ACCELERATING GLOBAL OPTIMISATION WITH PIECEWISE-LINEAR KOLMOGOROV–ARNOLD NETWORK SURROGATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural surrogates can provide highly accurate approximations of complex functions, yet this alone does not guarantee reliable and efficient downstream global optimisation. When neural networks are embedded into mathematical programs, their architecture and representation directly affect formulation size, relaxation strength, and ultimately the optimisation runtime. This paper focuses on continuous piecewise-linear Kolmogorov–Arnold networks (CPWL-KAN), whose learnable edge functions are parametrised as linear combinations of ReLU functions and admit exact mixed-integer programming (MIP) representations. Three CPWL-KAN variants are considered, which correspond to different weight combinations and expressivity properties. Numerical experiments conducted across three benchmark functions and an engineering optimisation problem show that CPWL-KANs can provide predictive accuracy comparable to, and sometimes better than, that of conventional ReLU MLPs of similar complexity; but the computational runtime with MIP solvers to certify global optimality of the surrogates can be significantly lower with CPWL-KANs embedded. Overall, these results demonstrate that functional parametrisation of neural surrogates is as important as predictive accuracy when their intended use is in downstream optimisation.

1 INTRODUCTION

Although machine learning models have achieved impressive predictive performance in numerous tasks, their application as surrogate models within systematic optimisation based on mathematical programming remains much less explored, particularly when seeking a certificate of global optimality (Bhosekar & Ierapetritou, 2018; Grimstad & Andersson, 2019; Schweidtmann & Mitsos, 2019; Tjandraatmadja et al., 2020; Anderson et al., 2020). The central challenge is that a high predictive accuracy does not necessarily translate into, and often conflicts with, optimisation tractability in practice. For instance, embedding neural surrogates into mathematical programs often increases problem size and results in highly nonconvex formulations, ultimately increasing the computational efforts for global optimisation (Misener & Biegler, 2023; Wang et al., 2023; Witte et al., 2025). Improved optimisation tractability can be achieved by imposing structural restrictions on the surrogate. For example, input convex neural networks enforce convexity with respect to inputs at the cost of representation flexibility (Amos et al., 2017; Liu et al., 2026). This generates a fundamental trade-off between predictive performance and optimisation tractability, motivating the development of surrogate architectures that are tractable for certifying global optimality while retaining a suitable approximation accuracy.

Benefiting from their exact representability as mixed-integer linear programs (MIP), ReLU multilayer perceptrons (MLPs), also known as ReLU artificial neural networks (ANNs), have been widely investigated in optimisation and verification tasks (Fischetti & Jo, 2018). A common strategy entails encoding each ReLU term in all the layers separately using the big-M method, where a binary variable is introduced to indicate its activation state (Grimstad & Andersson, 2019). However, ignoring the dependencies among activation states can result in loose pre-activation bounds and weak continuous relaxations, which substantially increase the MIP solution effort (Vecchietti et al., 2003; Vielma, 2015; 2018; Bunel et al., 2020; Kronqvist et al., 2021). Developing small ANNs is an intuitive way of reducing the computational costs but usually sacrifices some predictive accuracy (Las-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

trucci et al., 2026). Stronger formulations that leverage the continuous piecewise-linear (CPWL) structure of ReLU ANNs, along with effective bound tightening strategies have also been proposed to improve the optimisation performance over ReLU networks, although these techniques typically can introduce a high number of auxiliary variables and constraints, resulting in increasing preprocessing complexity (Anderson et al., 2020; Tsay et al., 2021; Lastrucci et al., 2026; Zha et al., 2026). These studies highlight the importance of carefully encoding the CPWL activation into disjunctions, rather than controlling the model size alone.

Kolmogorov–Arnold networks (KANs), which have emerged as an attractive substitute for MLPs, rely on a different factorization approach by assigning a learnable univariate function on each edge instead of placing a fixed nonlinear activation function on each neuron (Liu et al., 2025; Somvanshi et al., 2025). The original B-spline function enables excellent approximation performance but does not admit an exact MIP representation (Karia et al., 2025). To improve computational efficiency during training and inference, ReLU functions have subsequently been introduced to simplify the edge function evaluation (Warin, 2024; He, 2024; So & Yung, 2025; Qiu et al., 2025; Giroux & Fanelli, 2025). Recent work has established a direct connection between ReLU networks and CPWL-KANs by providing explicit constructions that transform one architecture into the other while preserving the mapping between inputs and outputs (Schoots et al., 2025). However, this equivalence does not necessarily imply comparable computational performance when these neural surrogates are embedded into mathematical programs. The respective architectures may differ substantially in formulation size and the underlying CPWL decompositions, both of which directly affect the downstream optimisation on the MILP problem.

Motivated by this, we develop a CPWL-KAN framework in which the parametrisation of the univariate edge functions is tailored to deterministic global optimisation. Rather than treating the CPWL representation solely as a means of simplifying KAN computation, we investigate how different decompositions of the same or related function classes impact predictive accuracy, formulation size, and continuous relaxation strength after embedding. Three variants, KAN-TITO, KAN-FITO, and KAN-TIFO, are introduced by selectively constraining the inner and outer weights of the ReLU encoding. Their expressivity properties are analysed, and both exact separate and joint encodings, specifically big-M and SOS2 formulations, are derived. An interval pruning approach for reducing redundant regions before optimisation is also introduced. The predictive and optimisation behaviour of the three variants is then evaluated across analytical benchmark functions and an industrial hydrogen production problem. These experiments assess the interplay among function parametrisation, predictive accuracy, formulation complexity, and deterministic global optimisation performance, compared to ReLU ANNs.

2 Background

Kolmogorov–Arnold networks (KANs) replace the scalar weights and node-wise activation terms in classical ANNs with learnable univariate functions on network edges (Liu et al., 2025). For a layer $l = 1, \dots, L$, the output of node j is defined as

$$x_j^{(l+1)} = \sum_{k=1}^{n_l} \phi_{kj}^{(l)}\left(x_k^{(l)}\right), \qquad j = 1, \dots, n_{l+1}, \tag{2.1}$$

where $\phi_{kj}^{(l)}$ denotes the learnable univariate function associated with the edge from node k in layer l to node j in layer $l+1$. Collecting vector elements in Eq. (2.1), the corresponding mapping is denoted by $\mathbf{\Phi}^{(l)} : \mathbb{R}^{n_l} \to \mathbb{R}^{n_{l+1}}$. Then, an L-layer KAN is expressed as the recursive composition

$$f_{\text{KAN}}(\mathbf{x}) = \mathbf{\Phi}^{(L)} \circ \mathbf{\Phi}^{(L-1)} \circ \cdots \circ \mathbf{\Phi}^{(1)}(\mathbf{x}). \tag{2.2}$$

The edge function parametrisation determines both the approximation flexibility and the mathematical structure of the network. Here, we adopt a general parametrisation the univariate CPWL function ϕ by a linear combination of ReLU terms,

$$\phi(x) = w_0^s x + b_0 + \sum_{i=1}^{I-1} w_i^s \,\text{ReLU}\left(w_i^t x + b_i\right), \tag{2.3}$$

Table 1: Variants of the CPWL encoding with ReLU terms in KAN.

Variant	CPWL function	Fixed parameters	# parameters
KAN-TITO	$w_0^s x + b_0 + \sum_{i=1}^{I-1} w_i^s \,\mathrm{ReLU}(w_i^t x + b_i)$	–	$3I-1$
KAN-FITO	$w_0^s x + b_0 + \sum_{i=1}^{I-1} w_i^s \,\mathrm{ReLU}(x + b_i)$	$w_i^t = 1$	$2I$
KAN-TIFO	$w_0^s x + b_0 + \sum_{i=1}^{I-1} \mathrm{ReLU}(w_i^t x + b_i)$	$w_i^s = 1$	$2I$

where I denotes the preset number of intervals; w_i^t and b_i define the inner affine weights of the ith ReLU term, and w_i^s its outer weight; and w_0^s and b_0 define the affine baseline. It is clear that each ReLU term $i = 1, \ldots, I-1$ for which $w_i^t \neq 0$ introduces a breakpoint at

$$\xi_i = -\frac{b_i}{w_i^t}, \tag{2.4}$$

thereby yielding a parametrisable partition of the input domain. The inner parameters w_i^t and b_i determine the locations and orientations of the ReLU functions, while the outer weights w_i^s encode their contributions to the resulting CPWL summand. For comparison, Schoots et al. (2025) introduced a similar parametrisation in the special case with $w_i^t = 1$.

3 CPWL-KAN Formulation and Analysis

3.1 Parametrisation Variants

Building on the CPWL encoding (2.3), two variants are constructed by selectively fixing the inner and/or outer weights of the ReLU terms. The CPWL summands, as well as respective total number of trainable parameters are summarised in Table 1.

The following proposition establishes that the additional trainable weights w_i^t of KAN-TITO do not add to the expressivity of KAN-FITO.

Proposition 3.1 (Equivalent expressive power of KAN-TITO and KAN-FITO)**.** *For the same network architecture and number of ReLU terms, KAN-TITO and KAN-FITO describe the same class of univariate CPWL functions on edges,*

$$\mathcal{F}_{\mathrm{TITO}}(I) = \mathcal{F}_{\mathrm{FITO}}(I). \tag{3.1}$$

See Appendix A.1 for a proof. Although it does not improve expressivity, KAN-TITO brings extra flexibility to network parametrisation. Conversely, KAN-FITO provides a more compact parametrisation, reducing the number of trainable parameters from $3I-1$ to $2I$ per CPWL summand.

Remark 3.2 (Relation to ReLU ANNs)**.** *Schoots et al. (2025) established that any ReLU ANN can be converted exactly into a KAN-FITO and vice versa; in particular, the transformation from KAN-FITO to ReLU ANN requires no any extra non-zero parameters. Both KAN-FITO, and KAN-TITO by Proposition 3.1, thus inherit the asymptotic approximation rate properties of ReLU ANNs (Yarotsky, 2017).*

By contrast, KAN-TIFO contains the same number of trainable parameters as KAN-FITO, but these two variants impose fundamentally different structural restrictions. In particular, the next proposition shows that KAN-TIFO defines a subclass of KAN-FITO (and KAN-TITO).

Proposition 3.3 (Expressivity limitation of KAN-TIFO)**.** *The univariate CPWL summands of KAN-TIFO are convex by construction. For the same network architecture and number of ReLU terms, therefore, KAN-TIFO describes a strict subclass of the univariate CPWL functions on edges expressed by KAN-FITO and KAN-TITO,*

$$\mathcal{F}_{\mathrm{TIFO}}(I) \subsetneq \mathcal{F}_{\mathrm{FITO}}(I) = \mathcal{F}_{\mathrm{TITO}}(I). \tag{3.2}$$

See Appendix A.2 for a proof. By setting the outer weights to $w_i^s = 1$, KAN-TIFO constrains the slope change introduced at every breakpoint to be nonnegative, implying convexity. This convexity at the individual edge level hinders the expressivity of KAN-TIFO. However, it does not generally imply convexity at the network level of KAN-TIFO itself.

Overall, these structural constraints can impact training and downstream optimisation differently. Although KAN-TITO and KAN-FITO share the same expressive power, the trainable inner weights in KAN-TITO provide greater parametrisation flexibility and compute alternative gradient directions during backpropagation process. In practice, gradient-descent algorithms thus result in different CPWL decompositions, with distinct breakpoints and slopes, which can ultimately affect the optimisation performance; and while KAN-TIFO restrict their edges to convex CPWL functions, they may yield a more structured edge representation that becomes advantageous for optimisation.

3.2 Optimisation Formulations

3.2.1 MIP Reformulations

Any CPWL edge function (2.3) is readily MIP representable. Similar to ReLU ANNs, one can apply the big-M method separately to each neuron in a given edge function, then recombine them to encode the full CPWL function. Another approach entails the description of a CPWL edge function in terms of its breakpoints and encoding of the function as a whole. Of the possible encodings, our focus is on the SOS2 formulation, although other formulations are possible, including the disaggregated convex hull (Vielma, 2015; Huchette & Vielma, 2023).

Separate ReLU Encoding. The big-M method allows for any ReLU term $r_i = \mathrm{ReLU}\,(w_i^t x + b_i)$ to be represented exactly with the following four linear inequality constraints

$$\begin{aligned} w_i^t x + b_i \le r_i \le w_i^t x + b_i - (1-\delta_i)L_i \\ 0 \le r_i \le \delta_i U_i, \end{aligned} \tag{3.3}$$

where the binary variable $\delta_i \in \{0,1\}$ encodes the activation state of the ith ReLU term; and L_i and U_i are valid lower and upper bounds on its preactivation value $w_i^t x + b_i$. The complete CPWL edge is simply recovered as

$$\phi(x) = w_0^s x + b_0 + \sum_{i=1}^{I-1} w_i^s r_i. \tag{3.4}$$

For a given range $x \in [\underline{x}, \overline{x}]$, exact pre-activation bounds are computed as

$$L_i = \min\{w_i^t \underline{x} + b_i, w_i^t \overline{x} + b_i\}, \qquad U_i = \max\{w_i^t \underline{x} + b_i, w_i^t \overline{x} + b_i\}. \tag{3.5}$$

An enclosure $[\underline{y}, \overline{y}]$ on the range of ϕ, in turn, is propagated as

$$\begin{aligned} \underline{y} &= b_0 + \min\{w_0^s \underline{x}, w_0^s \overline{x}\} + \sum_{i=1}^{I-1} \min\left\{w_i^s \max\{0, L_i\}, w_i^s \max\{0, U_i\}\right\}, \\ \overline{y} &= b_0 + \max\{w_0^s \underline{x}, w_0^s \overline{x}\} + \sum_{i=1}^{I-1} \max\left\{w_i^s \max\{0, L_i\}, w_i^s \max\{0, U_i\}\right\}. \end{aligned} \tag{3.6}$$

Although the pre-activation bounds (3.5) on each individual ReLU term are exact, the propagated enclosure $[\underline{y}, \overline{y}]$ over the CPWL edge function ϕ is generally conservative since the summands all depend on the same scalar variable x, a well-understood dependency problem in interval arithmetic (Moore et al., 2009).

