Computing and Software for Big Science (2025) 9:9
https://doi.org/10.1007/s41781-025-00138-3

RESEARCH

KAN We Improve on HEP Classification Tasks? Kolmogorov–Arnold Networks Applied to an LHC Physics Example

Johannes Erdmann[1] · **Florian Mausolf**[1] · **Jan Lukas Späh**[1]

Received: 8 August 2024 / Accepted: 28 April 2025

Abstract

Recently, Kolmogorov-Arnold Networks (KANs) have been proposed as an alternative to multilayer perceptrons, suggesting advantages in performance and interpretability. We study a typical binary event classification task in high-energy physics including high-level features and comment on the performance and interpretability of KANs in this context. Consistent with expectations, we find that the learned activation functions of a one-layer KAN resemble the univariate log-likelihood ratios of the respective input features. In deeper KANs, the activations in the first layer differ from those in the one-layer KAN, which indicates that the deeper KANs learn more complex representations of the data, a pattern commonly observed in other deep-learning architectures. We study KANs with different depths and widths and we compare them to multilayer perceptrons in terms of performance and number of trainable parameters. For the chosen classification task, we do not find that KANs are more parameter efficient. However, small KANs may offer advantages in terms of interpretability that come at the cost of only a moderate loss in performance.

Introduction

Classifying events as signal or background is a crucial ingredient of data analysis at collider experiments. At the Large Hadron Collider (LHC), separating small signals from large backgrounds is an omnipresent challenge. To achieve higher precision in the analysis of collider data, excellent classifiers are necessary. Machine-learning-based classifiers have a long history in high-energy physics (HEP). For example, the observation of electroweak production of single top quarks in 2009 at the Tevatron [1, 2] was aided by boosted decision trees and by shallow neural networks, i.e., multilayer perceptrons (MLPs) with one hidden layer. With the development of deep neural networks, MLPs with several hidden layers have been proposed for HEP classification tasks [3] and have become a standard tool for event classification, particle identification, fast simulations and many more applications at the LHC [4–8].

The strong performance of MLPs comes with a trade-off in terms of interpretability. Interpretability, i.e., the "ability to explain or to present in understandable terms to a human" [9], remains a challenge for MLPs, particularly for deep networks with many trainable parameters. At the same time, understanding what such a model has learned about the underlying physics is of genuine interest in physics applications. Several methods have been developed to address this challenge of explaining the outputs of MLPs for given input examples [10–12]. Techniques such as Shapley values [13] and permutation feature importance [14] are established methods for assessing the contribution of individual input features to the model output. Surrogate models, such as LIME [15], aim to explain the reasoning of complex architectures by approximating them with simplified models. On the other hand, approaches like Neural Additive Models [16] provide interpretability by constructing models that are transparent by design.

Recently, Kolmogorov–Arnold Networks (KANs) have been proposed as an alternative to MLPs [17]. While MLPs are grounded in the universal approximation theorem [18], KANs are motivated by the Kolmogorov–Arnold representation theorem [19]. The layers of the KAN have learnable activation functions on the edges that are summed on the nodes. In contrast, MLP layers use learnable weights on the edges as inputs to fixed activation functions on the nodes. Many approaches have been explored to improve the expressiveness and performance of MLPs by introducing learnable activation functions. Examples include implementations

✉ Florian Mausolf
florian.mausolf@rwth-aachen.de

[1] RWTH Aachen University, III. Physikalisches Institut A, Aachen, Germany

Published online: 22 May 2025

based on splines [20, 21], other parametric functions [22–25] or even neural networks [26] as activation functions.

While networks based on the Kolmogorov–Arnold representation theorem were proposed before [27–34], recently the capabilities of KANs in terms of performance and interpretability were highlighted [17]. In Ref. [17], KANs were found to have promising performance with a substantially smaller number of trainable parameters than MLPs. KANs offer advantages in terms of interpretability, complementarily to existing explainability methods applicable to both KANs and MLPs, due to their shallower structures with significantly fewer nodes than typical MLPs. Each edge in a KAN contains multiple trainable parameters that determine the shape of a single function. Therefore, an entire KAN can be represented as a comprehensive graph. In addition, the potential for interpretability by approximating the learned activation functions symbolically with a set of known functions was discussed. Ref. [17] has sparked active discussion on the potential advantages of KANs and their relation to MLPs [35–86].

We apply KANs to a typical HEP event classification task. As an example, we choose the binary separation of the associated production of a Higgs boson with a single top quark (tH) and with a top quark and an anti-top quark ($t\bar{t}H$) at the LHC, where the Higgs boson decays to a pair of photons ($H \to \gamma\gamma$). We study the interpretability of KANs for this classification task. In addition, we compare KANs to MLPs in terms of performance and parameter efficiency, where we use KANs and MLPs with different numbers of layers and nodes per layer. We document our findings in the practical training of KANs. To our knowledge, this is the first application of KANs to a task in particle physics.

Kolmogorov–Arnold Networks

For the comparison to KANs, we briefly summarize the concept of MLPs. An MLP consists of multiple layers of nodes, each connected to nodes in subsequent layers through weighted edges. The core component of an MLP is the fully connected layer, which holds the trainable parameters defining the strength of the connections between nodes of two layers. Each layer applies an affine transformation, represented by a weight matrix $\mathbf{W}$ and a bias vector $\vec{b}$, followed by an activation function $\mathcal{A}$. The transformation applied in each MLP layer can then be written as $\vec{y} = \mathcal{A}\left(\mathbf{W}\vec{x} + \vec{b}\right)$, where $\vec{x}$ denotes the input to the layer and $\vec{y}$ is its output. The activation function introduces non-linearity in the model and is a hyperparameter that has to be chosen. Common choices include the rectified linear unit $\mathrm{ReLU}(x) = \max(0, x)$, the logistic sigmoid function $\sigma(x)$, and the hyperbolic tangent function.

In contrast, KANs are inspired by the Kolmogorov–Arnold representation theorem, which states that any continuous multivariate function $f : [0,1]^n \to \mathbb{R}$ can be represented as a finite sum of continuous functions of only one variable. Formally, for any continuous real-valued function $f(x_1, x_2, \ldots, x_n)$, continuous functions $\phi_{i(j)}$ exist, such that

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{2n+1} \phi_i \left(\sum_{j=1}^{n} \phi_{ij}(x_j) \right), \tag{1}$$

where n is the number of variables that parameterize the multivariate function, and ϕ_i and ϕ_{ij} are univariate functions. This representation reduces the problem of approximating a multivariate function to a problem involving only univariate functions and the sum operation. The objective of the network training is to approximate these univariate functions $\phi_{i(j)}$.

Motivated by the theorem, the appropriate network architecture to approximate a multivariate function of n variables consists of two layers with n input nodes, $2n + 1$ hidden nodes, and a single output node. However, the authors of Ref. [17] generalized the concept by defining a KAN layer as a basic building block. As in MLPs, the number of nodes in these layers can be customized and layers can be stacked arbitrarily to enhance the performance of the model. Similar to MLPs, each node in a given layer is connected to each node of the subsequent layer. For each edge, an individual, learnable activation function is used. On the nodes, only the sum operation over all incoming edges is performed.

In the implementation of Ref. [17], the learnable activation functions are defined as the weighted sum of a B-spline, expressed by B-spline basis functions B_i, and a fixed residual function, chosen as the sigmoid-linear unit $\mathrm{SiLU}(x) = x \cdot \sigma(x)$:

$$\mathrm{activation}(x) = w_1 \cdot \mathrm{SiLU}(x) + w_2 \cdot \sum_{i=0}^{G+k-1} c_i \cdot B_i(x). \tag{2}$$

The weights $w_{1,2}$ and the basis-function coefficients c_i are the trainable parameters of the spline. The basis functions B_i are chosen as polynomials of degree k, with default value $k = 3$. The grid parameter G determines how many basis functions build the B-spline and serves as a hyperparameter of the KANs. Furthermore, the domain of the activation function needs to be chosen and can be updated several times during network training to match the input range of the activation function. Specifically, for given parameters k, G and the domain $[t_0, t_G]$, a vector $\vec{t} = (t_{-k}, \ldots, t_0, \ldots, t_{G+k})$ of equidistant knot points is constructed. Then, $G + k$ basis functions $B_i^k(x)$ are recursively defined:

$$B_i^0(x) = \begin{cases} 1 & \text{if } t_i \le x < t_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

and for $k > 0$:

$$B_i^k(x) = \frac{x - t_i}{t_{i+k} - t_i} B_i^{k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i+1}^{k-1}(x). \quad (4)$$

The basis functions are only non-zero over a portion of the interval, allowing the coefficients c_i to adapt and change the overall spline locally. This enables the approximation of functions without strong assumptions about their functional form. More details on the implementation can be found in Ref. [17].

