an 85% – 15% split and a 0.33 ratio for extracting the validation data which slightly improved results due to more data being assigned for training. While it may seem insufficient, the data proved to be enough both to capture the transfer function of the CCPP, and to test and optimize the KAN intelligent system. For the training phase we fed the training set and the validation set into the model, instead of the test set, and kept the test set for further error metrics calculation and plotting.

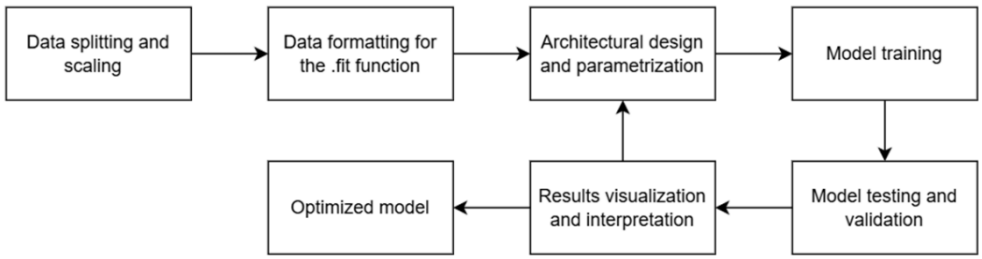

Fig. 1. Overview of the developed KAN system

Data scaling is a must, as the spline functions used to approximate the 1D-functions on the network's edges work only in a fixed grid range of [-1, 1]. In the original documentation of the paper, the authors do data normalization as part of preparing the features for training. In our work, we have initially standardized the data sets and were able to obtain noticeable results after optimization. For the results section, the algorithm uses data normalization to improve stability. In this regard, there are some considerations which must be taken into account when deciding between the two scaling methods. First, because the spline functions are defined between -1 and 1, standardizing the data, which brings the datapoints approximatively between -3 and 3, leads to the spline functions potentially losing some information during the approximation stage. Additionally, normalization squeezes the data into a tighter interval which requires an increase of the grid parameter during training. This parameter describes how many local regions each edge can be divided into. In theory, the more intervals the smoother the approximation. However, considering that for each interval we have an associated edge coefficient, a large number will create a chaotic gradient space affecting how the optimization algorithm, our case LBFGS, converges. In this regard, the best testing results were obtained with a grid between 20 and 30, with a larger value required when normalizing the data, compared to standardization. Additionally, selecting a grid value which is too small will result in failure of the training procedure because the fit library function [16] will not be able to correctly compute the coefficients. Adjusting the grid corrects the computational problem and allows the training of the model to be completed. Overall, in our work we have varied the

grid parameter between 15 and 30 with an increment of 5, as smoother changes did not yield significant differences in the results.

In the next stage of the system, data formatting is also important for the training process. By default, the training function requires data to be structured under the format of a Python dictionary and cannot be parsed as Pandas dataframes which are characteristic to the tensorflow MLP training function. A dictionary will pack four tensors of torch type, which are characteristic of the Torch package, corresponding to the features and associated labels. For this paper, we have created one dictionary for training purposes which contains the training and validation data, instead of the testing set. Later in our work we converted the test set (features and associated labels) to torch tensors for compatibility reasons.

When designing the architecture of the KAN model one can define the number of inputs, hidden and output nodes, as well as the number of layers, similar to the design of an artificial neural network. However, it is important to remember that nodes are used only to aggregate signals, except for the input nodes which receive and propagate the data further into the network, while the edges between them are associated with the 1D functions to be approximated. Fig. 2 shows how the models look when plotted after initialization. This allows us to take advantage of the interpretability of the model and facilitates its analysis.