Joint CPWL Encoding. For an exact encoding of the CPWL edge function ϕ in (2.3) on the same range $[\underline{x}, \overline{x}]$, it is convenient to collect all $R \ge 1$ distinct breakpoints (2.4), ordered as

$$\underline{x} = \xi_0 < \xi_1 < \cdots < \xi_R = \overline{x},$$

with their corresponding edge values

$$y_r = \phi(\xi_r), \qquad r = 0, \ldots, R.$$

Table 2: Formulation size for a CPWL edge with I segments.

Encoding	# Cont. vars.	# Bin. vars.	# Linear cons.	# Special cons.
Joint SOS2	$I+1$	0	$I+4$	1
Separate big-M	$I-1$	$I-1$	$4I-3$	0

The SOS2 formulation introduces continuous variables and linear constraints as

$$x = \sum_{r=0}^{R} \lambda_r \xi_r, \qquad y = \sum_{r=0}^{R} \lambda_r y_r, \qquad \sum_{r=0}^{R} \lambda_r = 1, \\ \lambda_r \geq 0, \quad r = 0, \ldots, R, \qquad \boldsymbol{\lambda} \in \mathrm{SOS2}, \tag{3.7}$$

with the crucial condition is that $(\lambda_0, \ldots, \lambda_R)$ forms an SOS2 set, where at most two adjacent λ_r variables may be nonzero. Access to the breakpoints also makes it possible to propagate an exact enclosure $[\underline{y}, \overline{y}]$ over the CPWL edge function ϕ by setting

$$\underline{y} = \min_{r=0,\ldots,R} y_r, \qquad \overline{y} = \max_{r=0,\ldots,R} y_r. \tag{3.8}$$

Network Encoding and Bound Propagation. A complete CPWL-KAN is encoded by appending the encoded edges and their combinations across all the KAN layers. Starting from layer $l = 1$, the output of any edge connecting node k in layer l to node j in layer $l+1$,

$$y_{kj}^{(l)} = \phi_{kj}^{(l)}\big(x_k^{(l)}\big), \tag{3.9}$$

is encoded using either the big-M method (3.4) or the SOS2 method (3.7). These outputs are then aggregated and connected to the input to node j in layer $l+1$ per (2.1), via the linear constraint

$$x_j^{(l+1)} = \sum_{k=1}^{n_l} y_{kj}^{(l)}, \tag{3.10}$$

and so on. The corresponding bounds on $y_{kj}^{(l)}$ are propagated using either (3.6) or (3.8), and further aggregated to yield bounds on $x_j^{(l+1)}$ as

$$\underline{x}_j^{(l+1)} = \sum_{k=1}^{n_l} \underline{y}_{kj}^{(l)}, \qquad \overline{x}_j^{(l+1)} = \sum_{k=1}^{n_l} \overline{y}_{kj}^{(l)}. \tag{3.11}$$

This procedure is applied recursively, starting from the prescribed bounds on the variables $\mathbf{x}$. The tightness of the resulting bounds on intermediate factors in a KAN have a direct impact of the strength of the continuous relaxations (either omitting the binary or the SOS2 restrictions) generated by a MIP solver, and ultimately the runtime.

Formulation Complexity. The two candidate encodings (3.4) and (3.7) also differ in the auxiliary variables and constraints introduced for each CPWL edge. Table 2 summarises their nominal formulation sizes for a CPWL function with I linear intervals. For a KAN containing N_E CPWL edge functions, the formulation size scales linearly with N_E. For each CPWL function, the joint SOS2 encoding introduces $I+1$ continuous auxiliary variables, $I+4$ linear constraints, and 1 SOS2 constraint. In comparison, the separate big-M encoding introduces $I-1$ continuous auxiliary variables, $I-1$ binary variables, and $4I-3$ linear constraints. It should be noted that these nominal formulation sizes can be reduced after pruning breakpoints that are unreachable, as discussed next.

3.2.2 Breakpoint Pruning

Following the bound propagation (3.11) in each layer, a pruning step is conducted to remove unreachable breakpoints as part of a KAN's encoding. For edge (k, j) in layer l, let $\mathcal{T}_{kj}^{(l)}$ denote the set of trained breakpoints and let $x_k^{(l)} \in \left[\underline{x}_k^{(l)}, \overline{x}_k^{(l)}\right]$ denote its propagated input domain. Only breakpoints within this domain can induce a change of intervals and therefore need to be retained:

$$\mathcal{T}_{kj,\mathrm{eff}}^{(l)} = \left\{\xi \in \mathcal{T}_{kj}^{(l)} : \underline{x}_k^{(l)} < \xi < \overline{x}_k^{(l)}\right\}. \tag{3.12}$$

For the joint SOS2 encoding, the two propagated domain endpoints and the breakpoints in $\mathcal{T}^{(l)}_{kj,\text{eff}}$ are retained to construct the ordered knots. For the separate big-M encoding, the propagated pre-activation bounds L_i and U_i determine the status of each ReLU term. A term with $U_i \leq 0$ is always inactive and can be removed. A term with $L_i \geq 0$ is always active and becomes affine over the propagated domain with following equation,

$$w_i^s \,\mathrm{ReLU}(w_i^t x + b_i) = w_i^s w_i^t x + w_i^s b_i, \tag{3.13}$$

so that its contribution can be integrated into the affine component. Only ReLU terms satisfying $L_i < 0 < U_i$ require explicit binary variables and big-M constraints.

In practice, pruning is applied layer by layer. After propagating the current bounds, unreachable breakpoints and fixed ReLU terms are removed before the reduced representation is passed to the next layer. This reduces the number of intervals and auxiliary variables in the final MIP formulation while preserving the same model prediction over the domain:

Remark 3.4 (Equivalence of interval pruning)**.** *Breakpoints outside the propagated domain do not affect the output of CPWL edge function. Therefore, pruning redundant intervals preserves the trained CPWL-KAN exactly over the prescribed optimisation domain.*

3.2.3 Continuous Relaxation Strength

With the joint and separate encoding, the continuous relaxations can differ substantially even with the same CPWL function. For the separate big-M encoding, the exact pre-activation bounds (3.5) provide the convex hull of each individual ReLU term. However, these separate convex hull relaxation don't generally recover the convex hull over the complete CPWL edge because the dependence on the same scalar input is ignored. In contrast, relaxing the adjacency requirement in the joint SOS2 encoding yields the convex hull of the complete CPWL function over a fixed input domain (Vielma, 2015; Huchette & Vielma, 2023). This gives the following relation.

Proposition 3.5 (Relaxation of joint and separate encodings)**.** *Consider a fixed univariate CPWL function $\phi(x)$ over $x \in [\underline{x}, \overline{x}]$, represented either by the joint SOS2 encoding in Eq. (3.7) or by the separate big-M encoding in Eqs. (3.3)–(3.4). Let $\mathcal{R}_{\text{joint}}$ and $\mathcal{R}_{\text{sep}}$ denote their continuous relaxations projected onto the input–output space (x, y). Then*

$$\mathcal{R}_{\text{joint}} \subseteq \mathcal{R}_{\text{sep}}. \tag{3.14}$$

The proof is provided in Appendix A.3. The inclusion can be strict for nonconvex CPWL edges. For example, consider $\phi(x) = \mathrm{ReLU}(x) - \mathrm{ReLU}(x-1)$ over $x \in [-1, 2]$. At $x = 1/2$, the separate encoding allows $0 \leq y \leq 1$, whereas the joint encoding gives the convex hull bound $1/4 \leq y \leq 3/4$. For KAN-TIFO, each edge is convex by construction. Hence, over the same input domain, the joint SOS2 and separate big-M encodings yield the same convex hull relaxation over the edge while associated bound propagation can lead to different results.

Remark 3.6 (Network relaxation)**.** *When the two encodings use the same edge domains and the same layerwise encoding equations, the relaxation over CPWL edge function in Proposition 3.5 can be directly extended to the complete KAN:*

$$\mathcal{R}_{\text{KAN,joint}} \subseteq \mathcal{R}_{\text{KAN,sep}}. \tag{3.15}$$

This relation isolates the effect of the encoding methods. When the separate and joint encodings use respective propagated bounds, differences in the complete network relaxation can additionally arise from the intermediate domains generated during bound propagation.

4 Experiments

The experiments evaluate CPWL-KANs from both approximation and optimisation. CPWL-KANs are designed with two layers as $P \to Q$ and $Q \to O$, with I_ϕ and I_Φ defining total intervals per edge, respectively. Three ReLU ANNs with two hidden layers are constructed for comparison. ANN-RM is defined to match the total numbers of nodes in the corresponding CPWL-KAN layers, counting the $I - 1$ ReLU terms and one affine term on each edge, which gives widths PQI_ϕ and QOI_Φ for the two hidden layers. ANN-TW and ANN-FW preserve the same relative widths while being scaled to match the parameter counts of KAN-TITO and KAN-FITO/TIFO, respectively. Three

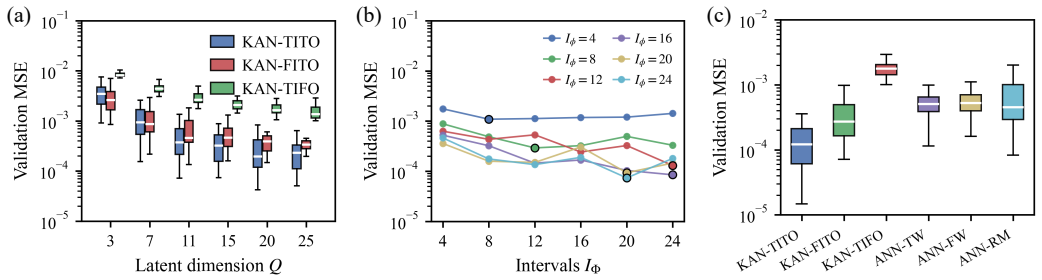


Figure 1: Predictive performance on Hartmann-6D. (a) Validation MSE of KANs across different latent dimensions Q, where each box summarises mean five-fold cross-validation MSE across candidate intervals at a fixed Q. (b) Mean five-fold cross-validation MSE of KAN-TITO under different I_ϕ and I_Φ for $Q = 15$. (c) Validation MSE of selected KAN and ANN architectures, where each box summarises mean five-fold cross-validation MSE across the considered hyperparameters.

analytical benchmarks (Surjanovic & Bingham, 2013), namely Ackley, Rosenbrock, and Hartmann, together with an engineering hydrogen production case (Lee et al., 2021), are considered. Details on architecture selection, training, optimisation, and case are provided in Appendix B.

4.1 Predictive Performance

Following the initialisation and training strategy in Appendix B.2, Figure 1 summarises the predictive performance on the Hartmann-6D benchmark. In Figure 1(a), KAN-TITO and KAN-FITO generally improve as the latent dimension Q increases and remain comparable, consistent with their equivalent representational capacity in Proposition 3.1. KAN-TITO nevertheless tends to achieve lower validation errors, indicating that its additional parametrisation flexibility can affect the trained models, while KAN-TIFO is less accurate because of convex edge restriction. Figure 1(b) shows a nonmonotonic dependence on I_ϕ and I_Φ: increasing the interval resolution initially improves accuracy, but further refinement provides limited benefit while increasing the optimisation complexity. Figure 1(c) compares the KAN and ANN models under the same training framework. KAN-TITO achieves the lowest validation errors overall, followed by KAN-FITO. Increasing model size does not consistently reduce MSE for ANNs: ANN-TW does not systematically outperform ANN-FW, while the much larger ANN-RM provides no consistent improvement over either. Thus, additional model capacity alone does not necessarily improve generalisation under the common training and early stopping procedure. As shown in the complete result in Appendix C.2, the CPWL-KANs usually require longer training time in total and per epoch. The respective ANNs tends to converge at early epoch. Large ANNs, for instance, the ANN-RM, possess greater representational capacity owing to much more parameters, but this does not consistently translate into improved predictive ability under the training procedure. Instead, ANN-RM often terminates relatively early, resulting in prediction errors comparable to or higher than other models.

4.2 Optimisation Performance

4.2.1 Relaxation Strength

The Ackley-2D benchmark is used to illustrate the continuous relaxation of KANs in Table 3. The joint SOS2 encoding consistently yields tighter relaxations than the separate big-M encoding, with the mean relaxation gaps under separate big-M encoding being approximately 2.2, 5.8, and 5.0 times those under joint SOS2 encoding for KAN-TITO, KAN-FITO, and KAN-TIFO, respectively. These results are consistent with the relaxation analysis in Section 3.2.3 and show that the differences in edge function can be further affected by layerwise bound propagation. The relaxation quality depends strongly on the parametrisation. As illustrated in Section 4.2.2 and Appendix C.3.2, the KAN-FITO tends to preserve more intervals, which results in larger relaxation through bound propagation under separate big-M encoding. The joint SOS2 encoding, on the opposite, alleviates it by constructing the convex hull over the CPWL edge, thereby leading to considerably smaller difference with the models. Besides, the KAN-TITO remains tight relaxation even with separate big-M encoding, indicating that the high accuracy is also important to maintain a good relaxation property.