Data Set

As an example for a typical HEP classification task, we use the separation of $t\bar{t}H$ from tH production in the $H \to \gamma\gamma$ decay channel at the LHC. Both processes offer complementary sensitivity to properties of the Yukawa coupling of the top quark [87, 88], but only $t\bar{t}H$ production has been observed so far [89, 90]. In the search for tH production, the $H \to \gamma\gamma$ decay offers one of the most sensitive channels [91, 92], for which $t\bar{t}H$ is a major background and hence excellent binary classification is necessary.

We simulate $t\bar{t}H$ and tH production[1] in proton–proton collisions at a center-of-mass energy of 14 TeV. We use `MadGraph5_aMC@NLO` [93] (version 3.5.3) at leading order in perturbative quantum chromodynamics for the hard-scattering processes with the `NNPDF23_lo_as_0130_qed` [94] set of parton distribution functions. We use the five-flavor scheme for the simulation of $t\bar{t}H$ production. The four-flavor scheme is chosen for tHproduction for an improved event modeling [95], where only the dominant t-channel contribution is considered. Only events with at least one semi-leptonic top quark decay are simulated.[2] For both processes, the factorization and renormalization scales are set event-by-event to the transverse mass of the irreducible $2 \to 2$ system resulting from a k_T clustering of the partons in the final state [96]. The events are interfaced to Pythia 8.3.1 [97] for the $H \to \gamma\gamma$ decay, parton shower and hadronization. We use Delphes 3.5.1 [98] for a fast simulation of the CMS detector response with the CMS card with default settings. These settings include jet clustering with the anti-k_T algorithm [99, 100] with a radius parameter of $R = 0.5$.

We focus on final states with two photons, at least one charged lepton (electron or muon), at least one b-jet and at least one additional jet. The following requirements are applied, where p_T is the transverse momentum and η is the pseudorapidity:

- exactly two photons (ordered in p_T) with $p_T(\gamma_1) > 35$ GeV and $p_T(\gamma_2) > 25$ GeV and $|\eta(\gamma_{1,2})| < 2.5$;
- invariant diphoton mass in the range $100\,\text{GeV} < m(\gamma\gamma) < 180\,\text{GeV}$ with $p_T(\gamma_1)/m(\gamma\gamma) > \frac{1}{3}$ and $p_T(\gamma_2)/m(\gamma\gamma) > \frac{1}{4}$;
- at least one charged lepton with $p_T(\ell) > 10$ GeV and $|\eta(\ell)| < 2.4$;
- at least one b-jet with $p_T(b) > 25$ GeV and $|\eta(b)| < 2.5$;
- at least one additional jet with $p_T(j) > 25$ GeV and $|\eta(j)| < 4.7$.

After applying these selection criteria, we have 100 000 events for training the classifiers and 33 000 events for validation during training, for each of the two processes. To ensure small statistical uncertainties in the metrics used in this study, we use 100 000 events per process of an independent test set to evaluate the networks.

We use 22 input features, which include four-vector components and high-level features based on photons, charged leptons, the missing transverse momentum (E_T^{miss}) and jets[3]. Among the selected jets, we focus on the highest-p_T b-jet and the leading jet of the event excluding this b-jet ("additional jet"). Ten of the features are shown in Fig. 1 for the two event classes. As it is typical for HEP event classification, no single feature provides sufficient discrimination on its own. Several features show the expected differences between $t\bar{t}H$ and tH production. For example, the number of jets (N(jets)) is larger in $t\bar{t}H$ production given the second top quark in the final state, and the pseudorapidity of the additional jet ($\eta(j)$) tends towards larger absolute values for tH due to the electroweak t-channel topology.

The matrix of the Pearson correlation coefficients is shown in Fig. 2, separately for $t\bar{t}H$ (lower triangle) and tH production (upper triangle). The 22 features show a non-trivial correlation structure with strong positive and negative correlations between some of the features. The correlations differ significantly in the $t\bar{t}H$ and tH data sets. For example, while the distributions of the transverse momentum of the diphoton system ($p_T(\gamma\gamma)$) in Fig. 1 are almost identical for the two data sets, the correlations of $p_T(\gamma\gamma)$ with other features are not.

[1] For tH production, the charge-conjugate process is also included, but we denote the sum of both processes as tH for simplicity.

[2] Final states with τ leptons are included in the event generation.

[3] While most of the high-level features are standard observables, the variable $y^j \oplus y^{\gamma\gamma}$ was proposed in Ref. [88] to disentangle $t\bar{t}H$ and tH production when testing different CP hypotheses for the top-quark Yukawa coupling.

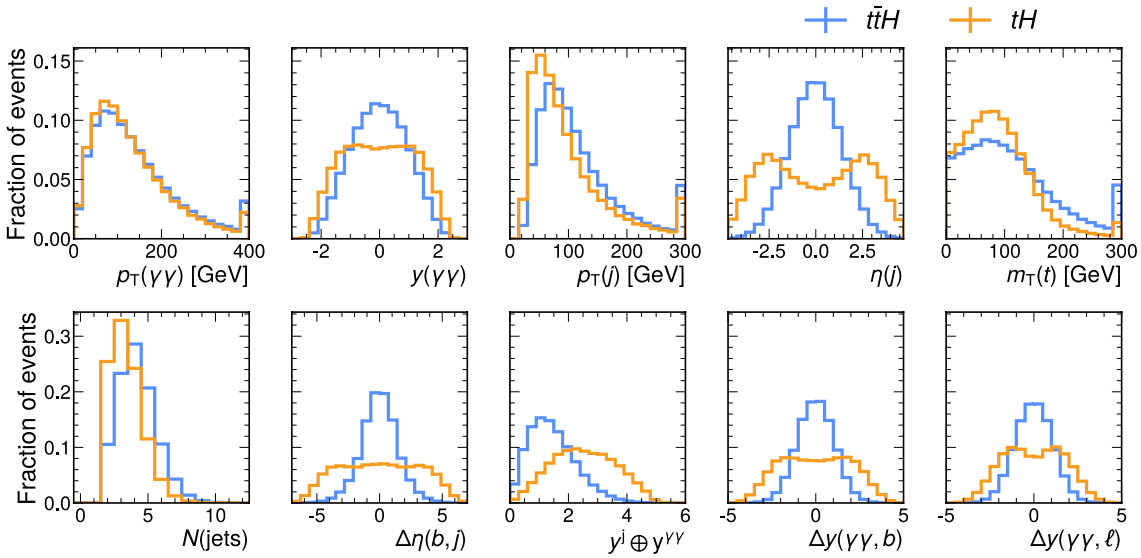

Fig. 1 Distributions of ten example features used for the classification. For distributions with overflow, the overflow is included in the last bin

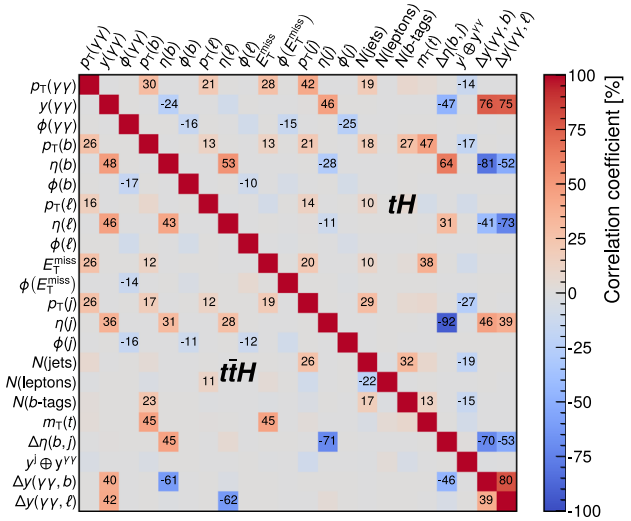

Fig. 2 Matrix of the Pearson correlation coefficients of all 22 input features. The upper triangle refers to the tH data set and the lower triangle refers to the $t\bar{t}H$ data set. Off-diagonal coefficients with absolute values of at least 10 % are shown as numbers on the plot

Results

We compare KANs and MLPs of different configurations for the classification of $t\bar{t}H$ and tH events. The KANs are implemented using the `Pykan` package from Ref. [17] in version 0.2.1 together with `PyTorch` [101] version 2.3.0. All MLPs are implemented with `TensorFlow` 2.17.0 [102].

