Sequential Complex-Valued Kolmogorov-Arnold Networks*

Uladzislau Yorsh[1,*], Matthias Wolff[2,*], Martin Holeňa[3], Xiaoyi Jiang[2] and Ondřej Bojar[4]

[1]Faculty of Mathematics and Physics, Charles University in Prague, 121 16 Prague, Czech Republic
[2]Department of Computer Science, University of Münster, 48149 Münster, Germany
[3]Institute of Computer Science, Academy of Sciences of the Czech Republic, 182 07 Prague, Czech Republic
[4]Institute of Formal and Applied Linguistics, Charles University in Prague, 118 00 Prague, Czech Republic

Abstract

Complex-Valued Kolmogorov-Arnold Networks (CVKANs) extend the original Kolmogorov-Arnold Network (KAN) model to the complex domain. Along with the new paradigm of building neural networks, these models offer natural interpretability, parameter efficiency and competitive performance on a multitude of tasks, making them a powerful building block for other models.

In this work, we transfer the concepts and benefits of KANs and, more specifically, CVKANs into the realm of sequence processing neural networks. Along with experiments inspired by Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) variants, we also demonstrate a Linear Recurrent Neural Network (LRNN) model exploiting both flexibility of (CV)KANs and associativity of the LRNN framework.

Keywords

Sequential Networks, Long Short-Term Memory, Recurrent Neural Networks, Kolmogorov-Arnold Networks, Deep Learning

1. Introduction

Kolmogorov-Arnold Networks (KANs) [1] have been shown to be a promising new approach, especially for symbolic function fitting tasks [2] and scientific discovery, like solving physics-informed Partial Differential Equations (PDEs) [3]. They can serve as an interesting alternative to Multilayer Perceptrons (MLPs) in some applications and offer increased accuracy for scientific tasks or symbolic regression [4]. In other domains KANs are still outperformed by classical MLPs in terms of prediction accuracy [2] as well as computational speed [4], but their intrinsic interpretability is still highly desired in critical applications, such as medical and engineering applications. KANs have been successfully applied in convolutional networks [5] and image processing [6], as well as in Graph Neural Networks (GNNs) [7, 8] and transformer architectures [9].

In the field of machine learning, there is an abundance of sequence-processing tasks coming from natural language processing [10], time-series modeling and other domains, which among others include also computer vision [11]. However, we have identified a research gap in the related work, and see a field of designing a recurrent mechanism involving KANs as underexplored. Our work aims to combine the flexibility of KANs with capabilities and benefits of sequential models, namely Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) and Linear Recurrent Neural Networks (LRNNs), to run in linear time with respect to the input length and produce meaningful representations. Since non-linear transforms in some regard are still a heart of these methods, we believe that by carefully designing a KAN module it can be used to produce a comparable model combining both the benefits of KAN and the original architecture.

ITAT 2026: 14th International Workshop on Computational Intelligence and Data Mining, September 25–29, 2026, Vršatec, SK
*Corresponding author.
vlad.yorsh@matfyz.cuni.cz (U. Yorsh); matthias.wolff@uni-muenster.de (M. Wolff)
https://vladyorsh.github.io/ (U. Yorsh); https://www.uni-muenster.de/PRIA/en/personen/wolff.shtml (M. Wolff)
0009-0003-2361-8073 (U. Yorsh); 0009-0006-1432-6265 (M. Wolff)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-09

CEUR-WS.org/Vol-4323/paper4.pdf

Contribution. Our contribution is threefold: first, we introduce a complex-valued variant of a gated recurrent unit driven by Complex-Valued Kolmogorov-Arnold Networks. Second, we introduce a variant of a Linear Recurrent Neural Network, combining the flexibility and parameter efficiency of KANs with the speed of LRNNs. We show on two synthetic tasks, namely Selective Copying and A5 State Tracking, that KAN-powered models can reach or even surpass the performance of the baseline models. And finally, we provide fast JAX implementations of all of these models for further experiments.

2. Related Work

Kolmogorov-Arnold Networks While classical MLPs rely on the Universal Approximation Theorem (UAT) [12], KANs are backed by the Kolmogorov-Arnold Representation Theorem (KAT) [13], which states that any multivariate continuous function can be represented by a finite composition of continuous univariate functions. Since these univariate functions may be highly non-smooth and thus hard to learn or represent, the KAT has been deemed to be of little to no use in practice for a long time [14]. Liu et al. [1] have then relaxed the shape restrictions of KAT to arbitrary network widths and depths to increase the smoothness of the inner functions and to make the networks better trainable in practice.