Table 3: Normalised relaxation gaps of CPWL-KANs for the Ackley-2D benchmark. Ratio denotes the mean gap under the separate big-M encoding relative to the corresponding joint SOS2 encoding.

Model	Encoding	Mean	Median	P_{95}	Ratio
KAN-TITO	Joint SOS2	0.736	0.758	0.953	–
	Separate big-M	1.648	1.674	1.975	2.24
KAN-FITO	Joint SOS2	1.715	1.765	2.247	–
	Separate big-M	9.951	10.023	11.003	5.80
KAN-TIFO	Joint SOS2	1.919	1.942	2.576	–
	Separate big-M	9.637	9.691	10.271	5.02

Table 4: Interval statistics before and after pruning for the hydrogen case across independent trainings. Statistics are pooled across all edge functions, and total intervals are reported as the mean $\pm$ standard deviation.

Model	Layer	Before pruning	After pruning	Total intervals	Reduction (%)
KAN-TITO	ϕ	20	10.42 ± 3.27	218.80 ± 14.69	47.90
	Φ	20	10.33 ± 3.45	217.00 ± 33.84	48.33
KAN-FITO	ϕ	20	17.76 ± 1.27	373.00 ± 3.61	11.19
	Φ	20	5.43 ± 1.96	114.00 ± 13.10	72.86
KAN-TIFO	ϕ	20	7.79 ± 3.12	163.60 ± 8.23	61.05
	Φ	20	3.71 ± 2.10	78.00 ± 11.90	81.43

For KAN-TIFO, the observation does not contradict the convexity of individual edge functions, as different intermediate domains are obtained through respective bound propagations. Complete statistics are provided in Appendix C.3.1.

4.2.2 Computational Performance

The hydrogen production case is presented as a representative industrial problem for global optimisation performance, with the complete results for the remaining test cases in Appendix C.3.

Interval Pruning. The effect of interval pruning is illustrated in Table 4. The three parametrisations produce markedly different CPWL representations after training. KAN-TITO removes approximately 48% of the intervals in both layers, whereas KAN-FITO retains nearly 89% of the intervals in the first ϕ layer but undergoes substantially stronger pruning in the subsequent Φ layer, with a reduction of approximately 73%. Notably, retaining considerably more intervals does not translate into higher predictive accuracy: KAN-FITO remains less accurate than KAN-TITO, indicating that approximation quality depends on the learned breakpoints and slopes rather than on the number of retained intervals alone. KAN-TIFO exhibits the strongest pruning, removing approximately 61% and 81% of the intervals in the two layers, respectively. Overall, these results show that the parametrisation influences not only predictive performance but also the effective representation size after training, which subsequently determines downstream optimisation size.

The respective runtime regarding the interval pruning is compared in Figure 2(a)-(b) for both encodings. For joint SOS2, pruning generally reduces the runtime for optimisation, with the improvement being more evident for KAN-FITO and KAN-TIFO, which undergo substantial reductions in CPWL representations. The improvement for KAN-TITO is comparatively limited. In contrast, the runtimes before and after pruning under separate big-M encoding remain largely overlapping as redundant terms can also be identified by solver presolve, eliminating the computational difference between the formulations before and after pruning. Additional results are provided in Appendix C.3.2

Runtime Comparison and Objective Agreement. Figure 2(c)–(d) compares the optimisation performance across instances with different constraints. The joint SOS2 encoding generally solves the hydrogen instances faster than separate big-M counterparts, consistent with their tighter relaxations and tighter propagated bounds. KAN-TIFO achieves the shortest runtimes overall, reflecting the compact effective representations obtained after pruning, while KAN-TITO combines competitive runtimes with the smallest median and P_{95} absolute objective gaps. This indicates that KAN-TITO maintains good objective value agreement of across varying process requirements rather than only for the nominal problem, indicating a good fitting across whole networks. KAN-TIFO favours com-

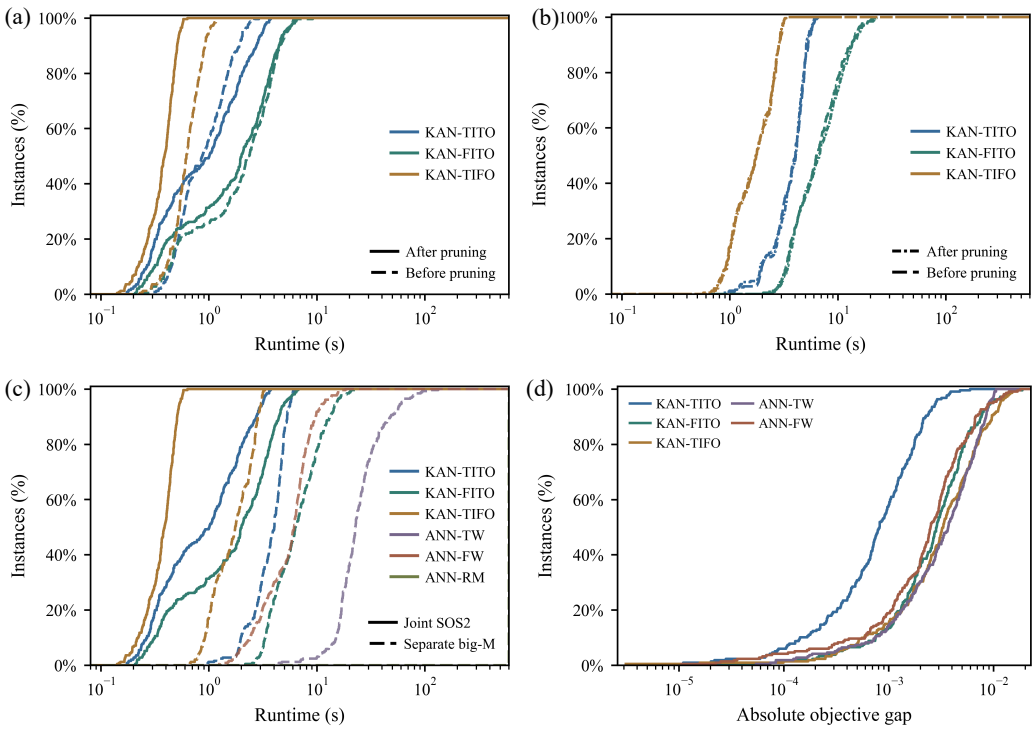


Figure 2: Computational performance for hydrogen case. (a)–(b) Effect of interval pruning on the joint SOS2 and separate big-M formulation, respectively. (c) Solved instances with respect to runtime. (d) Absolute objective gap distribution below a given ε to characterise the agreement between the objective values. No instance with ANN-RM is solved within 600 s.

putational efficiency at the expense of larger objective discrepancies, whereas KAN-TITO provides a more balanced trade-off between optimisation tractability and objective agreement.

5 Conclusion

This work develops CPWL-KANs for deterministic global optimisation and investigates how edge function parametrisation influences predictive accuracy, formulation structure, and optimisation performance. Three variants, KAN-TITO, KAN-FITO, and KAN-TIFO, admit exact joint SOS2 and separate big-M encodings. Theoretical analysis establishes the representational equivalence of KAN-TITO and KAN-FITO, the convexity of individual KAN-TIFO edge functions, and the no-weaker continuous relaxation of the joint encoding over common edge domains, while equivalent interval pruning reduces the effective formulation size without altering the trained surrogate. Across the considered benchmarks, the proposed KAN-TITO with additional parameters generally provides the strongest predictive accuracy and objective agreement, whereas KAN-TIFO often yields more compact representations and faster optimisation. Overall, the results show that surrogate design for downstream optimisation should account not only for predictive accuracy, but also for the parametrisation, effective CPWL representation, and optimisation formulation induced by the trained model.

6 AI Use Disclosure

In this work, we used generative AI tools only to improve the readability, grammar, and clarity of the manuscript. We reviewed all AI-assisted edits and take responsibility for the final content of this work, including all text produced with the aid of generative AI.

REFERENCES

Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 146–155. PMLR, 06–11 Aug 2017.

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1–2):3–39, 2020. doi: 10.1007/s10107-020-01474-5.

Atharv Bhosekar and Marianthi Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, 108:250–267, January 2018. doi: 10.1016/j.compchemeng.2017.09.017.

Rudy Bunel, Alessandro De Palma, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip H. S. Torr, and M. Pawan Kumar. Lagrangian decomposition for neural network verification. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, volume 124 of *Proceedings of Machine Learning Research*, pp. 370–379. PMLR, 2020.

Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, April 2018. doi: 10.1007/s10601-018-9285-6.

GAMS Development Corporation. GAMSPy: Algebraic modeling interface to GAMS. https://github.com/GAMS-dev/gamspy, 2026. Accessed: 2026.

J Giroux and C Fanelli. Uncertainty quantification with bayesian higher order relu-kans. *Machine Learning: Science and Technology*, 6(1):015073, March 2025. doi: 10.1088/2632-2153/adbeb7.

Bjarne Grimstad and Henrik Andersson. ReLU networks as surrogate models in mixed-integer linear programs. *Computers & Chemical Engineering*, 131:106580, 2019. doi: 10.1016/j.compchemeng.2019.106580.

Juncai He. On the optimal expressive power of relu dnns and its application in approximation with the kolmogorov superposition theorem. *IEEE Transactions on Neural Networks and Learning Systems*, 36(8):13886–13899, December 2024. doi: 10.1109/tnnls.2024.3514126.

Joey Huchette and Juan Pablo Vielma. Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools. *Operations Research*, 71(5):1835–1856, 2023. doi: 10.1287/opre.2019.1973.

Tanuj Karia, Giacomo Lastrucci, and Artur M. Schweidtmann. Deterministic global optimization over trained kolmogorov–arnold networks, 2025. arXiv:2503.02807.

Jan Kronqvist, Ruth Misener, and Calvin Tsay. Between steps: Intermediate relaxations between big-m and convex hull formulations. In Peter J. Stuckey (ed.), *Lecture Notes in Computer Science*, volume 12735 of *Lecture Notes in Computer Science*, pp. 299–314. Springer Nature, June 2021. doi: 10.1007/978-3-030-78230-6_19.

Giacomo Lastrucci, Tanuj Karia, Victor Schulte, Dominik Bongartz, and Artur M. Schweidtmann. Pruning for efficient deterministic global optimization over trained ReLU neural networks, 2026. arXiv:2603.23299.

Andrew Lee, Jaffer H Ghouse, John C Eslick, Carl D Laird, John D Siirola, Miguel A Zamarripa, Dan Gunter, John H Shinn, Alexander W Dowling, Debangsu Bhattacharyya, Lorenz T Biegler, Anthony P Burgard, and David C Miller. The IDAES process modeling framework and model library—Flexibility for process simulation and optimization. *Journal of Advanced Manufacturing and Processing*, 3, 2021. doi: 10.1002/amp2.10095.

Yu Liu, Jan Kronqvist, and Fabricio Oliveira. Input convex neural networks as surrogates in mathematical optimisation, 2026. arXiv:2608.09707.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–arnold networks. In *International Conference on Learning Representations*, 2025.

Ruth Misener and Lorenz Biegler. Formulating data-driven surrogate models for process optimization. *Computers & Chemical Engineering*, 179:108411, November 2023. doi: 10.1016/j.compchemeng.2023.108411.

Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*. Society for Industrial & Applied Mathematics (SIAM), January 2009. doi: 10.1137/1.9780898717716.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8026–8037, 2019.

Qi Qiu, Tao Zhu, Helin Gong, Liming Chen, and Huansheng Ning. Relu-kan: New kolmogorov-arnold networks that only need matrix addition, dot multiplication, and relu. In *2025 IEEE Smart World Congress (SWC)*, volume 00, pp. 1686–1694. Institute of Electrical and Electronics Engineers (IEEE), August 2025. doi: 10.1109/swc65939.2025.00262.

Nandi Schoots, Mattia Jacopo Villani, and Niels uit de Bos. Relating piecewise linear Kolmogorov Arnold networks to ReLU networks. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan (eds.), *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 199–207, 2025.

Artur M. Schweidtmann and Alexander Mitsos. Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory and Applications*, 180(3):925–948, 2019. doi: 10.1007/s10957-018-1396-0.

Chi Chiu So and Siu Pang Yung. Higher-order-ReLU-KANs (HRKANs) for solving physics-informed neural networks (PINNs) more accurately, robustly and faster. In *2025 IEEE 6th Annual World AI IoT Congress*, pp. 1035–1042, 2025. doi: 10.1109/AIIoT65859.2025.11105234.

I.M Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, January 1967. doi: 10.1016/0041-5553(67)90144-9.

Shriyank Somvanshi, Syed Aaqib Javed, Monzurul Islam, Diwas Pandit, and Subasish Das. A survey on kolmogorov-arnold network. *ACM Computing Surveys*, 58(2):1–35, September 2025. doi: 10.1145/3743128.

S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets, 2013. URL http://www.sfu.ca/~ssurjano. Accessed: September 2, 2026.