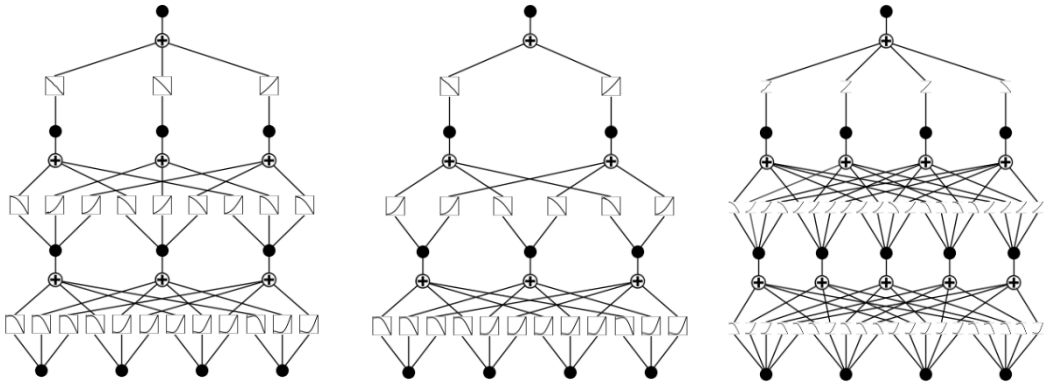

Fig. 2. KAN architectures before training

As Fig. 2 shows, all models have four input nodes associated to each one of the four input features of the CCPP dataset and one output node for the predicted value. In terms of architecture, a manual grid search was performed as follows: the number of hidden layers were varied between one and two, while the number of neurons were varied between one and eight. Additionally, larger architectures (e.g. three hidden layers) were randomly selected and then tested to see how they perform in comparison to smaller networks. If poor performance was observed

around certain regions of the search space, we increased our increment by 2. On the contrary, better results were followed by a more in-depth investigation, and the training was repeated up to three times. We have chosen manual search over automatic as in our initial experiments we observed that some architectures would result in the non-convergence of the coefficients and thus interrupted the search process. Given the fact that KAN performs well with small architectures manual search proved to be sufficient to identify multiple well-performing models with stable performance across the validation and test set.

During model initialization we must set the k variable which refers to the degree of the B-spline functions. This parameter directly contributes to the smoothness of the function approximation and determines if the functions used will be constant, linear, quadratic or cubic. The system we developed implemented cubic coefficients as it provided a nice balance between performance, measured via the error metrics, and training time, which averaged around 3-5 minutes per model.

During the training process the data is fed into the model. For the optimization algorithm we have used the default Limited-Memory BFGS (L-BFGS) instead of Adam which is more popular and frequently used in ANN training. The chosen algorithm is a quasi-Newton optimization algorithm and was selected due to its converging speed in the context of KAN. For this paper, a performance comparison was not performed but will be considered for future publications in order to get a quantitative measure of the differences with respect to this type of network. Considering the inner workings of L-BFGS algorithm, the data is fully fed into the model and does not involve batches. This means that during training one step is equivalent to one epoch in the case of Adam which takes into account a batch parameter. For our experiments, we varied the number of steps between 10 and 30 with an increment of 5, as it allowed enough time for the KANs to converge. This was concluded based on the validation vs training error graph. If additional training time was needed, we increased the number of steps up to 70, however such cases were rare.

When it comes to overfitting, the training function is equipped with six different regularization parameters which were used for our system training and optimization. In this regard, the overall penalty strength can be controlled by using a lambda parameter, as part of the loss computations. Another regularization parameter allows the user to control the L1 penalty which encourages some of the spline coefficients to go to zero resulting in simpler and interpretable results. Lambda entropy, on the other hand, encourages diversity and decisiveness. The principle is that, during training, edge functions try to approximate the real functions which form the transfer function of the modeled system. In this regard, lambda entropy encourages variation and shape instead of flat approximations which carry little to no information. Similar to L2 regularization, the lambda coefficients' parameter penalizes large coefficients of spline functions. This creates

