


# Kolmogorov–Arnold networks for genomic tasks

Oleksandr Cherednichenko and Maria Poptsova <sup>\*</sup>

International Laboratory of Bioinformatics, HSE University, 11 Pokrovsky Bulvar, Moscow, 109028, Russia

<sup>\*</sup>Corresponding author. International Laboratory of Bioinformatics, HSE University, Moscow, Russia. E-mail: mpoptsova@hse.ru

## Abstract

Kolmogorov–Arnold networks (KANs) emerged as a promising alternative for multilayer perceptrons (MLPs) in dense fully connected networks. Multiple attempts have been made to integrate KANs into various deep learning architectures in the domains of computer vision and natural language processing. Integrating KANs into deep learning models for genomic tasks has not been explored. Here, we tested linear KANs (LKANs) and convolutional KANs (CKANs) as a replacement for MLP in baseline deep learning architectures for classification and generation of genomic sequences. We used three genomic benchmark datasets: Genomic Benchmarks, Genome Understanding Evaluation, and Flipon Benchmark. We demonstrated that LKANs outperformed both baseline and CKANs on almost all datasets. CKANs can achieve comparable results but struggle with scaling over large number of parameters. Ablation analysis demonstrated that the number of KAN layers correlates with the model performance. Overall, linear KANs show promising results in improving the performance of deep learning models with relatively small number of parameters. Unleashing KAN potential in different state-of-the-art deep learning architectures currently used in genomics requires further research.

**Keywords:** Kolmogorov–Arnold networks; deep learning; large language models; diffusion models; genomic benchmarks; regulatory genomics

## Introduction

Deep learning models have been successfully applied to a wide range of genomic tasks including prediction of variant effect [1, 2], and location of functional genomic elements, such as promoters [3], enhancers [4], transcription factor binding sites [5], histone marks [6, 7], flipons [8, 9], splice sites [10], and others. Neural networks also proved to be efficient in the field of RNA biology (see [11] and [12] for review) predicting RNA-protein binding [13], RNA structure [14], and noncoding RNA [15]. The evolution of deep learning applications in genomics followed the path of the evolution of deep learning architectures. It started with pioneering application of convolutional neural networks (CNNs) [1, 16] and recurrent neural networks (RNNs) to DNA sequences [17], and then continued with Transformer-based large language models (LLMs) such as DNABERT [18], DNABERT2 [19], and Nucleotide Transformer [20]. DNABERT was trained on one reference human genome, while DNABERT2 and Nucleotide transformer were trained on the multi-species genome comprising up to 40B of bases pairs. After training, foundation models are fine-tuned on different downstream tasks. However, even with parameter efficient fine-tuning techniques, resource limitations can impose significant constraints on the task execution.

Transformers require substantial computational resources due to quadratic attention scaling, and the next architecture Hyena, based on long convolutions, incorporated longer context up to 1M nucleotides by scaling subquadratically, which resulted in HyenaDNA [21]. Another promising alternative to overcome transformers' computational inefficiency is Mamba architecture [22], which lies at the basis of Caduceus model in genomics

[23]. Both Hyena and Mamba achieved a certain tradeoff between computational resources and performance quality. In comparison with the latest genomic foundational model DNABERT2, HyenaDNA has 10x less parameters and on some genomic tasks outperforms DNABERT2.

With an evergrowing size of deep learning architectures, it has been a challenge to find smaller architectures that would perform equally well. Here, we aim to explore the potential of small neural networks that utilize only few multilayer perceptrons (MLPs), incorporating recent innovations in deep learning, such as Kolmogorov–Arnold networks (KANs) [24].

The neural networks discussed above have one thing in common: they rely on the universal approximation theorem. One of the recent advances in deep learning architectures, KAN [24], leverages the Kolmogorov–Arnold theorem to incorporate splines into a neural network architecture, offering a compelling alternative to traditional MLPs. KANs have already been successfully applied in different areas such as mechanics [25, 26], computer vision [27–31], NLP [32, 33], time series [33–35], physics [24, 25, 36], speech enhancement [37], molecular representations [38, 39], and recommendation systems [40]. Inspired by this advancement, multiple modifications have emerged that attempt to overcome various issues associated with the spline-based approach, namely computational overhead and a large number of trainable parameters.

In this study we aim at evaluating the potential of KAN-based models in genomics. Given the limitations in computational resources, we applied KAN to simple convolutions and dense networks with a relatively small number of parameters. We tested

**Received:** December 9, 2024. **Revised:** February 12, 2025. **Accepted:** March 5, 2025

© The Author(s) 2025. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0/>), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

KANs to the task of classification and generation by replacing layers in the baseline model with linear KAN layers (LKAN) and convolutional KAN (CKAN) layers. The test was performed on three benchmark datasets: Genomic Benchmarks [41], Genome Understanding Evaluation (GUE), and our collection of datasets for flipons or non-B DNA structures [42].