Christian Tjandraatmadja, Ross Anderson, Joey Huchette, Will Ma, Krunal Patel, and Juan Pablo Vielma. The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21675–21686, 2020.

Calvin Tsay, Jan Kronqvist, Alexander Thebelt, and Ruth Misener. Partition-based formulations for mixed-integer optimization of trained ReLU neural networks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 3068–3080. Curran Associates, Inc., 2021.

Aldo Vecchietti, Sangbum Lee, and Ignacio E. Grossmann. Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations. *Computers & Chemical Engineering*, 27(3):433–448, March 2003. doi: 10.1016/s0098-1354(02)00220-x.

Juan Pablo Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57 (1):3–57, January 2015. doi: 10.1137/130915303.

Juan Pablo Vielma. Embedding formulations and complexity for unions of polyhedra. *Management Science*, 64(10):4721–4734, October 2018. doi: 10.1287/mnsc.2017.2856.

Keliang Wang, Leonardo Lozano, Carlos Cardonha, and David Bergman. Optimizing over an ensemble of trained neural networks. *INFORMS Journal on Computing*, 35(3):652–674, May 2023. doi: 10.1287/ijoc.2023.1285.

Xavier Warin. P1-KAN: An effective Kolmogorov–Arnold network with application to hydraulic valley optimization, 2024. arXiv:2410.03801.

Clara Witte, Jannik T. Lüthje, Victor Schulte, Alexander Mitsos, and Dominik Bongartz. Deterministic global optimization with trained neural networks: What is the benefit of the envelope of single neurons? *Optimization Online*, 2025.

Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017. doi: https://doi.org/10.1016/j.neunet.2017.07.002.

Yanlin Zha, Mario Eduardo Villanueva, Boris Houska, and Benoît Chachuat. Relaxation via separable estimators: Arithmetic and implementation, 2026. arXiv:2605.10854.

A Property of CPWL-KAN encoding

A.1 Proof of Representation Equivalence of KAN-TITO and KAN-FITO

Consider a CPWL function in KAN-TITO:

$$\phi_{\mathrm{TITO}}(x) = w_0^s x + b_0 + \sum_{i=1}^{I-1} w_i^s \,\mathrm{ReLU}(w_i^t x + b_i). \tag{A.1}$$

For $w_i^t > 0$, the positive homogeneity of the ReLU function gives

$$w_i^s \,\mathrm{ReLU}(w_i^t x + b_i) = w_i^s w_i^t \,\mathrm{ReLU}\left(x + \frac{b_i}{w_i^t}\right). \tag{A.2}$$

For $w_i^t < 0$, using $\mathrm{ReLU}(-u) = \mathrm{ReLU}(u) - u$ gives

$$w_i^s \,\mathrm{ReLU}(w_i^t x + b_i) = -w_i^s w_i^t \,\mathrm{ReLU}\left(x + \frac{b_i}{w_i^t}\right) + w_i^s w_i^t x + w_i^s b_i. \tag{A.3}$$

Hence, every ReLU component with a nonzero inner weight can be rewritten with a fixed inner weight equal to one, while the additional linear and constant terms arising when $w_i^t < 0$ can be absorbed into the affine term $w_0^s x + b_0$. If $w_i^t = 0$, the corresponding ReLU component is constant and can likewise be absorbed into b_0. Therefore, every KAN-TITO CPWL function admits an equivalent KAN-FITO representation with the same number of available ReLU components, which establishes $\mathcal{F}_{\mathrm{TITO}}(I) \subseteq \mathcal{F}_{\mathrm{FITO}}(I)$.

Conversely, KAN-FITO is a special case of KAN-TITO obtained by setting $w_i^t = 1$ for all ReLU components. Thus, $\mathcal{F}_{\mathrm{FITO}}(I) \subseteq \mathcal{F}_{\mathrm{TITO}}(I)$. Combining the two inclusions yields

$$\mathcal{F}_{\mathrm{TITO}}(I) = \mathcal{F}_{\mathrm{FITO}}(I), \tag{A.4}$$

which proves Proposition 3.1.

A.2 Proof of Convexity of CPWL function in KAN-TIFO

The ReLU function, defined as follows:

$$\mathrm{ReLU}(w_i^t x + b_i) = \max\{0, w_i^t x + b_i\} \tag{A.5}$$

is convex because it is the pointwise maximum of two affine functions. Since convexity is preserved under composition with affine mappings, for arbitrary w_i^t and b_i, $\mathrm{ReLU}(w_i^t x + b_i)$ is convex in x.

Therefore, $\sum_{i=1}^{I-1} \mathrm{ReLU}(w_i^t x + b_i)$ is also convex because all ReLU components have the fixed nonnegative outer weight $w_i^s = 1$. The remaining term $w_0^s x + b_0$ is affine, and adding an affine function preserves convexity. Hence, $\phi_{\mathrm{TIFO}}(x)$ is convex over its entire domain.

This property applies to each individual CPWL function in KAN-TIFO. Convexity of the complete KAN is not guaranteed because convexity is not generally preserved under arbitrary compositions of convex functions.

A.3 Proof of Relaxation Strength for Joint and Separate Encodings

Consider a fixed univariate CPWL edge function

$$\phi(x) = w_0^s x + b_0 + \sum_{i=1}^{I-1} w_i^s g_i(x), \qquad g_i(x) = \mathrm{ReLU}(w_i^t x + b_i), \tag{A.6}$$

over the common input domain $\mathcal{D} = [\underline{x}, \overline{x}]$. Introduce the lifted variable vector $\boldsymbol{z} = (z_1, \ldots, z_{I-1})$, where $z_i = g_i(x)$, and define the affine coupling set

$$\mathcal{A} = \left\{ (x, \boldsymbol{z}, y) : x \in \mathcal{D},\ y = w_0^s x + b_0 + \sum_{i=1}^{I-1} w_i^s z_i \right\}. \tag{A.7}$$

For each ReLU term, define

$$\mathcal{S}_i = \{ (x, \boldsymbol{z}, y) : x \in \mathcal{D},\ z_i = g_i(x) \}. \tag{A.8}$$

The exact lifted feasible set of the complete CPWL function is therefore

$$\mathcal{S} = \mathcal{A} \cap \bigcap_{i=1}^{I-1} \mathcal{S}_i. \tag{A.9}$$

With the exact pre-activation bounds in Eq. (3.5), relaxing the binary variable in the separate big-M encoding yields the convex hull of each individual ReLU term. Hence, the corresponding lifted continuous relaxation is

$$\widehat{\mathcal{R}}_{\mathrm{sep}} = \mathcal{A} \cap \bigcap_{i=1}^{I-1} \mathrm{conv}(\mathcal{S}_i). \tag{A.10}$$

Since $\mathcal{S} \subseteq \mathcal{S}_i$ for every i,

$$\mathrm{conv}(\mathcal{S}) \subseteq \mathrm{conv}(\mathcal{S}_i), \qquad i = 1, \ldots, I-1. \tag{A.11}$$

Moreover, $\mathcal{S} \subseteq \mathcal{A}$, and $\mathcal{A}$ is affine and therefore convex, giving

$$\mathrm{conv}(\mathcal{S}) \subseteq \mathcal{A}. \tag{A.12}$$

Combining these relations yields

$$\mathrm{conv}(\mathcal{S}) \subseteq \mathcal{A} \cap \bigcap_{i=1}^{I-1} \mathrm{conv}(\mathcal{S}_i) = \widehat{\mathcal{R}}_{\mathrm{sep}}. \tag{A.13}$$

Let $\mathcal{P}_{x,y}$ denote the projection onto the input–output space (x, y). Since linear projection commutes with convexification,

$$\mathcal{P}_{x,y}\left(\mathrm{conv}(\mathcal{S})\right) = \mathrm{conv}\left(\mathcal{P}_{x,y}(\mathcal{S})\right). \tag{A.14}$$

Because

$$\mathcal{P}_{x,y}(\mathcal{S}) = \{ (x, \phi(x)) : x \in \mathcal{D} \}, \tag{A.15}$$

we obtain

$$\mathcal{P}_{x,y}\left(\mathrm{conv}(\mathcal{S})\right) = \mathrm{conv}\left\{ (x, \phi(x)) : x \in \mathcal{D} \right\}. \tag{A.16}$$

For the joint SOS2 encoding, relaxing the adjacency requirement permits arbitrary convex combinations of the ordered knots. Since the CPWL function consists of the joint adjacent knots, its continuous relaxation is exactly

$$\mathcal{R}_{\mathrm{joint}} = \mathrm{conv}\left\{ (x, \phi(x)) : x \in \mathcal{D} \right\}. \tag{A.17}$$

For the separate big-M encoding,

$$\mathcal{R}_{\text{sep}} = \mathcal{P}_{x,y}\left(\widehat{\mathcal{R}}_{\text{sep}}\right). \tag{A.18}$$

Projecting Eq. (A.13) onto (x, y) and using Eq. (A.17) therefore gives

$$\mathcal{R}_{\text{joint}} \subseteq \mathcal{R}_{\text{sep}}, \tag{A.19}$$

which proves Proposition 3.5.

B Experimental Setup

All surrogate models are implemented and trained using PyTorch (Paszke et al., 2019), while the corresponding optimisation problems are formulated using GAMSPy (GAMS Development Corporation, 2026). All experiments are conducted within the Windows Subsystem for Linux 2 (WSL 2) with Ubuntu 24.04.4 LTS on an HP Z8 G4 workstation equipped with an Intel Xeon Gold 6226R CPU at 2.90 GHz, 64 GB of system memory, and an NVIDIA RTX A4000 GPU with 16 GB of graphics memory. The surrogate models are trained using the GPU, whereas the optimisation experiments are executed on the CPU.

B.1 Data Preparation

For each case, a Sobol sequence (Sobol', 1967) is used to sample the prescribed input domain for dataset generation. A total of 2^{12} samples are generated, and the corresponding outputs are obtained by evaluating the analytical function or process simulator associated with each case. The dataset is split into 80% for model development and 20% for final evaluation. The input variables are normalised to $[0, 1]$.

B.2 Model Initialisation and Training

For each CPWL edge function with I linear intervals over an input domain $[\underline{x}, \overline{x}]$, the $I - 1$ initial breakpoints are uniformly distributed as

$$\xi_i^{(0)} = \underline{x} + \frac{i}{I}\left(\overline{x} - \underline{x}\right), \qquad i = 1, \ldots, I-1. \tag{B.1}$$

For the first layer, $[\underline{x}, \overline{x}]$ is given by the prescribed input domain. For each subsequent layer, the corresponding node ranges are obtained by propagating the bounds through the preceding initialised layer, and these propagated ranges are then used to initialise the breakpoints of the outgoing CPWL edge functions. According to the breakpoint definition in Eq. (2.4), $\xi_i = -b_i/w_i^t$, the corresponding bias is initialised as

$$b_i^{(0)} = -w_i^{t,(0)}\xi_i^{(0)}. \tag{B.2}$$

The inner and outer weights, when trainable, are initialised using Kaiming normal initialisation. In KAN-TITO, both w_i^t and w_i^s are trainable; KAN-FITO fixes $w_i^t = 1$, whereas KAN-TIFO fixes $w_i^s = 1$. The biases b_i remain trainable in all three variants, allowing the breakpoint locations to move during training. The affine coefficient w_0^s is initialised using Kaiming normal initialisation, and b_0 is set to zero.

All surrogate models are trained using the Adam optimiser with mean squared error (MSE) loss for a maximum of 1,000 epochs. Early stopping is applied with a patience of 20 epochs, and the model parameters corresponding to the lowest validation loss are retained. The learning rate and batch size are determined through hyperparameter tuning.

B.3 Architecture and Hyperparameter Tuning

The CPWL-KAN architecture is determined using the 80% dataset, with KAN-TITO used as the reference variant. For the two layers $P \to Q \to O$, the input and output dimensions P and O are fixed by the corresponding regression problem. The latent dimension Q is first determined as the candidate achieving the lowest mean five-fold cross-validation MSE. With Q fixed, the numbers of intervals I_ϕ and I_Φ are then varied, and their values are selected near the empirical elbow of the MSE loss, where further increases in interval resolution provide only limited improvement

Table 5: Grid search space for architecture and training hyperparameters.

Category	Hyperparameter	Candidate values
Architecture	Hidden nodes Q	$\{3, 7, 11, 15, 20, 25\}$
Architecture	intervals I_ϕ, I_Φ	$\{4, 8, 12, 16, 20, 24\}$
Training	Learning rate	$\{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$
Training	Batch size	$\{64, 128, 256, 512\}$

Table 6: Summary of the validation cases.

Case	Characteristics
Ackley	Multimodal nonlinear landscape
Rosenbrock	Narrow and curved valley
Hartmann	Multimodal and nonconvex landscape
Hydrogen production	Industrial black-box process simulation

in predictive accuracy while increasing the size of the subsequent optimisation formulation. The resulting configuration $(P, Q, O, I_\phi, I_\Phi)$ is shared by KAN-TITO, KAN-FITO, and KAN-TIFO to enable comparison under the same network architecture.

The respective KAN architecture is subsequently used to construct the three ANN baselines according to the matching rules described in the main text. The learning rate and batch size are tuned separately for each surrogate model using grid search, while the remaining training settings are kept fixed. The candidate values considered for architecture and training hyperparameters are summarised in Table 5.