stability by encouraging smooth scaling. Similarly, the training function also allows the system to penalize large differences, but in this case between neighboring spline coefficients controlling the oscillations of the trained model. Additionally, the user can decide how to apply the regularization penalties inside the KAN. When designing the intelligent system, we have decided to maintain the default option of regularizing the forward spline functions using normalized coefficients. Overall, the regularization parameters needed to be carefully selected to prevent model overfitting, while also choosing them to still allow for the model to learn the patterns of the dataset. In our experiments, we have initially tried different configurations with regularization parameters varied as follows: lambda general between 10^{-4} to 10^{-1}, lambda L1 between 10^{-3} to 10^{-1}, lambda entropy between 1 and 25, and lambda coefficients between 10^{-3} to 10^{-1}. It is important to mention that for this search process we did not try all the combinations, the goal being to identify which of the parameters described have a positive effect on the KAN performance and around which intervals. Based on this initial search we have concluded that only lambda general, lambda entropy and lambda coefficients are needed. In our manual grid search we kept lambda entropy and lambda coefficients constant with values of 1 and 0.01 and chose lambda general as 0.0005, 0.005 or 0.05 depending on the validation vs training error graph (e.g. larger architectures are prone to overfitting and require stronger regularization).

Once the training process was completed, the model was plotted, to see how the edge functions were approximated. Nevertheless, while KANs are described as interpretable due to their functional representation, the main objective of the current paper was to test the predictive performance of these models rather than to assess their interpretability. Thus, the spline visualizations are presented as qualitative illustrations. Each model was evaluated based on the training versus validation loss graph, as well as computed error metrics. This ensured that the selected models could generalize on unseen data. To get a sense of the magnitude we have considered the mean squared error, the root mean squared error and the mean absolute error. Additionally, we have computed the R2 score which shows us how much information was assimilated by our model. These metrics were computed on the training set and were considered sufficient to determine the performance of the model. For visual representation, we have also plotted the predicted versus actual net hourly electrical energy output of the CCPP. This graph shows how well the points are distributed against a perfect predictor. The larger the scatter the less reliable the model is when doing predictions based on test features. Once a model is evaluated and interpreted, its parameters are adjusted to find the optimal configuration.

Considering that KANs are a potential competitor for ANNs we trained fully connected feed-forward artificial neural networks with Adam for the optimization algorithm and ReLU for the neuron's activation function. All models were trained

on the same data splits as KAN models to ensure a fair comparison. During training we have used a similar manual grid search methodology. Initially, we trained ANNs with one hidden layer and neurons between one and eight. However, the training time was very large (the ANNs were converging slowly and required up to 1000 epochs). In this regard, we have decided to increase the number of hidden layers to two and vary the neurons between 8 and 32. Training was performed for up to 200 epochs using a batch of 32. L2 regularization was applied to all hidden layers with the parameter being 0 (no regularization), 1e-4 or 1e-3. For the testing phase we have computed the same errors as with KAN to allow for direct comparison.

3. Results and discussion

This section will summarize the best rated models based on computed errors, as well as training vs validation loss graph. To provide the readers with a complete picture of the testing process, the results will be discussed. In this regard, we will provide insights into the most important model parameters and how they relate to each other, pitfalls and best practices. The selected results were summarized in Table 1.

Table 1

Highest rated KAN systems based on computed errors									
No.	Architecture	Splitting Ratio	Grid	Steps	Regularization			RMSE	R2
					General	Entropy	Coeff		
1	[4 5 1]	0.2 / 0.25	30	12	0.0005	1	0.01	3.9391	0.9463
2	[4 3 3 1]	0.2 / 0.25	30	30	0.0005	1	0.01	3.9941	0.9448
3	[4 5 1]	0.2 / 0.25	30	30	0.0005	1	0.01	4.0201	0.9441
4	[4 3 1]	0.2 / 0.25	30	30	0.0005	1	0.01	4.0440	0.9435
5	[4 3 3 2 1]	0.2 / 0.25	25	20	0.005	1	0.01	4.2149	0.9386
6	[4 5 5 1]	0.2 / 0.25	30	30	0.0005	1	0.01	4.2488	0.9376
7	[4 8 1]	0.15 / 0.33	20	30	0.0005	1	0.01	3.7344	0.9543
8	[4 5 1]	0.15 / 0.33	15	30	0.0005	1	0.01	3.9091	0.9542
9	[4 7 1]	0.15 / 0.33	15	30	0.0005	1	0.01	3.8242	0.9521
10	[4 6 1]	0.15 / 0.33	20	30	0.0005	1	0.01	3.8692	0.9510
11	[4 4 3 1]	0.15 / 0.33	30	25	0.005	1	0.01	3.9065	0.9500
12	[4 2 1]	0.15 / 0.33	25	25	0.0005	1	0.01	3.9434	0.9491
KAN system parametrization which resulted in failed coefficients calculations									
1	[4 5 3 1]	0.2 / 0.25	30	30	0.0005	1	0.01	-	-
2	[4 3 3 2 1]	0.2 / 0.25	30	20	0.005	1	0.01	-	-
3	[4 3 3 1]	0.15 / 0.33	20	30	0.0005	1	0.01	-	-