We scale all input features before feeding them to the networks. A common approach for MLPs is studentization, where the sample mean is subtracted from each variable and the values are divided by the sample standard deviation. We apply this method for all MLP trainings. For KANs, we apply a different transformation to avoid the impact of outliers far away from the bulk of the distributions, which are common in typical HEP data sets. KANs require the domain of each learnable activation function to be defined by the range of the input for each spline. Outliers can extend the domain boundaries beyond the bulk of the distributions. As a result, the spline that acts on the majority of events may be parametrized by only a small fraction of the basis functions, which reduces the flexibility of curve approximation for the most relevant domain. To mitigate this, we first apply a logarithmic transformation to all transverse momenta and the transverse top quark mass, $\tilde{x} = \ln(1 + x)$, where x denotes the observable in units of GeV. We then apply min–max scaling in the range $[0, 1]$ and initialize the spline domains accordingly.

The output layers of all trained models consist of a single node. Although KANs learn activation functions and a learnable function can also be placed on the output node, we choose the sigmoid function to normalize the model outputs to the range $(0, 1)$. In addition, for the MLPs, we choose the sigmoid function as output activation. For all trainings, we use the binary cross-entropy as a loss function. We consider tH events as signal (label 1) and $t\bar{t}H$ as background (label 0).

We use the Adam [103] optimizer to train our models. For the KAN trainings, we also compare with the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [104] optimizer, as it was used in Ref. [17]. For Adam, we find stable trainings for all tested learning rates in the range $[10^{-4}, 10^{-3}]$ and select 3×10^{-4} for all trainings. The training is performed in mini-batches of batch size 256.

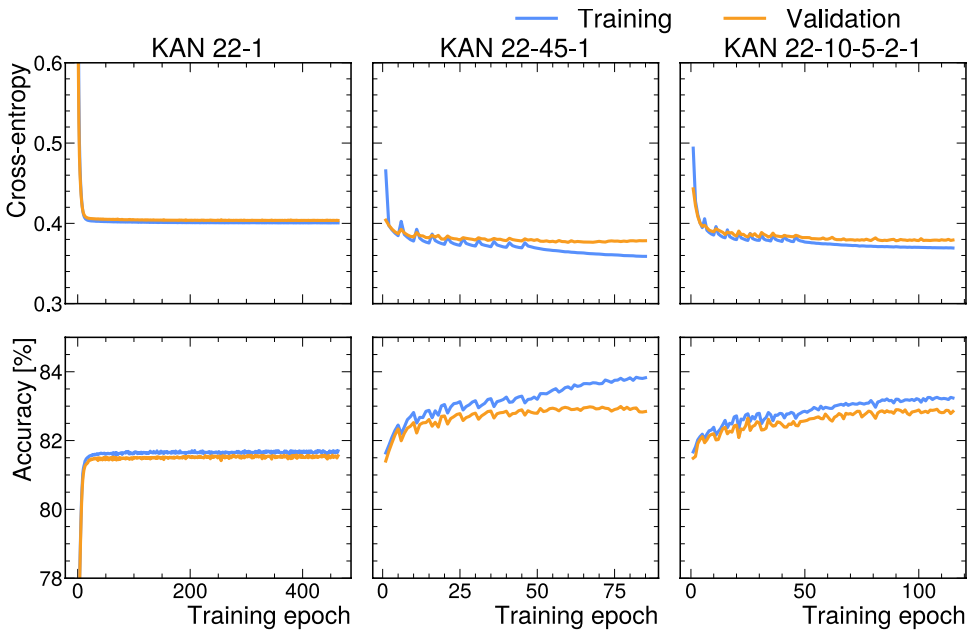

Fig. 3 Evolution of the loss (upper row) and the accuracy (lower row) of three KAN models of depth one, two and four, respectively. Due to the early-stopping approach, the epoch from which the model parameters are used appears 25 epochs before the end of the optimization. Instabilities occur in training epochs, where the spline domains of multi-layer KANs are adapted

Trainings with the L-BFGS optimizer are performed using the entire data set in full-batch training with a learning rate of 10^{-3}. No significant performance differences between KANs trained with the two optimizers were found and hence we use Adam for all trainings discussed below due to its faster convergence. To ensure that all models converge during training and to allow for a fair performance comparison, we employ early stopping. The loss obtained from the validation data set is monitored during training and if there is no improvement over 25 epochs, the training is terminated. The model parameters from the epoch with the lowest loss on the validation set are used for the comparisons.

We compare multiple KAN structures: The first model uses the configuration inspired by the Kolmogorov–Arnold theorem and hence consists of two layers with a node structure of 22–45–1. This is compared to the simplest possible KAN with only a single layer (22–1), as well as other models with varying widths and depths. For these models, we choose node structures of 22–3–1, 22–10–5–2–1 and 22–45–10–5–2–1. We use the default grid parameter, $G = 5$, and the default degree of the basis functions, $k = 3$.

The evolution of the loss and accuracy[4] over the training is shown in Fig. 3 for three KANs. As expected, the single-layer KAN shows the lowest performance among these models but still reaches an accuracy of 81.5% on the validation set, which is only about 1.5 percentage points lower than the accuracy of the two-layer and four-layer KANs. The latter two models reach similar accuracies and loss values. The two-layer KAN, with its wide second layer, has the highest number of trainable parameters in this comparison. It converges within the fewest number of training epochs and at the same time shows the largest generalization gap. While the range of the input variables is fixed with the min–max scaling, this is not the case for the input range of subsequent layers in the networks. Therefore, we use the default setting to update the domains ten times within the first 50 epochs of training. Training instabilities are visible at these epochs.

The output distributions of the test data set classified by these KANs are shown in Fig. 4. The separation of the two classes is clearly visible with the networks accumulating the majority of the events close to the respective label. Also here, only slight differences are visible between the two- and the four-layer network, while the one-layer KAN achieves a visibly worse separation.

A possible advantage of KANs over MLPs lies in their potential for interpretability. While the patterns learned by MLPs are usually embedded in large matrices of trainable parameters and thus hard to understand, multiple trained KAN parameters can be visualized in form of a single spline. As demonstrated in Ref. [17], patterns learned by KANs can often be understood more easily. However, these examples are much lower in dimensionality and complexity than typical HEP machine-learning tasks. In our study with 22 input features, we find that the interpretability of wide models, such as the 22–45–1 KAN, is limited. For instance, this particular model consists of more than 1000 learnable activation functions. Its visualization is, hence, complex and difficult for humans to interpret. Therefore, we focus on shallow KAN structures for the interpretability.

[4] The accuracy is defined as the fraction of correctly classified examples with a decision threshold of 0.5 in the network output.