B.4 Optimisation Settings

The trained surrogate models are embedded into the corresponding optimisation problems and solved primarily using Gurobi 13.0.0. CPLEX, COPT, CBC, and Xpress are additionally considered in the supplementary solver comparison. A relative optimality tolerance of 10^{-4} and a time limit of 600 s are imposed for each optimisation instance, with the solver restricted to 32 CPU threads. The same solver settings are used across all surrogate architectures and encoding formulations to ensure a consistent comparison. The separate big-M and joint SOS2 encodings, together with the associated bound propagation and interval pruning procedures, follow the formulations described in Section 3.2.

B.5 Case Descriptions

Four cases are considered, including three analytical benchmark functions, namely Ackley, Rosenbrock, and Hartmann, and an industrial hydrogen production case. The analytical benchmarks provide controlled nonlinear landscapes with different approximation and optimisation characteristics, while the hydrogen production case evaluates the proposed approach on a black-box process simulation. An overview of the test cases is provided in Table 6, with their detailed definitions given below.

B.5.1 Benchmarks

Ackley Function. The d-dimensional Ackley function is defined over $\mathbf{x} \in [0, 1]^d$ as

$$f_{\mathrm{A}}(\mathbf{x}) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + e, \tag{B.3}$$

where $a = 20$, $b = 0.2$, and $c = 2\pi$. The function provides a multimodal nonlinear landscape and is used to assess the ability of the surrogate models to represent nonconvex responses with multiple local features.

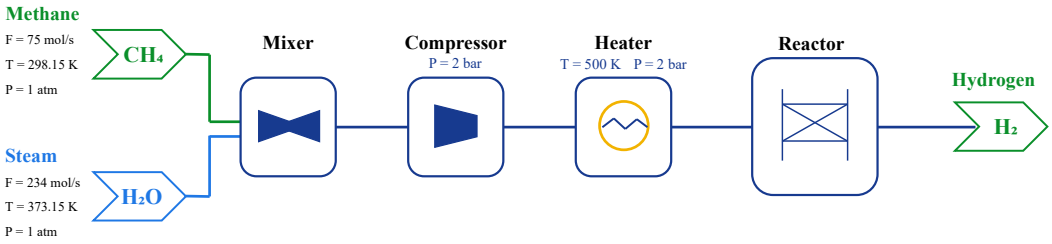


Figure 3: Schematic of the hydrogen production process, including feed mixing, compression, heating, reaction, and hydrogen product.

Rosenbrock Function. The d-dimensional Rosenbrock function is defined over $\mathbf{x} \in [0,1]^d$ as

$$f_{\mathrm{R}}(\mathbf{x}) = \sum_{i=1}^{d-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (1 - x_i)^2 \right]. \tag{B.4}$$

The Rosenbrock function is characterised by a narrow and curved valley leading to the global optimum, with strong interactions between neighbouring variables. It is therefore used to assess the ability of the surrogate models to represent strongly coupled nonlinear responses.

Hartmann Function. The six-dimensional Hartmann function is defined over $\mathbf{x} \in [0,1]^6$ as

$$f_{\mathrm{H}}(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left[-\sum_{j=1}^{6} A_{ij} \left(x_j - V_{ij} \right)^2 \right]. \tag{B.5}$$

The function contains several local minima with different basin sizes and is used as a higher-dimensional nonconvex benchmark. The parameter values used in Eq. (B.5) are

$$\begin{aligned} \boldsymbol{\alpha} &= [1.0 \quad 1.2 \quad 3.0 \quad 3.2], \\ \mathbf{A} &= \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, \\ \mathbf{V} &= 10^{-4} \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}. \end{aligned} \tag{B.6}$$

B.5.2 Hydrogen Production

The hydrogen production process is adapted from the IDAES-PSE framework (Lee et al., 2021) as an industrial case study. As illustrated in Figure 3, the process consists of a feed mixer, a compressor, a heater, and a reactor for hydrogen production from methane and steam. The process simulator is treated as a black-box model that maps the operating variables to the corresponding process responses without an explicit analytical representation. Consequently, the global optimum is not known a priori, and direct optimisation through repeated process simulations can be computationally demanding. The decision variables $\mathbf{x}$ and corresponding process outputs $\mathbf{y}$ are summarised in Table 7.

B.5.3 Instance Generation

For each analytical benchmark, the surrogate optimisation problem for model m is defined as

$$\min_{\boldsymbol{x}} \quad \hat{f}_m(\boldsymbol{x}) \qquad \text{s.t.} \qquad \boldsymbol{x} \in \mathcal{X}, \tag{B.7}$$

where $\hat{f}_m$ denotes the trained surrogate of the corresponding analytical function f, and $\mathcal{X}$ is the original input domain. To evaluate optimisation performance over different regions of the response

Table 7: Decision variables, process outputs, and nominal optimisation specifications for the hydrogen production case.

Variable	Description	Unit	Bounds / Constraint
Decision variables			
x_1	Methane feed molar flowrate	$\mathrm{mol\,s^{-1}}$	$50 \leq x_1 \leq 100$
x_2	Heater outlet temperature	K	$400 \leq x_2 \leq 600$
x_3	Steam-to-methane molar ratio	–	$1 \leq x_3 \leq 5$
Process outputs			
y_1	Reactor heat duty	W	$\min\ y_1$
y_2	Product molar flowrate	$\mathrm{mol\,s^{-1}}$	$y_2 \geq 500$
y_3	Hydrogen mole fraction in product	–	$0.5 \leq y_3 \leq 0.6$

Table 8: Selected model architectures and parameter counts for all cases.

Model / Statistic	Ackley (2D)	Rosenbrock (2D)	Hartmann (6D)	Hydrogen
KAN config. $(P, Q, O, I_\phi, I_\Phi)$	$(2, 20, 1, 12, 12)$	$(2, 15, 1, 16, 12)$	$(6, 15, 1, 24, 20)$	$(3, 7, 3, 20, 20)$
KAN-TITO params.	2,100	1,935	7,275	2,478
KAN-FITO/TIFO params.	1,440	1,320	4,920	1,680
ANN-TW config.	$[2, 62, 30, 1]$	$[2, 67, 25, 1]$	$[6, 205, 28, 1]$	$[3, 46, 46, 3]$
ANN-TW params.	2,107	1,927	7,232	2,487
ANN-FW config.	$[2, 50, 25, 1]$	$[2, 56, 20, 1]$	$[6, 164, 23, 1]$	$[3, 37, 37, 3]$
ANN-FW params.	1,451	1,329	4,967	1,668
ANN-RM config.	$[2, 480, 240, 1]$	$[2, 480, 180, 1]$	$[6, 2160, 300, 1]$	$[3, 420, 420, 3]$
ANN-RM params.	117,121	88,201	663,721	179,763

landscape, multiple instances are generated by restricting the feasible region around sampled centres $\boldsymbol{x}_0^{(k)}$:

$$\mathcal{X}^{(k)} = \left\{ \boldsymbol{x} \in \mathcal{X} : x_{0,i}^{(k)} - r \leq x_i \leq x_{0,i}^{(k)} + r, \quad i = 1, \ldots, d \right\}. \tag{B.8}$$

For each instance, the original analytical function f and the corresponding trained surrogate $\hat{f}_m$ are optimised over the same feasible region $\mathcal{X}^{(k)}$, enabling their optimisation results to be compared under identical domain restrictions.

For the hydrogen production case, the instances are generated by varying the process requirements rather than restricting the input domain locally. Operating points are sampled over the prescribed decision variable domain and evaluated using the process simulator. The resulting product flowrate and hydrogen mole fraction are used to define specific process constraints, while the reactor heat duty is minimised subject to these requirements and the original operating bounds.

For each benchmark, 128 instance are generated while for hydrogen production case, 256 instance are implemented. For each instance, the original analytical problem is solved with Gurobi and process simulation problem is solved using IPOPT to obtain a reference objective value under the same instance specifications as the surrogate optimisation problem. Since IPOPT is a local nonlinear optimisation solver, these reference solutions are not guaranteed to be globally optimal and are therefore used only as reference solutions for assessing agreement with the surrogate optimisation results.

C Additional Results and Discussions

C.1 Model Architectures and Statistics

The selected model architectures and corresponding parameter counts for all the cases are summarised in Table 8. The reported parameter counts include all trainable weights and biases. For the CPWL-KANs, the numbers of intervals are additionally reported because they directly determine the size of the subsequent optimisation formulations.

Table 9: Average training cost of the surrogate models. Training time and the stopping epoch are reported as mean ± sample standard deviation. The stopping epoch denotes the number of epochs executed before early stopping or reaching the maximum of 1000 epochs.

Model	Metric	Ackley (2D)	Rosenbrock (2D)	Hartmann (6D)	Hydrogen
KAN-TITO	Time (s)	128.19 ± 4.87	119.43 ± 13.57	356.80 ± 64.28	217.76 ± 29.69
	Epoch	988.6 ± 25.5	687.8 ± 75.9	866.8 ± 150.8	359.4 ± 54.4
KAN-FITO	Time (s)	98.71 ± 12.78	96.36 ± 8.11	364.99 ± 43.49	247.01 ± 67.75
	Epoch	546.2 ± 71.3	860.4 ± 81.1	904.2 ± 104.6	420.6 ± 112.8
KAN-TIFO	Time (s)	100.47 ± 12.39	117.43 ± 30.01	284.08 ± 112.47	205.82 ± 49.19
	Epoch	798.8 ± 101.6	697.4 ± 175.9	708.0 ± 275.9	347.4 ± 70.0
ANN-TW	Time (s)	16.47 ± 4.35	29.17 ± 6.06	197.81 ± 37.79	50.99 ± 16.46
	Epoch	278.2 ± 72.8	285.8 ± 59.5	799.4 ± 134.8	121.8 ± 41.4
ANN-FW	Time (s)	26.93 ± 7.23	30.23 ± 4.55	222.17 ± 30.07	104.43 ± 31.54
	Epoch	268.0 ± 71.5	508.6 ± 76.6	922.0 ± 120.5	254.2 ± 78.6
ANN-RM	Time (s)	14.83 ± 1.92	29.68 ± 5.04	75.94 ± 32.06	29.39 ± 7.41
	Epoch	245.8 ± 32.9	289.0 ± 44.9	310.0 ± 129.9	123.0 ± 30.7

C.2 Predictive Performance

The corresponding training costs are summarised in Table 9. Under the early stopping strategy, the CPWL-KAN models generally require longer training times than the ANN baselines. This difference arises from both more training epochs in several cases and a higher computational cost per epoch, consistent with the CPWL operations each edge required by the KAN architectures. The ANN results further show that parameter count alone does not determine training cost: despite its substantially larger size, ANN-RM typically stops after fewer epochs and therefore exhibits comparable or shorter training times than ANN-TW and ANN-FW. More generally, the reported training costs reflect the combined effects of model architecture, computational cost per epoch, and convergence behaviour under early stopping.

Additional evidence from the remaining test cases is provided in Figure 4, complementing the Hartmann-6D results discussed in Section 4.1. Across the Ackley, Rosenbrock, and hydrogen production cases, increasing the latent dimension Q generally provides the largest reduction in validation MSE at small to moderate values of Q, after which the improvement becomes less pronounced. This is particularly evident for the hydrogen production case, where the validation error approaches a plateau for $Q \geq 7$, while KAN-FITO and KAN-TIFO can exhibit slightly higher errors at larger Q. A similarly non-monotonic dependence is observed for the interval candidates in Figure 4(b), (e), and (h). Increasing I_ϕ and I_Φ is beneficial when the CPWL resolution is relatively coarse, whereas further refinement provides diminishing or inconsistent improvements once sufficient resolution is reached. These results reinforce the observation from the Hartmann case that increasing architectural capacity does not necessarily translate into a proportional improvement in predictive accuracy.

Table 10: Predictive performance of the models across all the cases, reported as mean ± standard deviation of the normalised test MSE. Bold and underlined values indicate the best and worst performance for each one, respectively.