KANs have replaced the weight matrices of MLPs with learnable activation functions. There are many different ways to represent and learn the activation functions. The original KAN architecture by Liu et al. makes use of splines [1], which are very flexible but more difficult to train and optimize, especially for parallelization on Graphics Processing Units (GPUs). There have been attempts to make the fitting of splines more efficient [15] or replace them with a weighted composition of Radial Basis Functions (RBFs) on a fixed grid [16]. Other alternatives for the bases include Chebyshev polynomials [17] or wavelets [18]. Noorizadegan et al. [19] give a detailed overview of different basis functions and their respective advantages and disadvantages for KAN architectures.

In a KAN, each learnable activation function on every edge contains many parameters for representing this function, and therefore the whole KAN can be more compact with fewer nodes and edges while still containing the same number of parameters and thus offering a similar performance as a much larger MLP in terms of network width and depth. Additionally, each learnable activation function in the network is univariate and therefore easier to understand for a human than some weight matrix in an MLP. Both of these properties together allow KANs to be inherently interpretable by plotting the compact network and inspecting individual edges for functional correlations between input features and output decisions or by tracing the flow of information inside the network to understand the internal workings of the model.

Complex and Hyper-Complex KANs Wolff et al. have extended the RBF-based approach of [16] first into the complex domain while maintaining the intrinsic interpretability of KANs with their Complex-Valued Kolmogorov-Arnold Network (CVKAN) [20] and later also into arbitrarily high-dimensional Clifford Algebras (CAs), like Quaternions or other hyper-complex domains, with a quasi-random Sobol grid approach for mitigating the exponential scaling of grid points in higher dimensions [21].

Linear Recurrent Networks. Before the emergence of Transformers, Recurrent Neural Networks and Long Short-Term Memory models were a go-to approach for sequence processing tasks such as language modeling, audio processing, spatiotemporal analysis and others. Despite still being used in domains such as time series modeling, the widespread use of these models has faded in favor of the Transformer architecture, which demonstrated superior scalability.

However, in recent years the interest in recurrent models has sparked again, with the key direction of research being RNNs with linear transitions. The recent works [22, 23] show that with a careful parametrization these models remain stable and do not suffer from gradient vanishing or explosion, while the property of square matrices being diagonalizable almost everywhere [24] and a leverage of complex states [22, 23] allowed to approach Transformers in terms of performance at the significantly lesser computational cost. This motivates our interest in CVKANs [20], since a neural network layer

designed specifically to represent transforms in the complex field could be more suitable for our purposes and can be potentially applied on states directly.

Combining KANs and Recurrent Models. Alongside straightforward approaches like Kolmogorov-Arnold Transformer [9], that replace elementwise feedforward layers, there were attempts to leverage KANs directly in sequence processing layers. TKAN [25], for example, uses a KAN to modulate an output gate of an LSTM. A more interesting way to integrate a KAN was demonstrated by HiPPO-KAN [26], where an input sequence is encoded into coefficients of some orthogonal polynomial basis, a KAN is applied to these coefficients and then is projected back into the feature space. This is not a traditional causal prediction, which is a main concern of our present work, but it demonstrates an alternative way to leverage CVKANs, since the operators projecting an input to a space of orthogonal polynomial basis coefficients often enjoy low-rank decompositions involving complex diagonal matrices [23].

3. Models

3.1. KAN and CVKAN Layers

In the present work we consider KAN models following [16, 20] and representing their scalar functions as RBF superpositions. Assume an input vector $\mathbf{x} \in \mathcal{T}^m$, where $\mathcal{T}$ is either a field of real numbers $\mathcal{R}$ or complex numbers $\mathcal{C}$; a (CV)KAN layer can be represented as an $n \times m$ matrix $\Phi(\cdot)$ of scalar $\mathcal{T} \to \mathcal{T}$ functions. An application of this matrix can be described as a transform $\Phi : \mathcal{T}^m \to \mathcal{T}^n$:

$$[\Phi(\mathbf{x})]_i = \Sigma_{j=0}^{m} [\Phi_i(\mathbf{x})]_j \tag{1}$$

where $\Phi_i(\cdot)$ is an elementwise application of m scalar functions in the i^{th} row of the matrix $\Phi(\cdot)$, and $[\cdot]_k$ is an operation of taking a k^{th} element of a vector. The individual scalar functions $\phi : \mathcal{T} \to \mathcal{T}$, where $\phi = \Phi_{ij}$, are represented as:

$$\phi(z) = \Sigma_{g \in G} [w_g \, \text{RBF}_\rho(z - g)] \tag{2}$$

where G is a predefined grid and $w_g \in \mathcal{T}$ is a corresponding weight; we take a 2D grid in the complex plane for complex-valued layers and a 1D grid on the real line for real-valued ones.