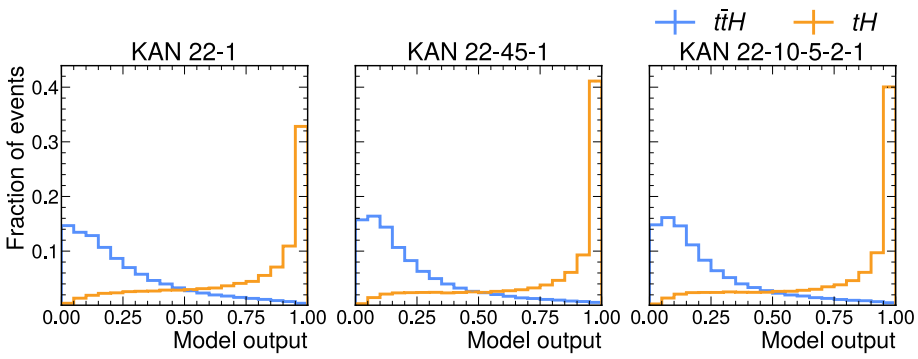

Fig. 4 Output distributions on the test data set for the two classes for three KANs with structures 22–1, 22–45–1 and 22–10–5–2–1, respectively

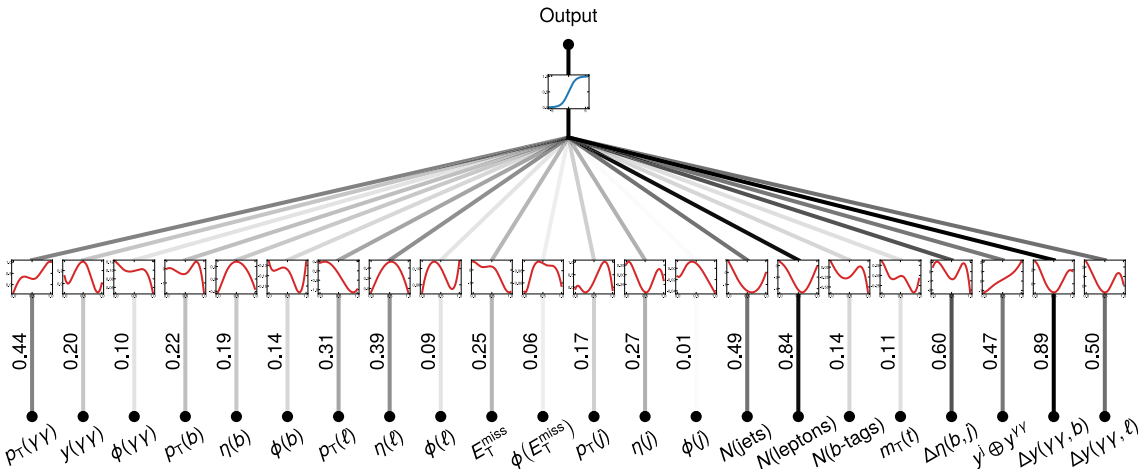

Fig. 5 Graphical representation of the trained KAN with a single layer (KAN 22–1). The red curves represent the learned activation functions, while the blue curve shows the sigmoid function used to normalize the network output. The L_1-norm of each spline is given, which also defines the grayscale of each edge

In Fig. 5, the one-layer KAN is depicted together with its learned activation functions. The corresponding figure for the KAN with a shallow second layer (22–3–1) is shown in the Appendix. In the following, we discuss the interpretability of the learned activation functions.

The strength of the edges is indicated by the grayscale and it is estimated by the L_1-norms of the learned activations, defined as the mean magnitude over the examples as

$$|\phi(x)|_1 = \frac{1}{N(\text{events})} \sum_{i=1}^{N(\text{events})} |\phi(x_i)|. \tag{5}$$

These values directly indicate the importance of the different input features for the one-layer KAN classification output due to the simple summation of input features transformed by a single activation function on its output node. The feature importances obtained from the commonly used techniques of permutation feature importance and Shapley values yield similar patterns. This confirms that the L_1 -norms in the one-layer KAN effectively capture feature importance. This network is expected to exploit the features with strong discrimination between the two classes. Values of the L_1-norms close to zero are found for the azimuthal angles, which provide no discrimination power on their own. The largest values of the L_1-norms are found for the lepton multiplicity, with which the network can identify dileptonic signatures present in a fraction of the $t\bar{t}H$ events but mostly absent in the tH process, and for variables based on (pseudo-)rapidity differences as well as the jet multiplicity. These reflect the good separation that can be achieved with these variables on their own. In deeper KANs, where multiple learnable functions transform each feature and their outputs are combined in a more complex manner, the L_1-norms are not directly interpretable as feature importance scores.

In Fig. 6, the learned activation functions are shown for five examples of features. These features were chosen to represent a set of variables with different shapes of their distributions, where the corresponding splines of the single-layer KAN have high L_1-norms. The distributions of the pre-processed features are also shown for the two classes, together with the univariate log-likelihood ratios of the signal over the background. As expected in binary classification, where

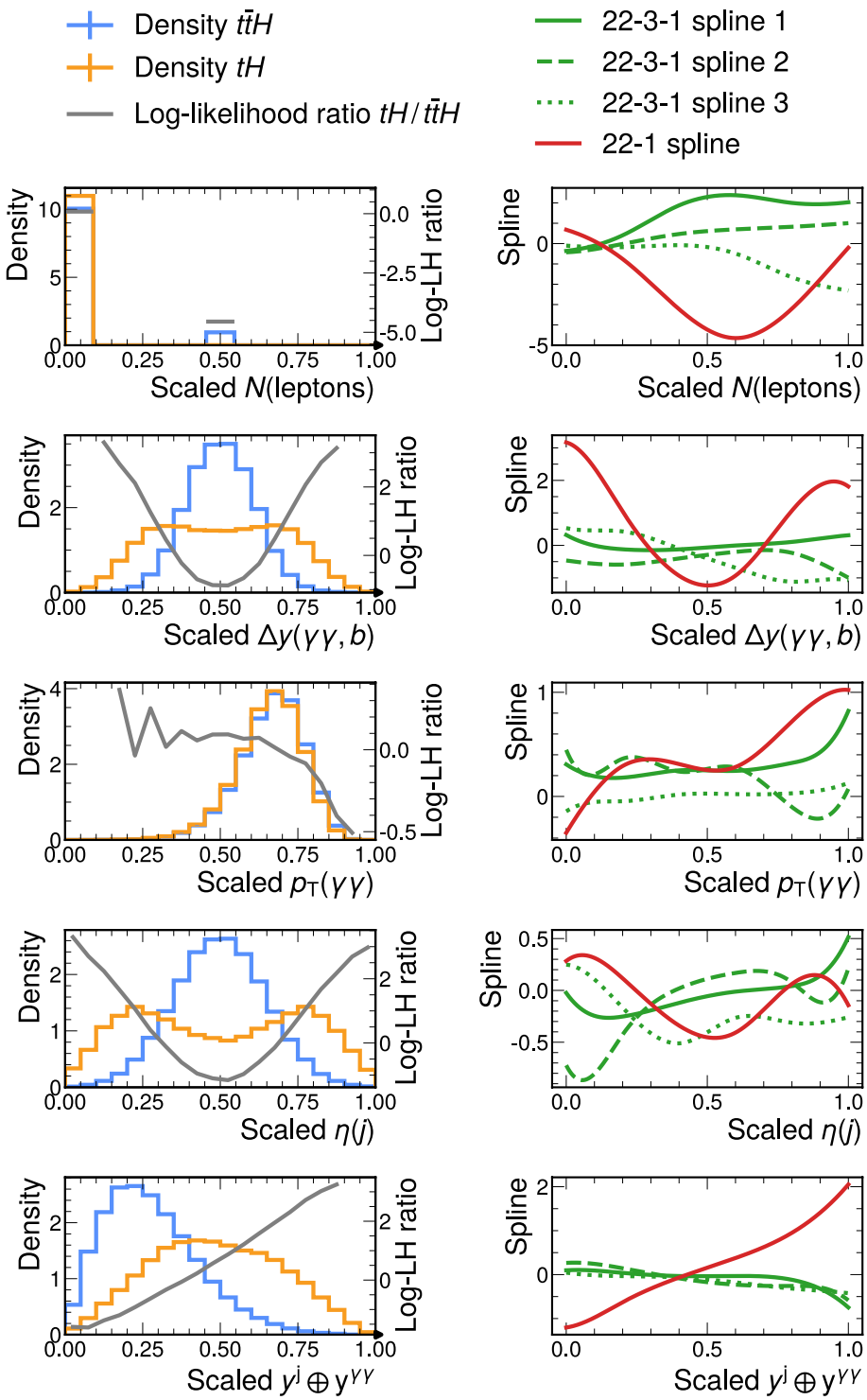

Fig. 6 Left: distributions of five examples of pre-processed input features for tH and $t\bar{t}H$ production together with the corresponding log-likelihood ratio. A ratio is shown if there are at least 25 examples of each process from the training data set in the respective bin. Right: the learned spline for these input features in KAN 22–1 (red) and the three learned splines in the first layer of KAN 22–3–1 (green)

the optimal decision boundary is related to the log-likelihood ratio of the classes, the activation functions learned by the one-layer KAN resemble these univariate log-likelihood ratios of the respective input features. This resemblance is anticipated, because the one-layer KAN sums univariate functions of each input feature, aligning with the additive form of the multivariate log-likelihood ratio, which decomposes into a sum of univariate log-likelihood ratios when features are independent. However, due to correlations, differences between the learned activation functions and the univariate log-likelihood ratios may arise, as we find, for example, in the normalization of the spline transforming the $\eta(j)$ variable. The corresponding splines obtained from the training of the 22–3–1 KAN are also shown in the same