Based on the above table, one can see the optimized models which provide the best error results. During model evaluation we used the methods discussed and analyzed the plots and diagrams represented in Fig. 3 which shows the KAN model's edges and nodes after training, the training versus validation graph and the scatter plot of predicted values vs real outputs of the CCPP, referenced to a perfect predictor.

Fig. 3. Examples of evaluation graphs obtained during evaluation

During this process we observed that model architecture is not necessarily a definitory parameter for the performance of the KAN system used to model a CCPP. More exactly, both smaller and denser architectures resulted in good error metrics. Nevertheless, architecture influences the choice of other parameters. For example, larger architectures tend to work better with a larger grid parameter, while a smaller number of layers and nodes can perform with a smaller grid. This is directly related to how KANs work intrinsically. The denser the network, the more 1D functions will be used to approximate the system. If the grid is too low, the model will not be able to capture sufficient information, will be poorly parametrized (from the perspective of the spline coefficients), resulting in high test errors. On the other hand, if the grid is too small or too large for the current model configuration the training will fail completely (see Table 1 for examples), as the optimization algorithm cannot fully compute the coefficients, or the error will increase significantly, making the model unusable.

Another crucial factor is the regularization parameters. The choice of parameters was made based on the training versus validation loss, as well as the errors obtained. If the distance between the validation and training plot is too large the model cannot generalize. Fig. 3 contains such a graph (center) and shows how the two plots are close to each other. Additionally, because the model is explainable, we can see how the one-dimensional functions look after the training (Fig. 3, left diagram) and increase or decrease the lambda parameters accordingly to smoothen out the curves. As can be observed from Table 1, for the final models, we only kept the lambdas of the overall model, entropy and coefficient in our control and set

them around specified values. The regularization parameters need to be carefully chosen as they strongly affect the learning capacity of the model. If nonexistent, some models overfit and thus are unreliable during testing. On the other hand, too large lambda values will prevent the model's learning or will produce large variations in the coefficients.

Another important consideration is evaluating the model on a test set. In this regard, the training versus validation loss graph may not show overfitting, but with large errors the model will perform poorly. For this reason, Table 1 contains error metrics which describe the model's performance on test data. While the scores presented are high, during the exploration phase, we also obtained low R2 scores which were below 0.75. As can be observed, the best error metrics, computed based on the top 5 performing models, stop at a mean R2 of 0.9523 and a mean RMSE of 3.8486. Varying the configurations did not raise the score higher than 0.9543, so we suspected that more data would allow the KAN system to learn new information. As we can see from Table 1, providing more data during training slightly improved the model's performance. Unfortunately, no significant increase was noticed, prompting us to conclude that for a KAN system, this was the highest percentage of learnable information with respect to the CCPP's dataset noise.

By analyzing scientific literature, we have identified papers which explore the integration of KANs for the energy sector by using various interesting and diverse approaches. In their paper, Bagrow and Bongard [22] implement a multi-exit KAN which is trained and tested both on toy datasets, as well as real-world data. For the latter, the same CCPP dataset was used as in our paper. In terms of results, they obtained a RMSE of 4.406 and R2 of 0.935 on their best multi-exit architecture with a structure of [4, 4, 4, 1] where the output was set after the second layer. Considering our evaluations, our models perform slightly better on the same dataset. The difference in result most likely happens because of our regularization procedures which increase the robustness of our model. This is also indirectly mentioned in the original paper of the authors [22] who explain that their method is an alternative to simplify models through architectural optimization rather than direct regularization. In terms of the overall methods, we consider their multi-exit KAN approach to be an elegant alternative solution for simplifying and optimizing intelligent systems from an architectural perspective.