Additionally, we wrap the layer into a block containing a parallel Feedforward Network (FFN) branch combined with a Sigmoid Linear Unit (SiLU) or complex-valued SiLU [20] activation function:

$$\text{KAN}(\mathbf{x}) = \Phi(\mathbf{x}) + \text{FFN}(\text{SiLU}(\mathbf{x})) \tag{3}$$

$$\text{CVKAN}(\mathbf{z}) = \Phi(\mathbf{z}) + \text{FFN}(\text{ComplexSiLU}(\mathbf{z})) \tag{4}$$

and optionally pass the output through the corresponding variant of a normalization layer.

3.2. KAN-Powered RNNs

An RNN layer produces a new state through a linear transform and a subsequent nonlinearity as

$$\text{RNN}(\mathbf{x}_t, \mathbf{h}_{t-1}) = \sigma(W[\mathbf{x}_t; \mathbf{h}_{t-1}]) \tag{5}$$

where a standard choice for $\sigma(\cdot)$ is tanh. We introduce a KAN-based recurrent layer by using a KAN layer to map $[\mathbf{x}_t; \mathbf{h}_{t-1}]$ to $\mathbf{h}_t$:

$$\text{KAN-RNN}(\mathbf{x}_t, \mathbf{h}_{t-1}) = \text{KAN}([\mathbf{x}_t; \mathbf{h}_{t-1}]) \tag{6}$$

where $[\cdot;\cdot]$ is concatenation. The complex-valued variant is defined analogously.

We base the complex-valued LSTM variant on the TKAN [25]. The TKAN layer itself is based on LSTM; however, the output gate is modulated by a recurrent unit analogous to the KAN-RNN unit

described above. Our contribution is adapting the gating scheme of LSTM and TKAN for complex values.

$$x_h = [x_t; h_{t-1}] \tag{7}$$
$$f_t = \tilde{\sigma}_f(W_f x_h + b_f + 2.0) \tag{8}$$
$$u_t = \tilde{\sigma}_u(W_u x_h + b_u - 1.0) \tag{9}$$
$$g_t = \tilde{\tau}(W_g x_h + b_g) \tag{10}$$
$$k_u = \text{CVKAN}(W_k[x_t; k_{t-1}] + b_k) \tag{11}$$
$$k_t = (1 - \sigma(k_g)) \odot k_{t-1} + \sigma(k_g) \odot k_u \tag{12}$$
$$o_t = \tilde{\sigma}_o(W_o k_u + b_o) \tag{13}$$
$$c_t = f_t \odot c_{t-1} + g_t \odot u_t \tag{14}$$
$$h_t = \tilde{\tau}(c_t) \odot o_t \tag{15}$$

where σ is a sigmoid function, k_g is a learnable gating parameter and weights W and b are complex.

This sophisticated unit replicates LSTM, with two key distinctions: it retains the KAN-modulated output gate o_t present in TKAN, and adapts nonlinear functions for the complex-valued setting. $\tilde{\sigma}$ is a parameterized function defined as $\tilde{\sigma}(z) = \sigma(a \cdot \text{Re}(z) + b \cdot \text{Im}(z) + c)$, where a, b and c are learnable real parameters. $\tilde{\tau}$ is another generalization of a sigmoidal function defined as $\tilde{\tau}(z) = \tanh(|z|)\frac{z}{|z|}$.

While most models are using real-valued weights and states, complex values are interesting from the sequence modeling point of view because they decouple amplitude and phase; for example, a scalar linear time-invariant continuous state space model impulse response is $Ce^{tA}B$, where $A \in \mathbb{C}^{d\times d}, B \in \mathbb{C}^{1\times d}, C \in \mathbb{C}^{d\times 1}$. For the case where A is diagonal and stable (real part of the spectrum is negative) this impulse response function is essentially a sum of damped oscillations, and limiting A to be real does not allow to capture the oscillatory behavior since it is given by the phase.

3.3. KAN-Powered Fixed-Point RNNs

One of the questions of this work is how to combine the flexibility and parameter-efficiency of KANs with the recent advances in LRNNs, and how to retain an ability to compute them in parallel along the time dimension using associative scan. A naïve drop-in replacement of a KAN transform instead of a linear transition breaks associativity, so an alternative is needed. We have found the answer in the framework of *Fixed-Point Recurrent Neural Networks (FPRNNs)* [27].