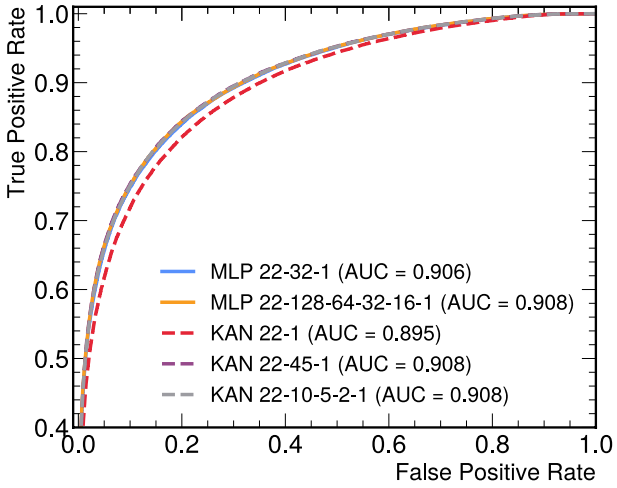

Fig. 7 ROC curves of selected models labeled with their node counts and the AUC scores. Results from two example MLPs and three example KANs are shown, including deep and shallow models

panels. These differ clearly from the likelihood ratios and the 22–1 KAN splines. This is consistent with the expectation that multi-layer KANs learn more abstract representations of the data. As the splines acting on the same input feature are allowed to have different functional shapes or L_1-norms, each of the three nodes of the internal layer of the 22–3–1 KAN captures different aspects of the data.

For comparison to the KANs, we train several MLPs with different configurations: two shallow MLPs, each with only a single hidden layer of 8 and 32 nodes, respectively, as well as deeper networks with up to five hidden layers. The largest model has a node count of 22–256–128–64–32–16–1. The hidden layers are activated by ReLU functions. The comparison includes models with a number of trainable parameters as small as approximately 200 to approximately 60 000.

Receiver operating characteristic (ROC) curves for selected models are shown in Fig. 7, including models with deep and shallow structures for both, MLPs and KANs. The overall shape of the ROC curves of KANs and MLPs is very similar, except for the one-layer KAN, where the area under the curve (AUC) is considerably lower. We observe that an MLP with only a single hidden layer reaches an AUC[5] of 0.906, only slightly below the AUCs obtained by our best MLP (0.908) and the best KAN, where the 22–45–1 network also reaches an AUC of 0.908. We find only slight differences in classification performance between well-tuned KANs and MLPs.

To evaluate the parameter efficiency, we compare the performance of KANs and MLPs as a function of the number of trainable model parameters. A small number of parameters is favorable for a given performance, as smaller models are computationally more efficient, better interpretable (if at all possible), and less prone to overfitting. In Fig. 8, two metrics often used in HEP for classification performance evaluation are included: the AUC, and the background rejection for a fixed signal efficiency, here chosen as 70 %. The rejection is defined as the inverse of the efficiency for a given threshold on the network output. Overall, we find that for very low numbers of parameters, the MLPs outperform the KANs, while for medium and high numbers of parameters, the performance of KANs and MLPs is similar. The smallest MLP trained with only eight nodes in the hidden layer has only 193 parameters and has reached a background rejection of 12.9 already.[6] A similar value is achieved by the 22–3–1 KAN, which has 556 trainable parameters. While increasing the number of parameters of the MLPs, their rejection saturates at around 14.9. The KANs achieve rejections ranging from 11.4 for the single-layer network to 15.2 for the 22–45–1 KAN. The deeper KANs from this study show a similar performance. Instead of varying the node count of KANs, the number of model parameters can be increased as well by raising the grid parameter G. For illustration, the 22–45–1 KAN is included in Fig. 8 when trained with $G = 10$ and $G = 25$. For KANs with grid parameters considerably higher than the default of $G = 5$, we find that these models overtrain faster and generalize worse.

Conclusions

We studied the application of Kolmogorov–Arnold Networks (KANs) to a typical binary event classification task in high-energy physics (HEP). The data set used contains simulated events of Higgs-boson production in association with a top quark pair and with a single top quark in the $H \to \gamma\gamma$ decay channel, where 22 discriminative input features were considered in the network trainings. We presented studies on the interpretability of KANs, compared their performance and parameter efficiency to traditional multilayer perceptrons (MLPs), and documented our findings in the practical training of KANs. To our knowledge, this is the first time KANs have been applied in the field of particle physics.

As long as the KANs have a hidden layer with multiple nodes, we observe very similar performance to MLPs. Numerically, our best KAN is even slightly better than the best MLP, but this difference is very small and most

[5] The uncertainty in the quoted AUCs from the limited size of the test data set are approximately 6×10^{-4}.

[6] The uncertainty in the quoted rejection values from the limited size of the test data set is approximately 0.2.

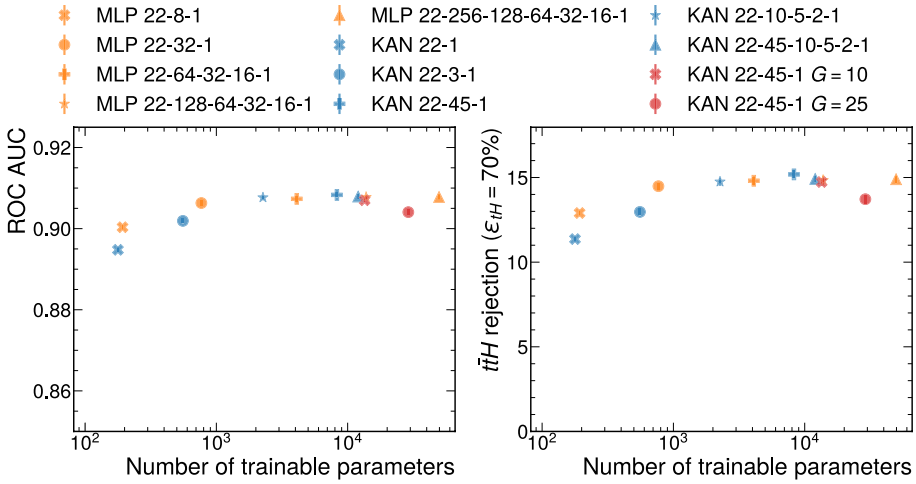

Fig. 8 Parameter efficiency of KANs and MLPs: The ROC AUC for different models (left) and the $t\bar{t}H$ rejection for a tH efficiency of 70 % (right) are shown as a function of the number of parameters of the different models. Error bars corresponding to the limited size of the test data set are shown, but they are smaller than the symbols

likely practically irrelevant when considering systematic uncertainties in a physics analysis. These findings seem to contradict Ref. [17], where a clear improvement over the performance of MLPs was found in several examples. In these examples, loss values were reached that were orders of magnitude lower than those of MLPs, for example, in fitting $f(x, y) = x \cdot y$. However, those examples are of much lower dimensionality and complexity than our classification task. We suppose that the examples in Ref. [17] are especially well-suited for learning the Kolmogorov–Arnold representation of the underlying functional relationship. However, our data set includes features with a stronger variability in shape. The representations that have to be learned to solve our classification task may hence not be particularly suited for the KAN architecture.

We find that MLPs outperform KANs for a very low number of trainable parameters. However, we note that for our binary classification task, which we consider typical in terms of complexity for event classification at the LHC, using such very small models only comes at a moderate cost regarding performance. Similar performance of MLPs and KANs is then reached for a number of trainable parameters above approximately 1000. We conclude that for our task KANs are not more parameter efficient than MLPs.