Model	Ackley (2D)	Rosenbrock (2D)	Hartmann (6D)	Hydrogen
KAN-TITO	$\mathbf{(2.30 \pm 0.66)10^{-6}}$	$\mathbf{(6.60 \pm 1.92)10^{-6}}$	$\mathbf{(7.13 \pm 3.10)10^{-5}}$	$\mathbf{(4.74 \pm 1.38)10^{-7}}$
KAN-FITO	$(5.29 \pm 0.75)10^{-6}$	$(1.34 \pm 0.63)10^{-5}$	$(1.60 \pm 0.41)10^{-4}$	$(3.16 \pm 0.91)10^{-6}$
KAN-TIFO	$(6.47 \pm 2.04)10^{-6}$	$\underline{(1.19 \pm 0.55)10^{-4}}$	$\underline{(1.51 \pm 0.22)10^{-3}}$	$(5.73 \pm 1.53)10^{-6}$
ANN-TW	$(5.25 \pm 1.37)10^{-5}$	$(3.24 \pm 1.19)10^{-5}$	$(2.52 \pm 1.03)10^{-4}$	$(3.93 \pm 0.53)10^{-6}$
ANN-FW	$\underline{(6.88 \pm 1.11)10^{-5}}$	$(3.95 \pm 2.58)10^{-5}$	$(2.13 \pm 0.39)10^{-4}$	$(3.52 \pm 0.73)10^{-6}$
ANN-RM	$(6.02 \pm 0.95)10^{-5}$	$(4.45 \pm 3.05)10^{-5}$	$(1.45 \pm 0.64)10^{-4}$	$\underline{(1.09 \pm 0.30)10^{-5}}$

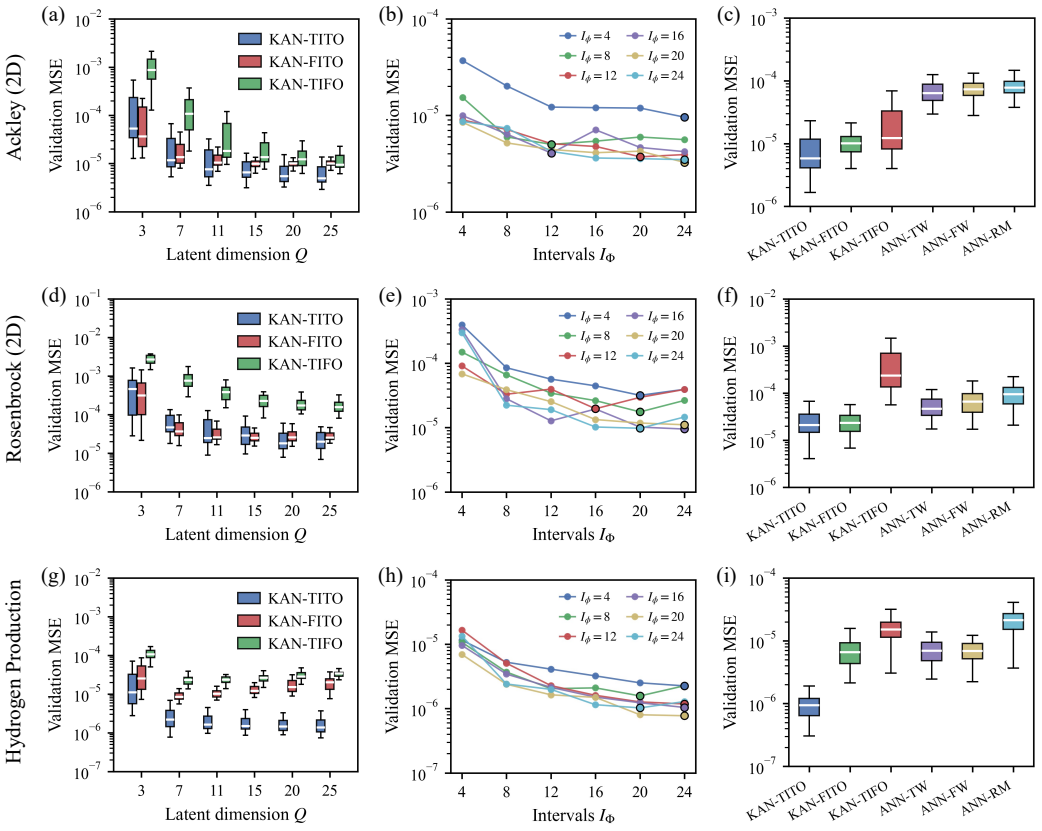


Figure 4: Predictive performance for the Ackley, Rosenbrock, and hydrogen production cases. The first column of (a), (d), and (g), shows the validation MSE of KANs across different latent dimensions Q, where each box summarises the mean five-fold cross-validation MSE obtained across the candidate intervals at a fixed Q. The second column of (b), (e), and (h), shows the mean five-fold cross-validation MSE of KAN-TITO under different combinations of I_ϕ and I_Φ for the determined latent dimension. The third column of (c), (f), and (i), compares the validation MSE of KANs and corresponding ANNs, where each box summarises the mean five-fold cross-validation MSE obtained across the considered hyperparameters. The rows correspond to the Ackley-2D function, the Rosenbrock-2D function, and the industrial hydrogen production case, respectively.

Together with the validation error distributions in Figure 4(c), (f), and (i), the corresponding test MSE statistics across all cases are summarised in Table 10. KAN-TITO achieves the lowest mean test MSE in all cases and maintains relatively low validation errors across the considered hyperparameters, indicating that its predictive advantage is not confined to a single architecture. KAN-FITO also exhibits comparatively stable performance, although with some loss of accuracy relative to KAN-TITO. In contrast, KAN-TIFO is more problem dependent, with substantially larger errors for Rosenbrock and Hartmann while remaining competitive for Ackley and hydrogen production. The ANN variants similarly exhibit varying sensitivity across problems and hyperparameters. When considered together with Table 9, these results indicate that the higher training cost of the CPWL-KAN models, particularly KAN-TITO, is accompanied by improved predictive accuracy under the adopted training procedure, whereas the lower training cost of the ANN models does not consistently translate into lower prediction error.

These comparisons should nevertheless be interpreted within the training protocol adopted in this work. Longer training, different early-stopping settings, or additional hyperparameter tuning could potentially improve the predictive performance of the ANN baselines and the other KAN variants. The reported errors therefore characterise the performance obtained under the adopted procedure rather than the best predictive performance of each architecture.

C.3 Global Optimisation Performance

C.3.1 Relaxation Strength

The complete relaxation statistics for the three analytical benchmarks are reported in Table 11. Across all cases, the joint SOS2 formulation yields smaller relaxation gaps than the corresponding separate big-M formulation for all three KAN variants, consistent with the observations in Section 4.2.1. For KAN-TITO, the mean relaxation gap under separate big-M is approximately 2.24, 3.64, and 5.83 times that under joint SOS2 for Ackley-2D, Rosenbrock-2D, and Hartmann-6D, respectively. The corresponding ratios are 5.80, 5.02, and 17.60 for KAN-FITO and 5.02, 3.04, and 5.87 for KAN-TIFO. The median and P_{95} statistics show similar trends, indicating that the differences persist across the evaluated domains rather than being driven by a small number of extreme instances.

The ANN baselines encoded using separate big-M generally exhibit larger relaxation gaps, with ANN-RM showing particularly large gaps across all three benchmarks. As each neuron accepted all the input from the previous layer, ignoring the dependence between inputs can exaggerate the neuron bounds and then the relaxation over networks. The trend is not uniform across all KAN variants: under the separate big-M encoding, KAN-FITO and KAN-TIFO can be comparable to or weaker than ANN-FW, whereas KAN-TITO maintains relatively tight relaxations across the considered cases. These results further show that relaxation quality depends strongly on the trained parametrisation and resulting network representation, rather than on model size alone.

Table 11: Summary statistics of the normalised relaxation gaps $\overline{y}_i - \underline{y}_i$ for the Ackley, Rosenbrock, and Hartmann cases. Median and P_{95} denote the median and 95th percentile, respectively. Ratio denotes the mean relaxation gap of the separate big-M formulation relative to the corresponding joint SOS2 formulation. Std. denotes the sample standard deviation.

Model	Formulation	Mean	Std.	Min.	Median	P_{95}	Max.	Ratio
Ackley-2D								
KAN-TITO	SOS2	0.736	0.159	0.183	0.758	0.953	0.983	–
	big-M	1.648	0.234	0.877	1.674	1.975	2.019	2.24
KAN-FITO	SOS2	1.715	0.400	0.261	1.765	2.247	2.306	–
	big-M	9.951	0.756	7.128	10.023	11.003	11.130	5.80
KAN-TIFO	SOS2	1.919	0.455	0.410	1.942	2.576	2.637	–
	big-M	9.637	0.458	8.046	9.691	10.271	10.340	5.02
ANN-TW	big-M	11.613	0.490	9.932	11.645	12.277	12.329	–
ANN-FW	big-M	13.292	0.470	11.810	13.309	13.992	14.068	–
ANN-RM	big-M	90.619	1.958	86.562	90.390	94.106	94.995	–
Rosenbrock-2D								
KAN-TITO	SOS2	1.094	0.322	0.024	1.176	1.466	1.487	–
	big-M	3.984	0.875	0.967	4.279	4.881	4.901	3.64
KAN-FITO	SOS2	2.739	0.659	0.211	2.864	3.556	3.621	–
	big-M	13.756	1.153	9.188	13.912	15.237	15.378	5.02
KAN-TIFO	SOS2	2.929	0.666	0.644	2.945	3.949	4.077	–
	big-M	8.890	0.782	6.011	8.973	10.005	10.177	3.04
ANN-TW	big-M	11.973	0.805	9.017	12.177	12.901	12.978	–
ANN-FW	big-M	13.808	0.877	10.808	13.909	15.004	15.111	–
ANN-RM	big-M	85.221	1.850	80.518	85.053	88.442	89.254	–
Hartmann-6D								
KAN-TITO	SOS2	2.463	0.264	1.425	2.491	2.866	3.009	–
	big-M	14.369	0.743	11.525	14.458	15.452	15.982	5.83
KAN-FITO	SOS2	2.939	0.264	1.913	2.958	3.324	3.467	–
	big-M	51.716	0.641	49.002	51.797	52.616	52.834	17.60
KAN-TIFO	SOS2	3.792	0.315	2.718	3.805	4.278	4.513	–
	big-M	22.276	0.551	20.520	22.329	23.108	23.445	5.87
ANN-TW	big-M	32.871	0.539	31.137	32.862	33.786	34.232	–
ANN-FW	big-M	32.076	0.589	30.419	32.059	33.108	33.785	–
ANN-RM	big-M	264.695	3.614	255.076	264.537	271.228	279.432	–

Table 12 reports the corresponding relaxation statistics for the three outputs of the hydrogen production case. Similar trends are observed across all outputs. For KAN-TITO, the mean relaxation gap under the separate big-M encoding is approximately 2.24–4.00 times that under the joint SOS2 encoding. The corresponding ratios increase to 22.65–27.72 for KAN-FITO and remain close to 12 for KAN-TIFO. Under the separate big-M encoding, KAN-TITO maintains markedly tighter relaxations than all ANN baselines across the three outputs, whereas KAN-FITO and KAN-TIFO are more comparable to ANN-TW and ANN-FW. ANN-RM consistently exhibits the largest relaxation gaps, consistent with its substantially larger network architecture. These results extend the trends observed for the analytical benchmarks to the hydrogen production case.

Table 12: Summary statistics of the normalised relaxation gaps $\overline{y}_o - \underline{y}_o$ for the three outputs in hydrogen production case. Median and P_{95} denote the median and 95th percentile, respectively. Ratio denotes the mean relaxation gap of the separate big-M formulation relative to the corresponding joint SOS2 formulation for each output. Std. denotes the sample standard deviation.

Model	Formulation	Output	Mean	Std.	Min.	Median	P_{95}	Max.	Ratio
KAN-TITO	SOS2	y_1	0.327	0.064	0.129	0.332	0.419	0.432	–
		y_2	0.401	0.067	0.176	0.414	0.492	0.514	–
		y_3	0.165	0.041	0.046	0.171	0.222	0.235	–
	big-M	y_1	0.988	0.124	0.605	0.999	1.180	1.217	3.03
		y_2	0.900	0.095	0.608	0.907	1.036	1.067	2.24
		y_3	0.658	0.084	0.405	0.671	0.787	0.812	4.00
KAN-FITO	SOS2	y_1	0.390	0.067	0.185	0.394	0.489	0.508	–
		y_2	0.424	0.086	0.148	0.432	0.543	0.580	–
		y_3	0.319	0.070	0.117	0.326	0.416	0.441	–
	big-M	y_1	10.807	0.374	9.678	10.834	11.334	11.472	27.72
		y_2	9.596	0.395	8.328	9.628	10.168	10.307	22.65
		y_3	8.598	0.486	7.137	8.619	9.303	9.498	26.97
KAN-TIFO	SOS2	y_1	0.729	0.138	0.261	0.741	0.930	0.995	–
		y_2	0.797	0.152	0.238	0.809	1.019	1.085	–
		y_3	0.724	0.176	0.193	0.741	0.971	1.057	–
	big-M	y_1	9.032	0.144	8.489	9.046	9.228	9.305	12.38
		y_2	9.526	0.186	8.725	9.567	9.766	9.830	11.96
		y_3	8.683	0.220	7.943	8.727	8.970	9.030	11.99
ANN-TW	big-M	y_1	9.648	0.321	8.895	9.617	10.217	10.497	–
		y_2	11.467	0.430	10.409	11.449	12.179	12.519	–
		y_3	13.193	0.415	12.228	13.149	13.907	14.240	–
ANN-FW	big-M	y_1	10.033	0.477	8.741	10.020	10.832	11.190	–
		y_2	9.210	0.444	7.955	9.201	9.952	10.328	–
		y_3	9.079	0.304	8.130	9.096	9.582	9.763	–
ANN-RM	big-M	y_1	115.161	2.027	110.366	114.916	118.653	120.410	–
		y_2	120.762	2.169	115.522	120.580	124.568	126.434	–
		y_3	137.264	2.405	131.549	136.937	141.399	143.574	–

C.3.2 Interval Pruning

Table 13 shows that interval pruning consistently reduces the effective CPWL representations across all three analytical benchmarks and independent training runs, although the extent of the reduction depends strongly on both the KAN parametrisation. A particularly consistent behaviour is observed for KAN-FITO, which retains most of the predefined intervals in the first ϕ layer, with reductions of only 13.58%, 10.83%, and 7.79% for Ackley, Rosenbrock, and Hartmann, respectively. In contrast, substantially stronger pruning occurs in the subsequent Φ layer, where the corresponding reductions increase to 59.25%, 53.11%, and 67.00%. This behaviour indicates that the FITO parametrisation tends to preserve a comparatively dense CPWL representation in the first layer while producing substantially more restricted intermediate domains for the second layer.