Assume an LRNN with a dense transition matrix $\mathbf{A}_t$ written as a product of an invertible dense matrix $\mathbf{Q}^{-1}$ and a diagonal Λ_t:

$$\mathbf{h}_t = \mathbf{A}_t\mathbf{h}_{t-1} + \mathbf{B}_t\mathbf{x}_t = \mathbf{Q}_t^{-1}\Lambda_t\mathbf{h}_{t-1} + \mathbf{B}_t\mathbf{x}_t \tag{16}$$

FPRNNs originate from LRNNs; however, instead of explicitly computing the state by applying a transform they define the hidden state implicitly as a solution of the equilibrium equation:

$$\mathbf{h}_t^* = \Lambda_t\mathbf{h}_{t-1}^* + \mathbf{Q}_t\mathbf{B}_t\mathbf{x}_t + (\mathbf{I} - \mathbf{Q}_t)\mathbf{h}_t^* \tag{17}$$

where $\mathbf{h}_t^*$ is a fixed point of the transform $\mathbf{F}(\mathbf{h}^*, \mathbf{x}) : \mathscr{R}^{L\times d} \times \mathscr{R}^{L\times d} \to \mathscr{R}^{L\times d}$. By Thm. 3.1 in [27], whenever $\|\Lambda\|_2 < 1$ and $\|\mathbf{I} - \mathbf{Q}\|_2 < 1$, the transform $\mathbf{F}(\cdot, \mathbf{x}_t)$ becomes a contraction, and by the Banach contraction theorem the unique fixed point exists.

The main benefit of such formulation is that it decomposes the dense and diagonal component of the recurrence, in a way that allows to compute the recurrence across time steps using the diagonal transform Λ_t rather than the full transition $\mathbf{Q}_t^{-1}\Lambda_t$. The algorithm to compute the hidden state representation consists of interleaving *channel mixing* steps (elementwise multiplication by $\mathbf{Q}_t$ and $\mathbf{I} - \mathbf{Q}_t$) and *sequence mixing* steps (computing the recurrence in Λ_t). Denoting a fixed-point iteration step as l, the update equation becomes [27]:

$$\mathbf{h}_t^l = \Lambda_t^l \odot \mathbf{h}_{t-1}^l + (\mathbf{I} - \Lambda_t^l) \odot (\mathbf{Q}_t^l \mathbf{B}_t^l \mathbf{x}_t + (\mathbf{I} - \mathbf{Q}_t^l)\mathbf{h}_t^{l-1}) \tag{18}$$

The existence of a fixed point for the whole trajectory follows from its existence for the first element of the sequence and causality. Empirically we observe that stopping an iteration when $|\mathbf{h}_{t-1} - \mathbf{h}_t|/|\mathbf{h}_{t-1}| < 0.1$ is enough for the model to train.

When implementing our experiments, we fix the $\Lambda_t^l = \Lambda_t$ across the channel iteration and instead of expecting all state values to converge we stop iteration after 90% does. For complex-state models we also assume the complex conjugacy of state vectors and keep only half of feature channels, which keeps the effective state size in real values across experiments the same.

Another interesting property of the model is that under certain conditions (the Jacobians $\frac{\partial \mathbf{F}}{\partial \mathbf{x}}(\mathbf{h}, \mathbf{x})$ and $\frac{\partial \mathbf{F}}{\partial \mathbf{h}}(\mathbf{h}, \mathbf{x})$ must be equal) its optimization does not require backpropagation through channel mixing iteration, and among all values of $\mathbf{h}$ only requires differentiating at the fixed point. However, we observe that ignoring these conditions and imposing additional dependencies on parameters (such as parameterizing $\mathbf{Q}$ with both $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$) benefits performance and still produces gradients in the direction of the loss decrease.

The fixed-point formulation allows us to use nonlinear models such as KAN to predict a channel mixer $\mathbf{Q}_t^l$. From Equation 18 it could be seen that a channel mixing step (computing the right bracket in Equation 18) takes a constant amount of time per position t, while a sequence mixing step (iteration over timesteps t) could be carried in linear time and parallelized using associative scans since it is a linear state space model with a diagonal transition matrix.

KAN-Powered Channel Mixer. One of the crucial elements of the FPRNN is the channel mixer. It was shown in [27] that parametrization of the channel mixer greatly affects the final performance, and can alone be a deciding factor whether the model will generalize to other input lengths. We leverage that and parameterize the channel mixer with a KAN layer, altering the key part of the architecture without breaking its associativity in sequence mixing.

4. Experiments

4.1. Selective Copying

A selective copying task [28] is a task of, given an input sequence of length L, to produce a sequence of $M << L$ elements present in the sequence which are not padding. This synthetic task was used in the Mamba work [28] to demonstrate its distinctive capabilities from the previous generation of LRNNs in terms of memory retaining ability; we use this task as a playground to demonstrate that complex-valued recurrent units powered by KANs can work. We train single-layer models and along with token accuracy we track sequence accuracy, i.e. a proportion of sequences where all predicted tokens are correct.