In terms of interpretability, we find that small KANs indeed offer advantages. For a one-layer KAN, we observe that the learned activation functions resemble the univariate log-likelihood ratios of the input features which these functions act on. In addition, the L_1-norms of the activation functions offer a straightforward interpretation of the importance of different input features for such KANs with only one layer. Somewhat larger KANs may still offer an illustrative visualization of the activation functions, which we regard as an intrinsic interpretability advantage of KANs. However, for KANs of greater depth or with wider layers, interpretability seems challenging.

Because of their better interpretability compared to MLPs, we conclude that KANs are a promising alternative for classification tasks in HEP when the performance of small KANs is sufficient or when moderate performance losses are acceptable in favor of interpretability. We believe that more research on applying KANs in HEP tasks is necessary. This study primarily focuses on the performance and parameter efficiency of KANs compared to MLPs, as well as the interpretability of small KANs. The interpretability of deeper KANs should be explored in future studies to address this limitation of our current work. Mechanistic interpretability techniques may provide systematic ways to understand the learned representations in these more complex models, and exploring symbolic regression to approximate the learned activation functions could open new avenues for applying KANs as a tool in HEP. In addition, assessing KAN performance and interpretability in HEP regression tasks may further broaden their applicability beyond classification.

Appendix

The graphical representation of the trained KAN 22–3–1 is shown in Fig. 9. We observe a clear hierarchy in the importance of the different input features for the KAN output, as indicated by the L_1-norms on the input nodes. For the edges with large L_1-norms, we observe the tendency towards simple and smooth activation functions. For edges with lower values of the L_1-norms, we also observe more complex activation functions with several local minima and maxima.

Acknowledgements This research was supported by the Deutsche Forschungsgemeinschaft (DFG) under grants 400140256 - GRK 2497 (The physics of the heaviest particles at the LHC, all authors) and 686709 - ER 866/1-1 (Heisenberg Programme, JE), and by the Studienstiftung des deutschen Volkes (FM, JLS).

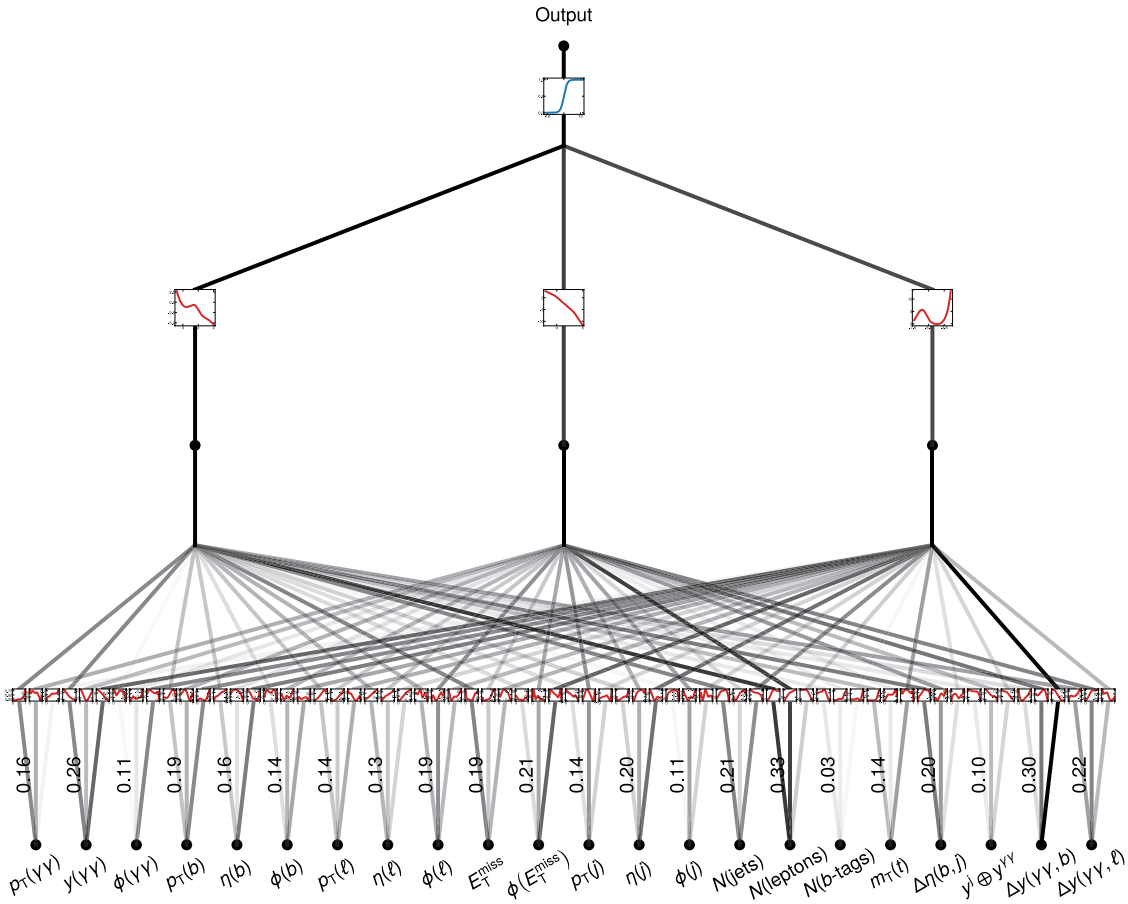

Fig. 9 Graphical representation of the trained KAN in the 22–3–1 configuration, i.e., with a second layer with three nodes. The red curves represent the learned activation functions, while the blue curve shows the sigmoid function used to normalize the network output. The values printed on the edges of the first KAN layer are the L_1-norm of each input node, averaged over the three activation functions

Author contributions F.M. conducted the machine-learning studies and prepared all figures. J.E. supervised F.M. and J.L.S. J.L.S. and F.M. simulated the datasets. J.E. and F.M. wrote the main manuscript text. All authors reviewed the manuscript.

Data availability Open Access funding enabled and organized by Projekt DEAL. The data sets generated and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests The authors declare no competing interests.

References

1. D0 collaboration, Abazov VM, et al (2009) Observation of Single Top Quark Production. Phys Rev Lett 103:092001. https://doi.org/10.1103/PhysRevLett.103.092001
2. CDF collaboration, Aaltonen T, et al (2009) First Observation of Electroweak Single Top Quark Production. Phys Rev Lett 103:092002. https://doi.org/10.1103/PhysRevLett.103.092002Phys.10.1103/PhysRevLett.103.092002Phys
3. Baldi P, Sadowski P, Whiteson D (2014) Searching for Exotic Particles in High-Energy Physics with Deep Learning. Commun 5:4308. https://doi.org/10.1038/ncomms5308Nature. [arXiv:1402.4735].
4. Feickert M, Nachman B, A Living Review of Machine Learning for Particle Physics, arXiv:2102.02770
5. Guest D, Cranmer K, Whiteson D (2018) Deep Learning and its Application to LHC Physics. Rev Nucl Part Sci 68:161. https://