A more complex approach on KAN implementation aimed at introducing KANs into an evolutionary rule-based machine learning framework called XCSF [23]. This approach searches to solve optimization problems such as cases where the function either presents local complexities or discontinuities [23]. To evaluate their models the authors also used the CCPP dataset, among others, but they evaluated their systems via mean absolute error (MAE) obtaining a MAE of 0.0868. Additionally, they normalized inputs between [0, 1] and the target value to [-1, 1]. This brings differences between our methods and theirs, making comparisons not

possible. Considering the importance of assessing results and evaluating performance with other publications, we have decided to compute the MAE for a new KAN model but maintaining our methods for consistency. In this regard, the model of choice has a [4, 3, 3, 1] architecture with a grid value of 30. In terms of training, we used the LBFGS optimization algorithm over 30 steps and a regularization configuration of [0.0005, 1, 0.01] considering the order and parameters in Table 1. The results of our model on the test set are as follows: RMSE of 4.0165, MAE of 3.1797 and R2 of 0.9472. By de-normalizing their target value [23] we obtain an approximative value for MAE of 3.2767. This proves that our models are highly relevant and provide reliable performance on the test data when compared to other papers. With respect to the X-KAN models they show a slight improvement of the results [23] proving their capabilities and benefits of integrating KANs with evolutionary algorithms for optimization.

To test how KANs perform in comparison to ANNs we have performed a series of experiments using the same training, validation and test sets split. The results and parameters can be seen in Table 2.

Table 2

Highest rated one hidden layer ANN systems based on computed errors							
No.	Architecture	Splitting Ratio	Epochs	Batch Size	L2 Regularization	RMSE	R2
1	[4 7 1]	0.15 / 0.33	700	64	0	4.1498	0.9432
2	[4 7 1]	0.15 / 0.33	700	64	1e-03	4.3041	0.9389
3	[4 3 1]	0.15 / 0.33	600	64	1e-04	4.3108	0.9387
4	[4 5 1]	0.15 / 0.33	700	64	1e-04	4.3138	0.9386
5	[4 1 1]	0.15 / 0.33	200	32	0	4.3359	0.9380
Highest rated two hidden layer ANN systems based on computed errors							
1	[4 8 8 1]	0.15 / 0.33	200	32	0	4.1879	0.9422
2	[4 8 16 1]	0.15 / 0.33	200	32	0	4.3531	0.9375
3	[4 16 8 1]	0.15 / 0.33	200	32	1e-03	4.3300	0.9382
4	[4 16 16 1]	0.15 / 0.33	200	32	1e-03	4.1100	0.9443
5	[4 32 16 1]	0.15 / 0.33	200	32	1e-04	4.3118	0.9387

By comparing Table 1 and Table 2 one can conclude that KANs achieve comparable performance to ANNs, on the CCPP dataset, thus showing that KANs represent a competitive modeling alternative for low-dimensional regression problems in cases where interpretability is a requirement. In this regard, while not many, there are other research who compare the performance between the two. From a hyperparameter perspective, a comparative study between KAN models and the classic multi-layer perceptron (MLP) performance in smart grids (battery capacity prediction) [24], concluded that a KAN model with a lower number of