Model	Width	Grid	Accuracy
RNN	32	-	30.3
LSTM	32	-	100.0
CVKAN-RNN	32	16	60.0
CVKAN-LSTM	32	16	100.0

Table 1
Test sequence accuracy of the considered models on the selective copying problem. We fix input length 32, model width 32, symbols to copy to 5 and alphabet size to 8.

Along with fixed-width comparison, we conduct experiments in Table 2 and Table 3 for two larger input configurations, and instead of controlling model width we control the number of real parameters. For this setup we have slightly changed the (CV)KAN-RNN units and instead of concatenating a state and an input as in Equation 6 we sum them.

Model	Width	Grid	Parameters	Token Accuracy	Sequence Accuracy
RNN	64	-	10320	13.69 ±3.45	0
KAN-RNN	48	4	10912	7.14 ±0.01	0
CVKAN-RNN	34	2	11100	21.59 ±10.02	0
LSTM	32	-	9360	71.32 ±10.84	19.64 ±15.20
KAN-LSTM (TKAN)	27	4	10573	76.77 ±19.47	39.23 ±32.41
CVKAN-LSTM	14	4	10824	94.80 ±3.15	78.14 ±12.32

Table 2
Selective Copying task, but now we fix the number of parameters near 10K. 128 input tokens, 5 symbols to copy, alphabet size 16. We run 7-10 experiments per parameter setup, and ± denotes standard deviation.

Model	Width	Grid	Parameters	Token Accuracy	Sequence Accuracy
RNN	224	-	107760	7.27 ±0.17	0
KAN-RNN	156	4	102820	7.15 ±0.02	0
CVKAN-RNN	112	2	106416	7.59 ±0.83	0
LSTM	112	-	104400	85.53 ±6.44	24.97 ±13.48
KAN-LSTM (TKAN)	84	4	95104	45.77 ±21.25	0.11 ±0.15
CVKAN-LSTM	44	4	99984	99.10 ±1.32	92.60 ±10.26

Table 3
Selective Copying task with number of parameters fixed around 100K. 256 input tokens, 10 symbols to copy, alphabet size 16. We run 7-10 experiments per parameter setup, and ± denotes standard deviation.

4.2. A5 Monoid Problem

A monoid problem is a task of learning a sequence-to-sequence transform, where inputs are elements of a given monoid and an expected output is a cumulative application of the monoid operation along the input sequence. These problems are of significant interest, because they serve as a proxy to test state-tracking capabilities of models in a setting similar to tracking chess board state, evaluating code or tracking entity states in a narrative [29]. In the finite-precision setting, Transformers and ostensibly stateful diagonal LRNNs are unable to solve the task—that is, to train on the corresponding data and generalize to inputs of arbitrary lengths. This shows a striking contrast between these models and LSTMs or LRNNs with dense matrix transitions, which are capable of solving the problem.

In the present work, we leverage the sequences from the A_5 symmetric group of even permutations of five elements. We train three variants of FPRNNs - the baseline model following [27], the FPRNN-KAN model with a channel mixer parameterized with a KAN and a complex variant CFPRNN-CVKAN which alters the layer to operate on complex states and leverages a CVKAN layer to parameterize the complex-valued channel mixer. We report the generalization capabilities of the three models in Table 4.

Model	Width	Grid	$L = 16$	$L = 32$	$L = 64$	$L = 128$	$L = 256$
FPRNN	512	-	100.0	98.67	69.8	0.09	0
FPRNN-KAN	512	4	100.0	100.0	91.5	30.4	0
FPRNN-CVKAN	512	4	100.0	99.47	87.1	33.4	14.0

Table 4
Test sequence accuracy of the considered models on the A5 problem. We fix model width at 512, train models on input sequences of length 16 and track their sequence accuracy on inputs of lengths up to 256.

Similarly to the Selective Copying experiments, along with fixed-width comparison, we conduct experiments in Table 5 where instead of the model width we control the number of parameters. In the parameter-controlled setting we observe better stability of the FP-CVKAN model for shorter inputs (high variance is due to models whi ch refused to converge) but worse generalization to longer inputs.

Model	Width	Params	$L = 16$	$L = 32$	$L = 64$	$L = 128$	$L = 256$
FPRNN	512	6.98M	89.15 ±30.68	88.44 ±32.65	87.25 ±33.43	77.43 ±32.45	49.87 ±27.23
			87.50 ±35.36	87.36 ±35.30	82.11 ±35.80	35.79 ±44.84	1.24 ±2.70
FP-KAN	416	6.98M	78.13 ±37.46	70.24 ±41.24	59.70 ±41.52	46.24 ±39.29	28.47 ±25.96
			65.12 ±48.26	57.77 ±48.96	37.87 ±48.75	8.34 ±18.40	0
FP-CVKAN	336	7.29M	99.98 ±0.04	93.03 ±12.99	70.53 ±20.31	41.13 ±17.85	21.60 ±10.90
			99.70 ±0.49	67.23 ±40.80	13.05 ±29.72	0.62 ±1.52	0

Table 5
Test accuracy of the considered models on the A5 problem. The first line for each model denotes token accuracy and the second denotes sequene accuracy. We fix the number of parameters, train models on input sequences of length 16 and track their sequence accuracy on inputs of lengths up to 256. Grid sizes are analogous to Table 4.