doi.org/10.1146/annurev-nucl-101917-021019Ann. [arXiv:1806.11484].
6. Radovic A, Williams M, Rousseau D, Kagan M, Bonacorsi D, Himmel A et al (2018) Mach Learn Energy Intensiv Front Part Phys 560:41. https://doi.org/10.1038/s41586-018-0361-2Nature
7. Schwartz MD (2021) Modern Machine Learning and Particle Physics. Harv Data Sci Rev. https://doi.org/10.1162/99608f92.beeb1183
8. Karagiorgi G, Kasieczka G, Kravitz S, Nachman B, Shih D (2022) Machine learning in the search for new fundamental physics. Rev Phys 4:399. https://doi.org/10.1038/s42254-022-00455-1Nature
9. Doshi-Velez F, Kim B, Towards A Rigorous Science of Interpretable Machine Learning, arXiv:1702.08608
10. Arrieta AB, Díaz-Rodríguez N, Ser JD, Bennetot A, Tabik S, Barbado A et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI, arXiv:1910.10045
11. Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F, Pedreschi D (2018) A survey of methods for explaining black box models. ACM Comput Surv 51:1
12. Li X, Xiong H, Li X, Wu X, Zhang X, Liu J et al (2022) Interpretable deep learning: interpretation, interpretability, trustworthiness, and beyond. Inf Syst 64:3197. https://doi.org/10.1007/s10115-022-01756-8Knowledgeand. [arXiv:2103.10689]
13. Lundberg S, Lee S-I A unified approach to interpreting model predictions, arXiv:1705.07874
14. Breiman L (2001) Random forests. Learning 45:5. https://doi.org/10.1023/A:1010933404324Machine
15. Ribeiro MT, Singh S, Guestrin C (2016) "Why Should I Trust You?": Explaining the Predictions of Any Classifier, Proceedings of KDD2016 1135
16. Agarwal R, Melnick L, Frosst N, Zhang X, Lengerich B, Caruana R et al (2021) Neural Additive Models: Interpretable machine learning with neural nets. Adv Neural Inf Process Syst 34:4699
17. Liu Z, Wang Y, Vaidya S, Ruehle F, Halverson J, Soljačić M et al. KAN: Kolmogorov-Arnold Networks, arXiv:2404.19756
18. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Networks 2:359. https://doi.org/10.1016/0893-6080(89)90020-8Neural
19. Kolmogorov AN (1957) On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. Doklady Akademii Nauk 114:953
20. Bohra P, Campos J, Gupta H, Aziznejad S, Unser M (2020) Learning Activation Functions in Deep (Spline) Neural Networks. IEEE Open J Sign Process 1:295
21. Aziznejad S, Unser M (2019) Deep Spline Networks with Control of Lipschitz Regularity, Proceedings of ICASSP, pp 3242
22. Agostinelli F, Learning activation functions to improve deep neural networks, arXiv:1412.6830
23. Goyal M, Goyal R, Lall B Learning activation functions: A new paradigm for understanding neural networks, arXiv:1906.09529
24. Ziyin L, Hartwig T, Ueda M (2020) Neural networks fail to learn periodic functions and how to fix it. Adv Neural Inf Process Syst 33:1583
25. Hashemi B, Corominas RG, Giacchetto A Can transformers do enumerative geometry?, arXiv:2408.14915
26. Zhang S, Shen Z, Yang H (2022) Neural network architecture beyond width and depth. Adv Neural Inf Process Syst 35:5669
27. Sprecher DA, Draghici S (2002) Space-filling curves and Kolmogorov superposition-based neural networks. Neural Netw 15:57
28. Köppen M(2002) On the Training of a Kolmogorov Network, Proceedings of ICANN, pp 474
29. Lin J-N, Unbehauen R (1993) On the Realization of a Kolmogorov Network. Neural Comput 5:18
30. Lai M-J, Shen Z The Kolmogorov Superposition Theorem can Break the Curse of Dimensionality When Approximating High Dimensional Functions, arXiv:2112.09963
31. Leni P-E, Fougerolle YD, Truchetet F (2013) The Kolmogorov Spline Network for Image Processing, Proceedings of Image Processing: Concepts, Methodologies, Tools, and Applications, pp 54
32. Fakhoury D, Fakhoury E, Speleers H (2022) ExSpliNet: An interpretable and expressive spline-based neural network. Neural Netw 152:332
33. Montanelli H, Yang H (2020) Error bounds for deep ReLU networks using the Kolmogorov-Arnold superposition theorem. Neural Netw 129:1
34. He J On the Optimal Expressive Power of ReLU DNNs and Its Application in Approximation with Kolmogorov Superposition Theorem, arXiv:2308.05509
35. Duda J Biology-inspired joint distribution neurons based on Hierarchical Correlation Reconstruction allowing for multidirectional neural networks, arXiv:2405.05097
36. Li Z Kolmogorov-Arnold Networks are radial basis function networks, arXiv:2405.06721
37. Genet R, Inzirillo H TKAN: Temporal Kolmogorov-Arnold Networks, arXiv:2405.07344
38. Peng Y, He M, Hu F, Mao Z, Huang X, Ding J Predictive Modeling of Flexible EHD Pumps using Kolmogorov-Arnold Networks, arXiv:2405.07488
39. Vaca-Rubio CJ, Blanco L, Pereira R, Caus M Kolmogorov-Arnold Networks (KANs) for time series analysis, arXiv:2405.08790
40. Samadi ME, Müller Y, Schuppert A Smooth Kolmogorov Arnold Networks enabling structural knowledge representation, arXiv:2405.11318
41. Bozorgasl Z, Chen H Wav-KAN: Wavelet Kolmogorov-Arnold Networks, arXiv:2405.12832
42. Yang S, Qin L, Yu X Endowing Interpretability for Neural Cognitive Diagnosis by Efficient Kolmogorov-Arnold Networks, arXiv:2405.14399
43. Abueidda DW, Pantidis P, Mobasher ME DeepOKAN: Deep Operator Network Based on Kolmogorov Arnold Networks for mechanics problems, arXiv:2405.19143
44. Cheon M Kolmogorov-Arnold Network for Satellite Image Classification in Remote Sensing, arXiv:2406.00600
45. Xu J, Chen Z, Li J, Yang S, Wang W, Hu X et al. FourierKAN-GCF: Fourier Kolmogorov-Arnold Network–An Effective and Efficient Feature Transformation for Graph Collaborative Filtering, arXiv:2406.01034
46. Xu K, Chen L, Wang S Kolmogorov-Arnold Networks for Time Series: Bridging Predictive Power and Interpretability, arXiv:2406.02496
47. Genet R, Inzirillo H A Temporal Kolmogorov-Arnold Transformer for Time Series Forecasting, arXiv:2406.02486
48. Nehma G, Tiwari M Leveraging KANs For Enhanced Deep Koopman Operator Discovery, arXiv:2406.02875
49. Li C, Liu X, Li W, Wang C, Liu H, Yuan Y U-KAN Makes Strong Backbone for Medical Image Segmentation and Generation, arXiv:2406.02918
50. Shukla K, Toscano JD, Wang Z, Zou Z, Karniadakis GE A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks, arXiv:2406.02917
51. Herbozo Contreras LF, Cui J, Yu L, Huang Z, Nikpour A, Kavehei O KAN-EEG: Towards Replacing