parameters tends to perform better than the denser models, as well as the MLPs. This statement proves to be different from what we have observed during our experiments. While we cannot compare our results directly, considering the difference in dataset selection, we can notice the difference in the choice of the grid parameter. In our work we used much larger grid values, between 15 to 30, to smoothen out the functions to be approximated, while the authors of the paper [24] set a grid of 3. From this observation we can draw two different arguments. Firstly, the smaller grid parameter is not sufficient for larger architectures to learn the non-linearities of their dataset during the training process. This produces difficulties with the optimization procedure which in turn leads to overfitting, and thus smaller architectures tend to perform better. The second argument is that their dataset may not be highly complex and a smaller architecture is sufficient to capture the overall information trend. If this is the case, a larger architecture only brings complexity to the training process which decreases the performance and it is not required for their data. While we cannot say which of the two is the reason for the difference in our observations, these arguments are aligned with our emphasis on the importance of selecting the optimum grid value, as well as the importance of regularization when increasing the model complexity. Nevertheless, their comparison of KANs with MLPs is an important indicator of the potential that such interpretable models have, and it is highly valuable considering KANs are presented as a possible replacement.

Similarly, another paper which explores the modeling and optimization of energy systems [25], both for KANs and MLPs, notices the high-performance output of the KAN when using a thermal powerplant dataset for their experiments. This strengthens the argument that KANs are a valuable choice when it comes to applications in the energy sector. From another view, a particularity of their paper [25] is the use of Adam as an optimization algorithm and the use of the Pearson correlation coefficient (PCC). As explained in our methods section, we have chosen to use LBFGS for convergence reasons and set to consider Adam for future studies. However, seeing that Adam was successful in optimizing their model supports our initial assumptions. While the use of PCC for the loss function did not yield a significant improvement in results, we consider that their approach [25] is an interesting variation, adding to the understanding of KANs.

3. Conclusions

AI explicability is a central field which develops methods and algorithms to explain the reasoning behind machine learning algorithms. In this regard, contributing to the evolution of this research area proves to be crucial as the social context demands new development requirements and adherence to legal and regulatory frameworks. As part of the larger field, AI interpretability is the preferred solution when developing intelligent systems considering that their intrinsic

structure allows us to decode information for our own representation and understanding. This not only supports trust and accountability, but also contributes to scientific advancement, as the models can be better optimized based on their own transparency. This paper developed a state-of-the-art Kolmogorov-Arnold Network, which changes the classical framework of the ANNs by approximating the systems function through aggregated 1D B-spline functions estimated during training. To take a step further from the original paper we trained the KAN system to model a Combined Cycle Power Plant, application which has great utility in real applications for predictive maintenance, simulations and optimization. By developing a KAN system on a more complex dataset, we further proved the potential of such models to replace the MLP where explicability is a critical factor. In terms of results, we optimized the model to obtain high approximation of the CCPP with a mean R2 of 0.9523 and a mean RMSE of 3.8486. To test the performance of KANs against ANNs we have performed additional experiments using the same training, validation and test sets. This showed that KANs have indeed potential to be used in low-dimensional regression problems where interpretability is required. Additionally, we have seen that our models are reliable and relevant in comparison with other similar publications, supporting our assumptions and drawn conclusions. For future developments, we consider that an automated search procedure to analyze a larger parametrization space would be beneficial reflecting that KANs remain sensitive to hyperparametrization and data scaling. Also, we are interested in exploring further the interpretable nature of such systems. While interpretability is a central characteristic of KANs, our paper did not focus on quantitatively analyzing or comparing them to other similar systems, nor providing guidance on how their transparency can be used by power plant operators. This remains a subject for our future research in this field. Overall, we encourage exploration in this field considering that KAN methods are still nascent, but proved to have tremendous potential for real applications, while also interpretability gains more traction due to the changing requirements for AI integration.

REFERENCES

[1] *S. Ali et al*., Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence, Information Fusion, Vol. **99**, 2023.

[2] *D. Minh et al*., Explainable artificial intelligence: a comprehensive review, Artificial Intelligence Review, Vol. **55**, p. 3503–3568, 2021.

[3] *P. Linardatos, V. Papastefanopoulos and S. Kotsiantis*, Explainable AI: A Review of Machine Learning Interpretability Methods, entropy, Vol. **23**, Iss. 1, 2021.

[4] *P. Angelov et al*., Explainable artificial intelligence: an analytical review, WIREs Data Mining and Knowledge Discovery, Vol. **11**, Iss. 5, 2021.