4.3. Explainability of LSTM CVKAN

In this experiment we provide a qualitative inspection of the learned representation of our CVKAN LSTM model by analyzing the internal representations of the model. As a simple demonstration task we ask a CVKAN LSTM to continue a real-valued sawtooth pattern with a complex-valued shifted triangle wave over $c = 3$ cycles and $T = 96$ time steps after conditioning the model only on the first cycle as a prefix. Figure 1 shows at the top the real parts of target and prediction over time. The second row shows signed real-valued readout terms of the $w = 8$ hidden units in the network. The third panel visualizes the frequencies of the prediction, target and each individual hidden unit, while the bottom panel showcases the change in forecast MSE when omitting single hidden units from final readout. It can be seen that the CVKAN LSTM is able to continue the periodic pattern on a relatively long time horizon with only little deviation and captures the fundamental target amplitude spectrum.

5. Conclusion and Future Work

To conclude, in the present work we have introduced two CVKAN-driven recurrent architectures. We have shown that in certain experimental setups, namely selective copying and an A_5 state tracking task with no length generalization, they can be a close match of or outperform the models from which they originate. We conducted a fair comparison of the different variants by fixing the parameter count to similar ranges and showed that our proposed model permits qualitative inspection of its latent representations and visualization of hidden unit contributions.

Even though we have not been able to conduct more massive experiments to study these questions, we believe that our results open new research perspectives, especially on scalability and interpretability of KAN-driven recurrent models. We leave larger-scale experiments for future work.

6. Acknowledgments

Uladzislau Yorsh's research was supported by the Grant Agency of Charles University (project no. 290223) and partially by SVV project number 260 821. Martin Holeňa's research was partially supported by the German Research Foundation (DFG) project funding no. 441926934.

Declaration on Generative AI

Large language models (LLMs), specifically ChatGPT 5.5 and 5.6, were used during this work for error checking and sanity checks as well as assisting with code generation, particularly for plotting and visualization. No LLMs were used to generate or write the manuscript text.