Backbone-MLP for an Effective Seizure Detection System, med Rxiv:10.1101/2024.06.05.24308471
52. Kiamari M, Kiamari M, Krishnamachari B GKAN: Graph Kolmogorov-Arnold Networks, arXiv:2406.06470
53. Aghaei AA fKAN: Fractional Kolmogorov-Arnold Networks with trainable Jacobi basis functions, arXiv:2406.07456
54. Seydi ST Unveiling the Power of Wavelets: A Wavelet-based Kolmogorov-Arnold Network for Hyperspectral Image Classification, arXiv:2406.07869
55. Azam B, Akhtar N Suitability of KANs for Computer Vision: A preliminary investigation, arXiv:2406.09087
56. Chen Y, Zhu Z, Zhu S, Qiu L, Zou B, Jia F et al. SCKansformer: Fine-Grained Classification of Bone Marrow Cells via Kansformer Backbone and Hierarchical Attention Mechanisms, arXiv:2406.09931
57. Wang Y, Sun J, Bai J, Anitescu C, Eshaghi MS, Zhuang X et al. Kolmogorov Arnold Informed neural network: A physics-informed deep learning framework for solving PDEs based on Kolmogorov Arnold Networks, arXiv:2406.11045
58. Ta H-T BSRBF-KAN: A combination of B-splines and Radial Basic Functions in Kolmogorov-Arnold Networks, arXiv:2406.11173
59. Zhang F, Zhang X GraphKAN: Enhancing Feature Extraction with Graph Kolmogorov Arnold Networks, arXiv:2406.13597
60. Bodner AD, Tepsich AS, Spolski JN, Pourteau S Convolutional Kolmogorov-Arnold Networks, arXiv:2406.13155
61. Poeta E, Giobergia F, Pastor E, Cerquitelli T, Baralis E A Benchmarking Study of Kolmogorov-Arnold Networks on Tabular Data, arXiv:2406.14529
62. Aghaei AA rKAN: Rational Kolmogorov-Arnold Networks, arXiv:2406.14495
63. Cheon M Demonstrating the Efficacy of Kolmogorov-Arnold Networks in Vision Tasks, arXiv:2406.14916
64. De Carlo G, Mastropietro A, Anagnostopoulos A Kolmogorov-Arnold Graph Neural Networks, arXiv:2406.18354
65. Bresson R, Nikolentzos G, Panagopoulos G, Chatzianastasis M, Pang J, Vazirgiannis M KAGNNs: Kolmogorov-Arnold Networks meet Graph Learning, arXiv:2406.18380
66. Howard AA, Jacob B, Murphy SH, Heinlein A, Stinis P Finite basis Kolmogorov-Arnold Networks: domain decomposition for data-driven and physics-informed problems, arXiv:2406.19662
67. Wang Y, Yu X, Gao Y, Sha J, Wang J, Gao L et al. SpectralKAN: Kolmogorov-Arnold Network for Hyperspectral Images Change Detection, arXiv:2407.00949
68. Lobanov V, Firsov N, Myasnikov E, Khabibullin R, Nikonorov A, HyperKAN: Kolmogorov-Arnold Networks make Hyperspectral Image Classificators Smarter, arXiv:2407.05278
69. Dong F TCKIN: A Novel Integrated Network Model for Predicting Mortality Risk in Sepsis Patients, arXiv:2407.06560
70. Lawan A, Pu J, Yunusa H, Umar A, Lawan M MambaForGCN: Enhancing Long-Range Dependency with State Space Model and Kolmogorov-Arnold Networks for Aspect-Based Sentiment Analysis, arXiv:2407.10347
71. Altarabichi MG DropKAN: Regularizing KANs by masking post-activations, arXiv:2407.13044
72. Shen H, Zeng C, Wang J, Wang Q Reduced Effectiveness of Kolmogorov-Arnold Networks on Functions with Noise, arXiv:2407.14882
73. Inzirillo H Deep State Space Recurrent Neural Networks for Time Series Forecasting, arXiv:2407.15236
74. Troy W Sparks of Quantum Advantage and Rapid Retraining in Machine Learning, arXiv:2407.16020
75. Toscano JD, Käufer T, Maxey M, Cierpka C, Karniadakis GE Inferring turbulent velocity and temperature fields and their statistics from Lagrangian velocity measurements using physics-informed Kolmogorov-Arnold Networks, arXiv:2407.15727
76. Li X, Feng Z, Chen Y, Dai W, He Z, Zhou Y et al. COEFF-KANs: A Paradigm to Address the Electrolyte Field with KANs, arXiv:2407.20265
77. Rigas S, Papachristou M, Papadopoulos T, Anagnostopoulos F, Alexandridis G Adaptive Training of Grid-Dependent Physics-Informed Kolmogorov-Arnold Networks, arXiv:2407.17611
78. Le TXH, Tran TD, Pham HL, Le VTD, Vu TH, Nguyen VT et al. Exploring the Limitations of Kolmogorov-Arnold Networks in Classification: Insights to Software Training and Hardware Implementation, arXiv:2407.17790
79. Seguel F, Salihu D, Hägele S, Steinbach E (2024) VLP-KAN: Low-complexity and Interpretable RSS-based Visible Light Positioning using Kolmogorov-Arnold Networks
80. Zeydan E, Vaca-Rubio CJ, Blanco L, Pereira R, Caus M, Aydeger A F-KANs: Federated Kolmogorov-Arnold Networks, arXiv:2407.20100
81. Zinage S, Mondal S, Sarkar S DKL-KAN: Scalable Deep Kernel Learning using Kolmogorov-Arnold Networks, arXiv:2407.21176
82. Pratyush P, Carrier C, Pokharel S, Ismail HD, Chaudhari M, KC DB CaLMPhosKAN: Prediction of General Phosphorylation Sites in Proteins via Fusion of Codon Aware Embeddings with Amino Acid Aware Embeddings and Wavelet-based Kolmogorov Arnold Network, bioRxiv:10.1101/2024.07.30.605530
83. Altarabichi MG Rethinking the Function of Neurons in KANs, arXiv:2407.20667
84. Liu H, Lei J, Ren Z From Complexity to Clarity: Kolmogorov-Arnold Networks in Nuclear Binding Energy Prediction, arXiv:2407.20737
85. Tang T, Chen Y, Shu H 3D U-KAN Implementation for Multimodal MRI Brain Tumor Segmentation, arXiv:2408.00273
86. Li R GNN-MolKAN: Harnessing the Power of KAN to Advance Molecular Representation Learning with GNNs, arXiv:2408.01018
87. Farina M, Grojean C, Maltoni F, Salvioni E, Thamm A (2013) Lifting degeneracies in Higgs couplings using single top production in association with a Higgs boson. JHEP 05:022. https://doi.org/10.1007/JHEP05(2013)022. [arXiv:1211.3736]
88. Bahl H, Bechtle P, Heinemeyer S, Katzy J, Klingl T, Peters K et al (2020) Indirect *CP* probes of the Higgs-top-quark interaction: current LHC constraints and future opportunities. JHEP 11:127. https://doi.org/10.1007/JHEP11(2020)127. [arXiv:2007.08542]
89. CMS collaboration, Sirunyan AM, et al (2018) Observation of $t\bar{t}H$ production. Phys Rev Lett 120:231801. https://doi.org/10.1103/PhysRevLett.120.231801
90. ATLAS collaboration, Aaboud M. et al (2018) Observation of Higgs boson production in association with a top quark pair at the LHC with the ATLAS detector. Phys Lett B 784:173. https://doi.org/10.1016/j.physletb.2018.07.035
91. CMS collaboration, Hayrapetyan A. et al. Measurement of the $t\bar{t}H$ and tH production rates in the $H \to b\bar{b}$ decay channel using proton-proton collision data at $\sqrt{s} = 13$ TeV, arXiv:2407.10896
92. ATLAS collaboration, Aad G. et al (2022) A detailed map of Higgs boson interactions by the ATLAS experiment ten years after the discovery. Nature 607:52. https://doi.org/10.1038/s41586-022-04893-w
93. Alwall J, Frederix R, Frixione S, Hirschi V, Maltoni F, Mattelaer O et al (2014) The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. JHEP 07:079. https://doi.org/10.1007/JHEP07(2014)079. [arXiv:1405.0301]
94. Ball RD et al (2013) Parton distributions with LHC data. Phys B 867:244. https://doi.org/10.1016/j.nuclphysb.2012.10.003Nucl. [arXiv:1207.1303].

95. Demartin F, Maltoni F, Mawatari K, Zaro M (2015) Higgs production in association with a single top quark at the LHC. Phys J C 75:267. https://doi.org/10.1140/epjc/s10052-015-3475-9Eur. [arXiv:1504.00611].
96. Hirschi V, Mattelaer O (2015) Automated event generation for loop-induced processes. JHEP 10:146. https://doi.org/10.1007/JHEP10(2015)146. [arXiv:1507.00020]
97. Bierlich C et al (2022) A comprehensive guide to the physics and usage of PYTHIA 83. SciPost Phys Codeb. https://doi.org/10.21468/SciPostPhysCodeb.8
98. DELPHES 3 collaboration, de Favereau J, Delaere C, Demin P, Giammanco A, Lemaître V, Mertens A et al. DELPHES 3: a modular framework for fast simulation of a generic collider experiment. JHEP 02:057 (2014) https://doi.org/10.1007/JHEP02(2014)057, [arXiv:1307.6346]
99. Cacciari M, Salam GP, Soyez G (2008) The anti-k_t jet clustering algorithm. JHEP 04:063. https://doi.org/10.1088/1126-6708/2008/04/063. [arXiv:0802.1189]
100. Cacciari M, Salam GP, Soyez G (2012) FastJet User Manual. Phys J C 72:1896. https://doi.org/10.1140/epjc/s10052-012-1896-2Eur. [arXiv:1111.6097].
101. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G et al. (2019) Pytorch: An imperative style, high-performance deep learning library, Proceedings of NeurIPS. arXiv:1912.01703
102. TensorFlow Developers, TensorFlow v2.17.0, (2024). https://doi.org/10.5281/zenodo.12726004
103. Kingma DP, Ba J (2015) Adam: A Method for Stochastic Optimization, Proceedings of ICLR. arXiv:1412.6980
104. Liu DC, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Programming 45:503. https://doi.org/10.1007/BF01589116Mathematical