[5] *F. Jiang et al*., Artificial intelligence in healthcare: past, present and future, Stroke and vascular neurology, Vol. **2**, Iss. 4, 2017.

[6] *P.V. Vezeteu, A.D. Andonescu and D.I.Năstac*, A Literature Review on Artificial Intelligence in Dermatological Diagnosis and Tissue Microscopy, IEEE Photonics Journal, Vol. **17**, pp. 1-14, 2025.

[7] *L. Cao*, AI in Finance: Challenges, Techniques, and Opportunities, ACM Computing Surveys (CSUR), Vol. **55**, Iss. 3, pp. 1-38, 2022.

[8] *A. Završnik*, Criminal justice, artificial intelligence systems, and human rights, ERA Forum, Vol. **20**, pp. 567-583, 2020.

[9] *R.X. Gao et al*., Artificial Intelligence in manufacturing: State of the art, perspectives, and future directions, CIRP Annals- Manufacturing Technology, Vol. **73**, pp. 723-749, 2024.

[10] *P.V Vezeteu and D.I. Năstac*, Fault Detection and Diagnosis of Rotor Broken Bars Using Artificial Intelligence, IEEE SIITME, pp. 177-184, 2023.

[11] *P.V. Vezeteu and D.I. Năstac*, Artificial Intelligence Integration in Business: Study of Employee Competences in Relation to the Organisational Needs, Amfiteatru Economic, Vol. **26**, Iss. 67, pp. 832 - 847, 2024.

[12] *European Union*, EU AI Act: first regulation on artificial intelligence, European Union, 2025. [Online]Available: https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence.

[13] *N.A.Smuha*, From a 'race to AI' to a 'race to AI regulation': regulatory competition for artificial intelligence, Law, Innovation and Technology, Vol. **13**, pp. 57-84, 2021.

[14] *B. Li et al*., Trustworthy AI: From Principles to Practices, ACM Computing Surveys, Vol. **55**, Iss. 9, pp. 1-46, 2023.

[15] *C. Rudin*, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature machine intelligence, Vol.**1**, Iss.5, pp. 206-215, 2019.

[16] *Z. Liu et al*., KAN: Kolmogorov-Arnold Networks, arXiv preprint arXiv:2404.19756, 2024.

[17] *J. Schmidt-Hieber*, The Kolmogorov–Arnold representation theorem revisited, Neural Networks, Vol. **137**, pp. 119-126, 2021.

[18] *E. Ersayin and L. Ozgener*, Performance analysis of combined cycle power plants: A case study, Renewable and Sustainable Energy Reviews, Vol. **43**, pp. 832-842, 2015.

[19] *V. Xezonakis et al*., Estimating power generation of a combined cycle power plant using artificial intelligence technique, Journal of Infrastructure, Vol. **8**, Iss. 16, 2024.

[20] *P. Tfekci and H. Kaya*, Combined Cycle Power Plant Dataset, UC Irvine Repository, 2014.

[21] *P. V. Vezeteu A. R. Morariu and D. I. Năstac*, Adaptive data predictions for the energy sector at national level, IEEE SIITME, pp. 292-297, 2021.

[22] *J. Bagrow and J. Bongard*, Multi-Exit Multi-Exit Kolmogorov-Arnold Networks: enhancing accuracy and parsimony, arXiv, 2025.

[23] *H. Shiraishi, H. Ishibuchi and M. Nakata,* X-KAN: Optimizing Local Kolmogorov-Arnold Networks via Evolutionary Rule-Based Machine Learning, arXiv, 2025.

[24] *N. Le, A. P. Ngo and H. T. Nguyen,* Kolmogorov-Arnold Networks for Supervised Learning Tasks in Smart Grids, 56th North American Power Symposium (NAPS), pp. 1-6, 2024.

[25] *T. Ansar, W. M. Ashraf,* Comparison of Kolmogorov–Arnold Networks and Multi-Layer Perceptron for modelling and optimisation analysis of energy systems, Energy and AI, Vol. **20**, 2025.