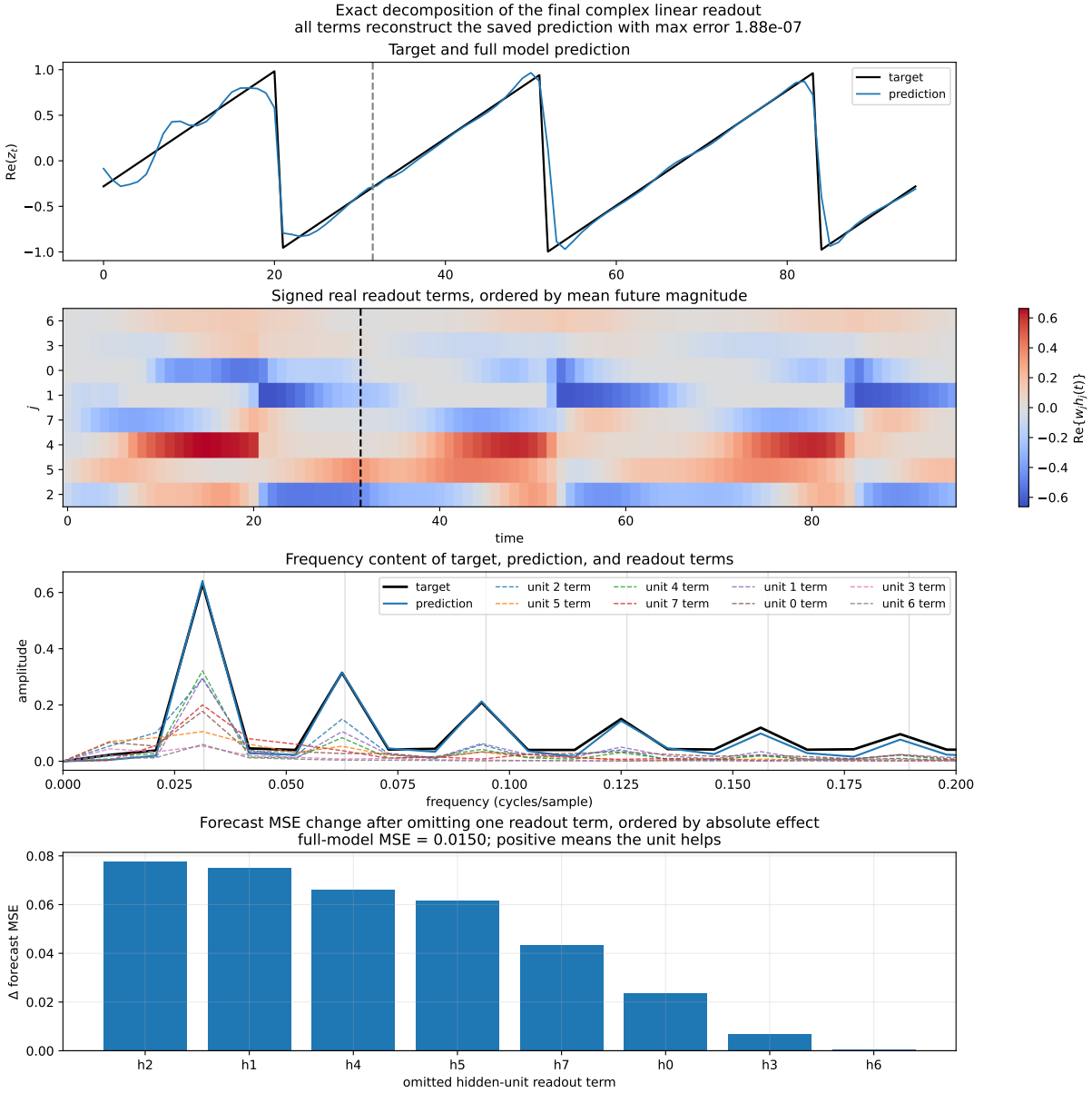


Figure 1: Visualization of prediction and hidden unit output for a CVKAN LSTM with $w = 8$ hidden units, trained on the first cycle of a complex-valued sawtooth pattern with a total of $c = 3$ cycles over $T = 96$ time steps. The target has a sawtooth real component and a phase-shifted triangle-wave imaginary component.

References

[1] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T. Hou, M. Tegmark, Kan: Kolmogorov–arnold networks, in: Y. Yue, A. Garg, N. Peng, F. Sha, R. Yu (Eds.), International Conference on Learning Representations, volume 2025, 2025, pp. 70367–70413. URL: https://proceedings.iclr.cc/paper_files/paper/2025/file/afaed89642ea100935e39d39a4da602c-Paper-Conference.pdf.

[2] R. Yu, W. Yu, X. Wang, Kan or mlp: A fairer comparison (2024). URL: https://arxiv.org/abs/2407.16674. arXiv:2407.16674.

[3] Y. Wang, J. Sun, J. Bai, C. Anitescu, M. S. Eshaghi, X. Zhuang, T. Rabczuk, Y. Liu, Kolmogorov–arnold-informed neural network: A physics-informed deep learning framework for solving forward and inverse problems based on kolmogorov–arnold networks, Computer Methods in Applied Mechanics and Engineering 433 (2025) 117518. URL: https://www.sciencedirect.com/science/article/pii/S0045782524007722. doi:https://doi.org/10.1016/j.cma.2024.117518.

[4] Y. Hou, T. Ji, D. Zhang, A. Stefanidis, Kolmogorov-arnold networks: A critical assessment

of claims, performance, and practical viability (2025). URL: https://arxiv.org/abs/2407.11075. arXiv:2407.11075.

[5] A. D. Bodner, A. S. Tepsich, J. N. Spolski, S. Pourteau, Convolutional kolmogorov-arnold networks (2025). URL: https://arxiv.org/abs/2406.13155. arXiv:2406.13155.

[6] D. Rege Cambrin, E. Poeta, E. Pastor, T. Cerquitelli, E. Baralis, P. Garza, Kan you see it? kans and sentinel for effective and explainable crop field segmentation, in: A. Del Bue, C. Canton, J. Pont-Tuset, T. Tommasi (Eds.), Computer Vision – ECCV 2024 Workshops, Springer Nature Switzerland, Cham, 2025, pp. 115–131.

[7] M. Kiamari, M. Kiamari, B. Krishnamachari, Gkan: Graph kolmogorov-arnold networks (2024). URL: https://arxiv.org/abs/2406.06470. arXiv:2406.06470.

[8] R. Bresson, G. Nikolentzos, G. Panagopoulos, M. Chatzianastasis, J. Pang, M. Vazirgiannis, Kagnns: Kolmogorov-arnold networks meet graph learning (2025). URL: https://arxiv.org/abs/2406.18380. arXiv:2406.18380.