KAN-TIFO generally yields the most compact effective representations after pruning, particularly in the first layer, where the reductions range from approximately 35% to 49%. Its reductions in

second layer are also substantial, although they are not uniformly larger than those of KAN-FITO. KAN-TITO exhibits more moderate and balanced pruning across the two layers, with the Hartmann case showing nearly identical reductions in ϕ and Φ. Together, these results confirm that the different parametrisations lead to systematically different learned breakpoint distributions and effective representation sizes, consistent with the observations for the hydrogen production case.

Table 13: Interval statistics before and after equivalent pruning for the analytical benchmark problems across five independent training runs. After-pruning statistics are pooled across all edge functions and seeds, while total intervals are reported as the mean $\pm$ standard deviation of the seed-level totals.

Model	Layer	Before pruning	After pruning	Total intervals	Reduction (%)
Ackley-2D					
KAN-TITO	ϕ	12	8.87 ± 1.25	354.80 ± 6.46	26.08
	Φ	12	6.17 ± 2.28	123.40 ± 4.67	48.58
KAN-FITO	ϕ	12	10.37 ± 0.60	414.80 ± 2.95	13.58
	Φ	12	4.89 ± 1.55	97.80 ± 7.73	59.25
KAN-TIFO	ϕ	12	7.75 ± 2.14	310.00 ± 18.22	35.42
	Φ	12	5.05 ± 1.90	101.00 ± 12.19	57.92
Rosenbrock-2D					
KAN-TITO	ϕ	16	10.68 ± 2.21	320.40 ± 15.34	33.25
	Φ	12	7.51 ± 2.72	112.60 ± 8.91	37.44
KAN-FITO	ϕ	16	14.27 ± 0.61	428.00 ± 4.85	10.83
	Φ	12	5.63 ± 2.38	84.40 ± 4.56	53.11
KAN-TIFO	ϕ	16	9.95 ± 2.86	298.60 ± 23.20	37.79
	Φ	12	5.07 ± 2.02	76.00 ± 11.38	57.78
Hartmann-6D					
KAN-TITO	ϕ	24	13.88 ± 3.50	1249.40 ± 20.38	42.16
	Φ	20	11.77 ± 3.06	176.60 ± 13.65	41.13
KAN-FITO	ϕ	24	22.13 ± 1.04	1991.80 ± 9.15	7.79
	Φ	20	6.60 ± 2.15	99.00 ± 8.28	67.00
KAN-TIFO	ϕ	24	12.32 ± 4.34	1108.40 ± 53.52	48.69
	Φ	20	9.15 ± 2.79	137.20 ± 12.24	54.27

The corresponding runtime results in Figure 5 further confirm that the computational benefit of pruning is formulation and problem dependent. Under joint SOS2 encoding, pruning generally improves or preserves the solution performance, with more noticeable gains for some KAN-FITO and KAN-TIFO instances. In contrast, the before- and after-pruning results under separate big-M remain largely comparable across the three benchmarks. Together with the hydrogen production results in the main text, these experiments show that equivalent pruning consistently reduces the explicit CPWL representation under the joint encoding approach.

C.3.3 Computational Runtime

To evaluate computational efficiency and agreement between the optimal objective values of the original and surrogate problems, the absolute objective gap for model m and instance k as

$$\Delta f_m^{(k)} = \left| f_{\text{ref}}^{*(k)} - \hat{f}_m\left(\boldsymbol{x}_m^{*(k)}\right) \right|, \tag{C.1}$$

where $f_{\text{ref}}^{*(k)}$ denotes the reference objective value obtained by optimising the original analytical function or process model for instance k, and $\boldsymbol{x}_m^{*(k)}$ denotes the solution obtained by optimising the corresponding surrogate model $\hat{f}_m$. The original and surrogate problems use the same instance specifications. Thus, $\Delta f_m^{(k)}$ compares objective values evaluated at their respective optimisation solutions. It measures optimal value agreement rather than the solver optimality gap or the prediction error at a decision solution. The gap statistics are reported using the joint SOS2 encoding solutions for KAN models and the big-M solutions for ANN models.

Figure 6 presents the optimisation results for the analytical benchmarks, while Table 14 additionally reports the hydrogen production case. Across all KAN configurations, the joint SOS2 encoding

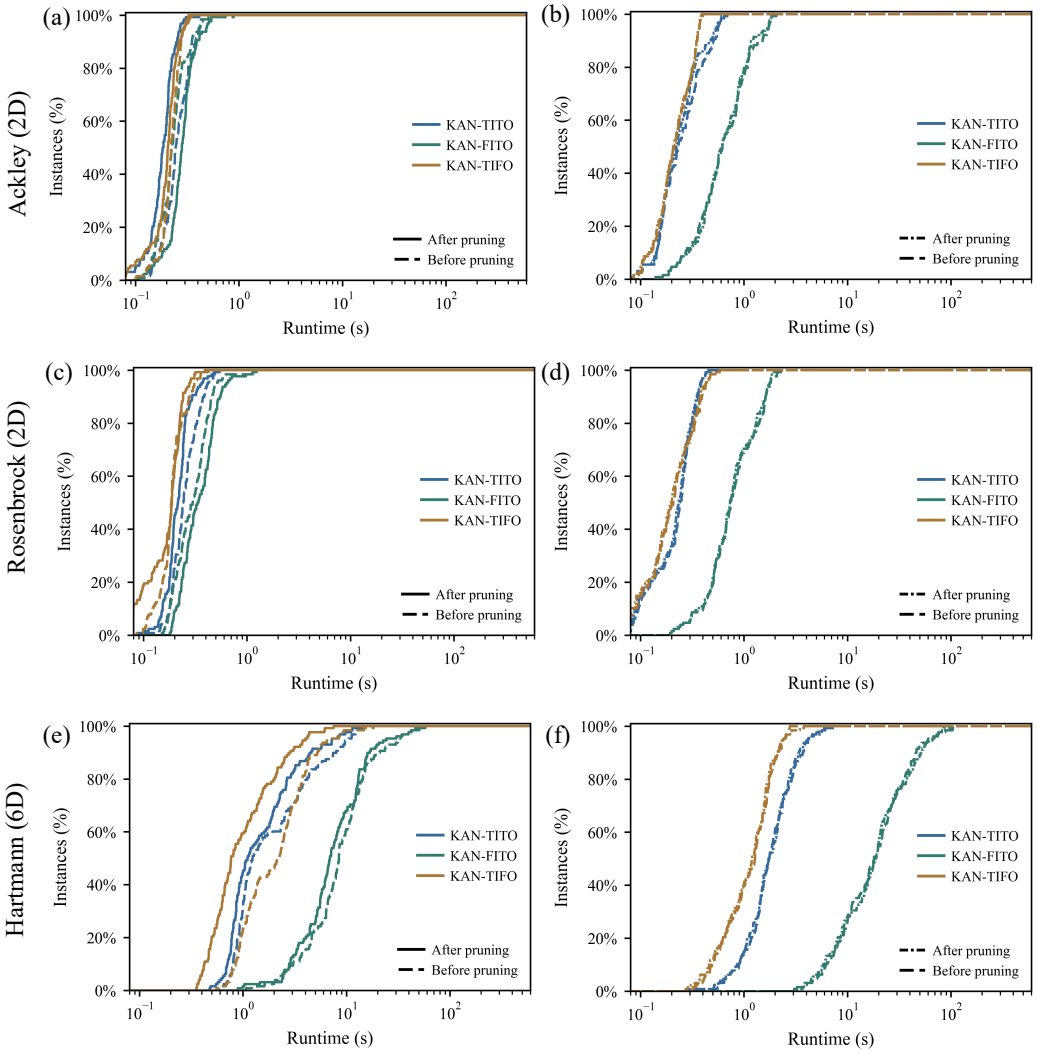


Figure 5: Effect of equivalent interval pruning on the computational performance of the KAN formulations for the analytical benchmark problems. Rows correspond to the Ackley, Rosenbrock, and Hartmann functions, respectively. Panels (a), (c), and (e) show the cumulative percentage of optimisation instances solved within a given runtime using the joint SOS2 encoding, while panels (b), (d), and (f) show the corresponding results using the separate big-M encoding.

achieves shorter median runtimes than the separate big-M encoding. The difference is particularly pronounced for KAN-FITO on Hartmann and for all three KAN variants in the hydrogen production case. This advantage is not uniform across the runtime distributions, however. For Hartmann, KAN-TITO and KAN-TIFO exhibit larger 95th% runtimes with the joint SOS2 encoding, indicating that its tighter relaxation and associated bound propagation do not necessarily translate into shorter runtimes for every instance. Among the KAN variants, KAN-TIFO achieves the shortest median runtime with the joint encoding for Rosenbrock, Hartmann, and hydrogen production, consistent with its more compact effective representation after pruning. The compact ANN models remain competitive on the 2D-benchmarks, with ANN-FW attaining the shortest median runtime for both Ackley and Rosenbrock. In contrast, ANN-RM requires substantially greater computational effort even on these cases, while none of its Hartmann or hydrogen production instances are solved within the 600 s time limit. The reported runtime quantiles are conditional on instances completed within this limit.

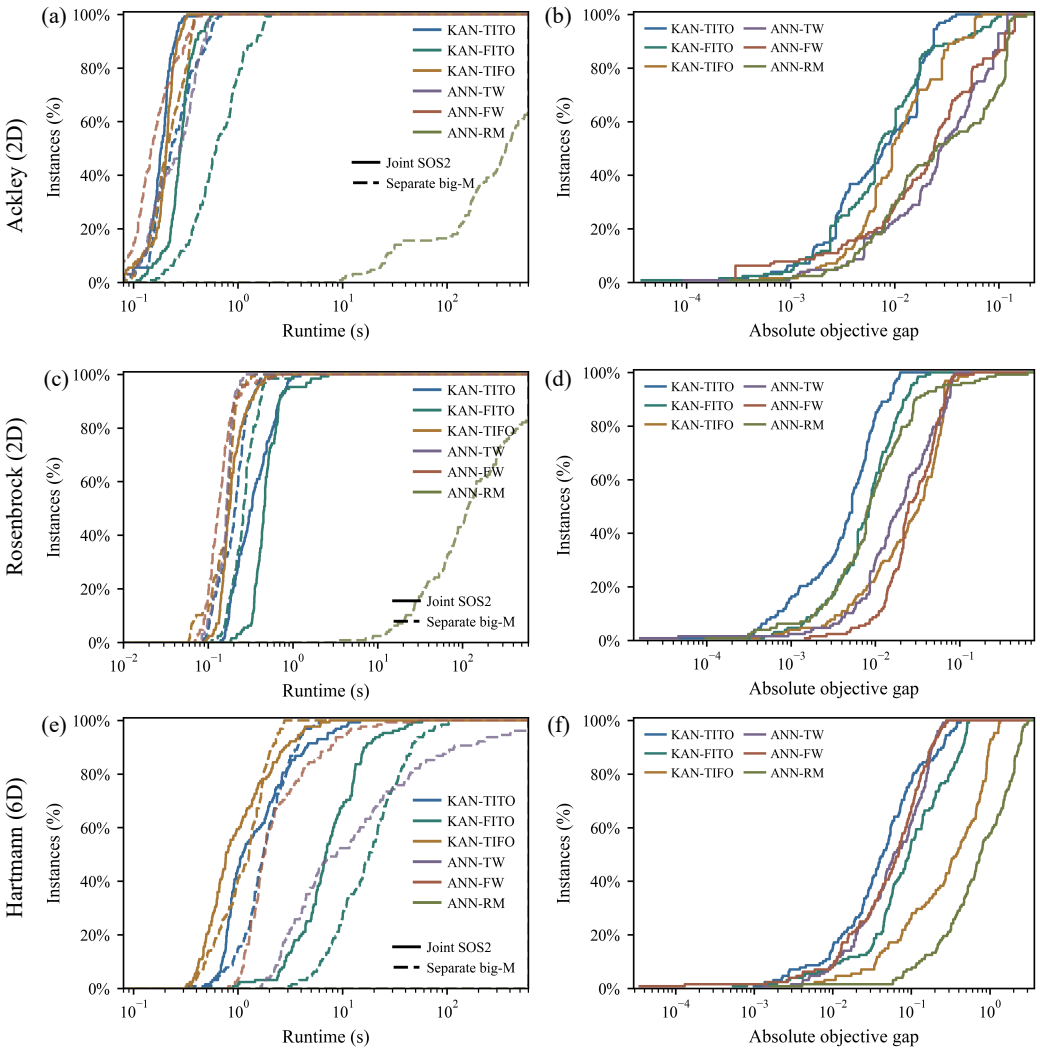


Figure 6: Optimisation performance for the analytical benchmark problems. Rows correspond to the Ackley, Rosenbrock, and Hartmann functions, respectively. Panels (a), (c), and (e) show the cumulative percentage of optimisation instances solved within a given runtime for the KAN and ANN formulations. Panels (b), (d), and (f) show the cumulative percentage of instances with an absolute objective gap below a given value, characterising the objective value agreement between the surrogate and reference optimisation problems.