[9] X. Yang, X. Wang, Kolmogorov-arnold transformer, in: Y. Yue, A. Garg, N. Peng, F. Sha, R. Yu (Eds.), International Conference on Learning Representations, volume 2025, 2025, pp. 76063–76086. URL: https://proceedings.iclr.cc/paper_files/paper/2025/file/bd611df389a2e2fd6baa43c381b2b012-Paper-Conference.pdf.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale (2021). URL: https://arxiv.org/abs/2010.11929. arXiv:2010.11929.

[12] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208. doi:https://doi.org/10.1016/0893-6080(89)90020-8.

[13] A. N. Kolmogorov, On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition, American Mathematical Society, 1961.

[14] F. Girosi, T. Poggio, Representation properties of networks: Kolmogorov's theorem is irrelevant, Neural Computation 1 (1989) 465–469.

[15] A. Dash, Blealtan, efficient-kan, 2024. URL: https://github.com/Blealtan/efficient-kan.

[16] Z. Li, Kolmogorov-arnold networks are radial basis function networks (2024). URL: https://arxiv.org/abs/2405.06721. arXiv:2405.06721.

[17] S. SS, K. AR, G. R, A. KP, Chebyshev polynomial-based kolmogorov-arnold networks: An efficient architecture for nonlinear function approximation (2024). URL: https://arxiv.org/abs/2405.07200. arXiv:2405.07200.

[18] Z. Bozorgasl, H. Chen, Wav-kan: Wavelet kolmogorov-arnold networks (2024). URL: https://arxiv.org/abs/2405.12832. arXiv:2405.12832.

[19] A. Noorizadegan, S. Wang, L. Ling, J. P. Dominguez-Morales, A practitioner's guide to kolmogorov–arnold networks, Computer Science Review 62 (2026) 100991. URL: https://www.sciencedirect.com/science/article/pii/S1574013726000997. doi:https://doi.org/10.1016/j.cosrev.2026.100991.

[20] M. Wolff, F. Eilers, X. Jiang, Cvkan: Complex-valued kolmogorov-arnold networks, in: 2025 International Joint Conference on Neural Networks (IJCNN), 2025, pp. 1–9. doi:10.1109/IJCNN64981.2025.11227425.

[21] M. Wolff, F. Alesiani, C. Duhme, X. Jiang, Clifford kolmogorov-arnold networks, in: 2026 International Joint Conference on Neural Networks (IJCNN), 2026. URL: https://arxiv.org/abs/2602.05977. arXiv:2602.05977.

[22] A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, S. De, Resurrecting recurrent neural networks for long sequences, in: A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, J. Scarlett (Eds.), Proceedings of the 40th International Conference on Machine Learning, volume 202 of *Proceedings of Machine Learning Research*, PMLR, 2023, pp. 26670–26698. URL: https://proceedings.mlr.press/v202/orvieto23a.html.

[23] A. Gu, T. Dao, S. Ermon, A. Rudra, C. Ré, Hippo: Recurrent memory with optimal polynomial projections, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1474–1487. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/102f0bb6efb3a6128a3c750dd16729be-Paper.pdf.

[24] R. A. Horn, C. R. Johnson, Matrix Analysis, 2nd ed ed., Cambridge University Press, Cambridge ; New York, 2012.

[25] R. Genet, H. Inzirillo, Tkan: Temporal kolmogorov-arnold networks (2025). URL: https://arxiv.org/abs/2405.07344. arXiv:2405.07344.

[26] S. Lee, J.-K. Kim, J. Kim, T. Kim, J. Lee, Hippo-kan: Efficient kan model for time series analysis (2024). URL: https://arxiv.org/abs/2410.14939. arXiv:2410.14939.

[27] S. Movahedi, F. Sarnthein, N. Muca Cirone, A. Orvieto, Fixed-point rnns: Interpolating from diagonal to dense, in: D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, N. Chen (Eds.), Advances in Neural Information Processing Systems, volume 38, Main Conference, Curran Associates, Inc., 2025, pp. 44873–44908. URL: https://proceedings.neurips.cc/paper_files/paper/2025/file/3f83c5050d900c7e11ffc58e4346a27a-Paper-Conference.pdf. doi:10.52202/085713-1495.

[28] A. Gu, T. Dao, Mamba: Linear-time sequence modeling with selective state spaces, in: First Conference on Language Modeling, 2024. URL: https://openreview.net/forum?id=tEYskw1VY2.

[29] W. Merrill, J. Petty, A. Sabharwal, The illusion of state in state-space models, in: R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, F. Berkenkamp (Eds.), Proceedings of the 41st International Conference on Machine Learning, volume 235 of *Proceedings of Machine Learning Research*, PMLR, 2024, pp. 35492–35506. URL: https://proceedings.mlr.press/v235/merrill24a.html.