The objective value gaps reveal a complementary trade-off between computational efficiency and agreement with the reference problems. KAN-TITO which has the highest accuracy, achieves the smallest median gap for Rosenbrock, Hartmann, and hydrogen production, with values of 0.00531, 0.0435, and 8.19×10^{-4}, respectively. For Ackley, KAN-FITO attains the smallest median gap, whereas KAN-TITO gives the smallest 95th% gap. KAN-TITO also achieves the smallest 95th% gaps for Rosenbrock and hydrogen production. Its advantage is less uniform for Hartmann: although its median gap remains the lowest, ANN-TW and ANN-FW attain smaller 95th% gaps, indicating better objective value agreement in the upper tail. KAN-TIFO generally provides faster optimisation but at the cost of larger objective gaps, most noticeably for Hartmann. ANN-RM achieves a comparatively small median gap for Rosenbrock, but only with substantially greater solution time. Taken together, these results indicate that KAN-TITO provides a consistent balance between computational performance and objective value agreement across the considered cases, whereas KAN-TIFO places

greater emphasis on optimisation speed and compact ANN models remain competitive for selected lower dimensional problems.

Table 14: Summary of optimisation performance across all cases. t_{50} and t_{95} denote the median and 95th-percentile runtimes, respectively, for instances solved within the 600 s time limit. Δf_{50} and Δf_{95} denote the median and 95th percentile of the absolute objective gaps of the surrogate models and the reference problems. For KAN models, the gap statistics are reported for joint SOS2 encoding; for ANN models, they are reported for the big-M formulation.

	SOS2 runtime (s)		big-M runtime (s)		Objective gap	
Model	t_{50}	t_{95}	t_{50}	t_{95}	Δf_{50}	Δf_{95}
Ackley-2D						
KAN-TITO	0.185	0.268	0.234	0.561	0.00782	0.0248
KAN-FITO	0.281	0.482	0.620	1.729	0.00700	0.0689
KAN-TIFO	0.205	0.296	0.207	0.371	0.00943	0.0589
ANN-TW	–	–	0.275	0.504	0.0279	0.121
ANN-FW	–	–	0.148	0.360	0.0239	0.142
ANN-RM	–	–	180.412	524.439	0.0797	0.126
Rosenbrock-2D						
KAN-TITO	0.215	0.360	0.241	0.382	0.00531	0.0165
KAN-FITO	0.336	0.630	0.743	1.753	0.00858	0.0265
KAN-TIFO	0.188	0.277	0.208	0.428	0.0329	0.0660
ANN-TW	–	–	0.163	0.374	0.0199	0.0784
ANN-FW	–	–	0.106	0.192	0.0247	0.0726
ANN-RM	–	–	84.033	393.742	0.00773	0.0281
Hartmann-6D						
KAN-TITO	1.107	7.226	1.749	4.106	0.0435	0.298
KAN-FITO	6.910	23.743	18.033	59.856	0.0930	0.496
KAN-TIFO	0.769	3.914	1.279	2.390	0.379	1.131
ANN-TW	–	–	7.061	186.265	0.0577	0.215
ANN-FW	–	–	1.794	10.904	0.0691	0.223
ANN-RM	–	–	–	–	–	–
Hydrogen production						
KAN-TITO	1.011	3.170	4.045	5.674	0.000819	0.00280
KAN-FITO	2.005	5.007	6.364	15.219	0.00287	0.00938
KAN-TIFO	0.389	0.536	1.754	3.050	0.00331	0.0116
ANN-TW	–	–	22.752	60.842	0.00371	0.00897
ANN-FW	–	–	6.154	11.645	0.00251	0.00921
ANN-RM	–	–	–	–	–	–

C.3.4 Solver Comparison

The comparison of solvers in Figure 7 shows that the computational performance varies substantially across different optimisation solvers, with the differences becoming more pronounced for the higher dimensional Hartmann and hydrogen production cases. Despite these variations, the general trends among the models remain broadly consistent. The KAN formulations retain competitive runtimes across the considered solvers, whereas large ANN-RM is substantially more expensive and frequently approaches or reaches the 600 s time limit for the more difficult cases. KAN-TITO and KAN-TIFO generally exhibit comparatively robust computational performance, while KAN-FITO shows greater sensitivity to the selected encoding and solver as it preserves more intervals.

The relative performance of joint SOS2 and separate big-M also depends on the solvers. Although joint SOS2 provides a theoretically no weaker continuous relaxation, as established in Section 3.2.3, this advantage does not translate into uniformly shorter runtimes for every solver and problem. SOS2 is particularly favourable for several solvers such as Gurobi and COPT, whereas big-M can remain competitive, or occasionally faster, under other solvers. These differences reflect the individual treatment with respect to joint SOS2 encoding, including presolve, branching, and internal reformulations within the solvers, can result in different optimisation performance. Overall, the results

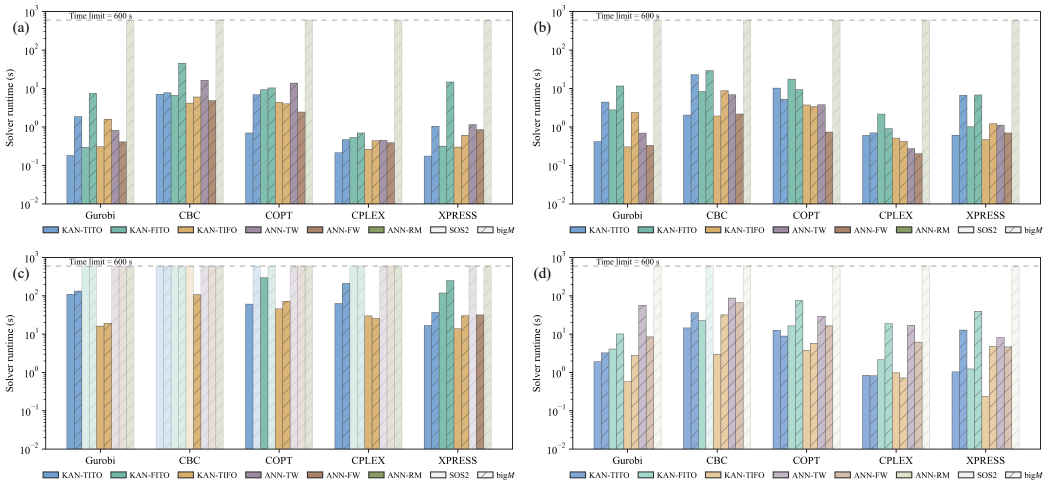


Figure 7: Computational performance of the KAN and ANN formulations across different optimisation solvers on the nominal problems. Panels (a)–(d) report the solver runtimes for Ackley-2D, Rosenbrock-2D, Hartmann-6D and hydrogen case, using Gurobi, CBC, COPT, CPLEX, and Xpress. Colours distinguish KAN-TITO, KAN-FITO, KAN-TIFO, ANN-TW, ANN-FW, and ANN-RM, while hatch patterns distinguish the joint SOS2 and seperate big-M encodings. The horizontal dashed line indicates the 600 s time limit.

indicate that the relaxation advantage of joint SOS2 is independent, whereas its practical runtime benefit depends additionally on the optimisation solver and problem structure.

C.3.5 Scaling Performance

Dimension scaling is tested on the Ackley and Rosenbrock benchmarks from 4D to 12D. No additional architecture search or hyperparameter tuning is performed here; instead, the architectures selected for the corresponding 2D cases are directly reused with the input dimension adjusted accordingly. The results therefore assess the behaviour of the original surrogate configurations under increasing dimensionality rather than the best achievable performance at each dimension.

As shown in Table 15, KAN-TITO and KAN-FITO maintain relatively stable predictive errors on Ackley and outperform the ANN baselines across all considered dimensions. The Rosenbrock results are more problem dependent: the compact ANNs are more accurate at 4D and 8D, while KAN-TITO and KAN-FITO become comparable to or more accurate than them at 12D. Thus, the prediction over dimensionality depends on the underlying problem.

Table 16 shows clearer differences in optimisation scalability when both model generation and solver time are considered. KAN-TITO and KAN-TIFO retain strong computational performance as the dimension increases, with most instances solved within the time limit across both benchmarks. In particular, the joint SOS2 encoding has a substantially lower generation cost than the separate big-M encoding in the current implementation. For the KAN models, the generation time of separate big-M increases from approximately 30–45 s at 4D to around 90–98 s at 12D, whereas the corresponding joint SOS2 models are generally generated within a few seconds. Consequently, although separate big-M can achieve shorter solver times in several cases, particularly for KAN-TITO and KAN-TIFO, this advantage can be offset by its substantially larger generation costs.

KAN-FITO is more sensitive to the encoding strategy. For higher dimensional Rosenbrock, the joint SOS2 encoding becomes difficult to solve despite its low generation cost, whereas separate big-M solves all instances but incurs considerably larger preprocessing time. The ANN formulations exhibit relatively small generation costs due to a more compact structure, but their solver performance deteriorates more substantially with dimension, particularly for Ackley-12D. Overall, these results show that computational scalability depends on both model generation and subsequent solution effort. The stronger relaxation of the joint SOS2 encoding therefore does not guarantee uniformly

Table 15: Predictive performance with scaling input dimension on the Ackley and Rosenbrock benchmarks. Values are reported as mean ± standard deviation of the raw test MSE.

Model	4D	8D	12D
Ackley			
KAN-TITO	0.0338 ± 0.0082	0.0349 ± 0.0148	0.0334 ± 0.0066
KAN-FITO	0.0320 ± 0.0036	0.0365 ± 0.0140	0.0267 ± 0.0097
KAN-TIFO	0.0557 ± 0.0077	0.0669 ± 0.0044	0.0447 ± 0.0022
ANN-TW	0.0629 ± 0.0118	0.1033 ± 0.0090	0.1099 ± 0.0074
ANN-FW	0.0596 ± 0.0073	0.1042 ± 0.0095	0.1127 ± 0.0083
ANN-RM	0.0896 ± 0.0570	0.1579 ± 0.0060	0.1375 ± 0.0054
Rosenbrock			
KAN-TITO	0.1041 ± 0.0175	0.0982 ± 0.0072	0.0856 ± 0.0042
KAN-FITO	0.1175 ± 0.0137	0.1096 ± 0.0078	0.0943 ± 0.0050
KAN-TIFO	0.1480 ± 0.0199	0.1243 ± 0.0145	0.1189 ± 0.0158
ANN-TW	0.0722 ± 0.0113	0.0797 ± 0.0074	0.0952 ± 0.0066
ANN-FW	0.0844 ± 0.0119	0.0801 ± 0.0086	0.1026 ± 0.0059
ANN-RM	0.1014 ± 0.0173	0.1001 ± 0.0068	0.1114 ± 0.0085

shorter solver times, but its considerably lower generation cost can make it more favourable in terms of total computational effort for several KAN configurations.

Table 16: Optimisation scaling performance on the Ackley and Rosenbrock benchmarks. Each dimension reports solved instances (%) on the first line and mean generation time + mean solver time (s) on the second line.

Model	Encoding	4D	8D	12D
Ackley				
		100.0%	100.0%	100.0%
KAN-TITO	Joint SOS2	$1.432 + 0.477$	$2.122 + 1.383$	$4.043 + 2.244$
		100.0%	100.0%	100.0%
	Separate big-M	$45.231 + 0.504$	$69.498 + 1.209$	$97.803 + 1.285$
		100.0%	78.9%	89.8%
KAN-FITO	Joint SOS2	$1.814 + 2.660$	$2.061 + 179.758$	$2.783 + 135.979$
		100.0%	100.0%	100.0%
	Separate big-M	$38.614 + 2.869$	$63.111 + 19.145$	$91.295 + 10.430$
		100.0%	100.0%	100.0%
KAN-TIFO	Joint SOS2	$1.384 + 0.296$	$2.084 + 0.753$	$2.708 + 1.844$
		100.0%	100.0%	100.0%
	Separate big-M	$33.128 + 0.174$	$64.990 + 0.350$	$91.589 + 0.539$
		100.0%	86.7%	2.3%
ANN-TW	big-M	$2.050 + 1.409$	$3.124 + 178.648$	$3.958 + 593.156$
		100.0%	100.0%	26.6%
ANN-FW	big-M	$1.970 + 0.823$	$2.623 + 25.384$	$3.306 + 512.939$
ANN-RM	big-M	–	–	–
Rosenbrock				
		100.0%	100.0%	82.0%
KAN-TITO	Joint SOS2	$1.174 + 0.940$	$1.702 + 7.190$	$2.177 + 148.392$
		100.0%	100.0%	100.0%
	Separate big-M	$32.009 + 0.696$	$59.939 + 1.519$	$89.773 + 3.367$
		100.0%	32.0%	0.0%
KAN-FITO	Joint SOS2	$1.161 + 4.111$	$1.680 + 464.631$	$2.236 + 600.000$
		100.0%	100.0%	100.0%
	Separate big-M	$31.984 + 3.035$	$59.823 + 32.037$	$90.592 + 99.828$
		100.0%	100.0%	100.0%
KAN-TIFO	Joint SOS2	$1.159 + 0.336$	$1.675 + 1.169$	$2.168 + 6.720$
		100.0%	100.0%	100.0%
	Separate big-M	$32.129 + 0.284$	$59.840 + 0.739$	$90.635 + 1.571$
		100.0%	97.7%	71.9%
ANN-TW	big-M	$2.134 + 11.641$	$3.281 + 42.736$	$4.252 + 255.517$
		100.0%	100.0%	95.3%
ANN-FW	big-M	$2.045 + 0.867$	$2.723 + 4.232$	$3.430 + 66.013$
ANN-RM	big-M	–	–	–