## Kolmogorov–Arnold networks

### Linear Kolmogorov–Arnold networks

Here we briefly introduce KAN and present key features of this architecture. MLPs [43] are inspired by the universal approximation theorem that states that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of  $\mathbb{R}^d$ . KAN [44] focuses on the Kolmogorov–Arnold representation theorem [45], which states that any multivariate continuous function can be represented as a composition of univariate functions and the addition operation:

$$f(\mathbf{x}) = \sum_{k=1}^{2n+1} \Phi_k \left( \sum_{l=1}^n \phi_{k,l}(x_l) \right), \quad (1)$$

where  $\phi_{k,l}$  are univariate functions that map each input variable  $x_l$  as follows:  $\phi_{k,l} : [0, 1] \rightarrow \mathbb{R}$ . The authors of the original KAN paper proposed how to extend the KAN application to deep learning networks by stacking KAN layers.

$$f(\mathbf{x}) = \sum_{i_{L-1}=1}^{n_{L-1}} \phi_{L-1,i_{L-1}} \left( \sum_{i_{L-2}=1}^{n_{L-2}} \left( \sum_{i_0=1}^{n_0} \phi_{0,i_1,i_0}(x_{i_0}) \right) \right) \quad (2)$$

Similarly it is possible to rewrite it in the following way:

$$\text{KAN}(\mathbf{x}) = (\Phi_{L-1} \circ \dots \circ \Phi_1 \circ \Phi_0)\mathbf{x} \quad (3)$$

Implementation of KANs is provided by the authors of [24]; however there are several aspects that could limit the scalability of KANs: a large number of trainable parameters, training time, and inference time. On top of that, in [32] the authors claim that weights have different initialization that makes it impossible to keep variance-preserving initialization [29]. To overcome these issues, in [29] the authors suggest an efficient implementation of KANs with the straightforward matrix multiplication and L1 regularization for the model's weights.

### Kolmogorov–Arnold convolutions

Kolmogorov–Arnold convolutions are proposed in [27] and designed to be similar to CNNs. The difference is that the convolutional layers are replaced by KAN convolutional layers, and, after flattening, one can use either KAN or MLP. The advantage of the CKANs is that they have significantly fewer parameters compared with other architectures. This is provided by the network architecture, because B-Splines are capable of smooth representations of arbitrary activation functions, which cannot be determined with ReLU. The authors concluded that Kolmogorov–Arnold convolutions are capable of achieving high performance as compared with original convolutional networks [28].

## Genomics tasks

### DNA classification

Classification of DNA sequences is one of the most common tasks in genomics, which is solved with deep learning models.

In this work, we perform classification on three large benchmark datasets: Genomic Benchmarks [41], Genomic Understanding Evaluation [19], and Flipon Benchmark assembled by us. For each of the datasets a train and a test sets are available.

### DNA generation

Another important application of deep learning in genomics is generative modeling. Generative models are usually used in data augmentation approaches to generate new samples from the learned distribution.

There are many types of generative models, but here we focus on the most popular models for DNA sequence generation: denoising diffusion implicit model (DDIM) [46] and generative adversarial network (GAN) [47]. In generative approaches, first, a generative model is trained on a train set, and the trained model produces synthetic samples of the same size as the test set. Secondly, synthetic data are supplied to a classifier trained on the real data to evaluate how good are generative models in capturing main patterns of the real data.

We will use the Flipons collection [48–53] for this task (see Methods).

## Methods

### Models

We use pytorch efficient implementation of KAN (EKAN) [29] as linear KAN layers. We use KAN-Conv [27] implementation as KAN convolutional neural network. We incorporate KAN layers into convolutional neural network that takes its core from Leg-Net [54]. We compare the impact of KAN layers by replacing MLP modules with KANs (see Tables 1–5). To make a comparison more fair and to exclude model size effect, in all the experiments for classification we used models of equal size: 22M parameters. We used the following notation for our tested models (Fig. 1): Baseline for LegNet-based convolutional neural network, LKAN for LegNet-based convolutional neural network [55] with incorporated Linear KANs, and CKAN for LegNet-based convolutional neural network with incorporated Convolutional KANs. Since CKANs scales with larger grid size (see Results), combining CKAN and LKAN in our architecture was impossible due to limitations in resources, thus hybrid CKAN-LKAN architectures were not conducted.

In experiments with DNA sequence generation we utilized DDIM [46] designed specifically for DNA [57]. We replaced linear layers with KAN layers in Unet like architecture (Fig. 2).

We used a Wasserstein GAN [47] as another commonly used architecture to produce synthetic DNA sequences. Also, we replaced MLP block with KAN layers to create LGAN and CGAN (Fig. 3).

## Datasets

### Genomic benchmarks

We use the following notations for all datasets: DMEE—dummy mouse enhancers ensemble, DCIS—demo coding vs intergenomic seqs, DHW—demo human or worm, DES—drosophila enhancers stark, HEC—human enhancers coh, HEE—human enhancers ensembl, HER—human ensembl regulatory, HNP—human non-tata promoters, HOE—human ocr ensembl.

### Flipons, or non-B DNA structures

Inspired by Genomic Benchmarks we created a collection of datasets for detecting flipons [42], or non-B DNA structures: Z-flipons, or Z-DNA [48, 49]; G-flipons, or G-quadruplexes (GQs) [50–53]; H-flipons, or H-DNA (triplexes) [48]. We provide train and test sets trying to prevent data leakage. Full description

Table 1. KAN performance on classification task for genomic benchmarks based on MCC-score averaged over five-folds

Data set	Baseline	LKAN	CKAN
DMEE	70.2 ± 0.5	<b>73.2 ± 0.2</b>	64.8 ± 1.3
DCIS	81.1 ± 0.5	<b>85.5 ± 1.4</b>	81.7 ± 1.2
DHW	89.2 ± 0.5	<b>90.2 ± 0.5</b>	88.1 ± 0.8
DES	<b>27.3 ± 0.9</b>	28.9 ± 0.6	25.3 ± 0.6
HEC	66.6 ± 0.8	<b>68.2 ± 0.5</b>	67.9 ± 0.8
HEE	63.2 ± 0.5	<b>69.9 ± 0.5</b>	59.0 ± 1.0
HER	<b>90.0 ± 0.8</b>	89.3 ± 0.8	88.4 ± 0.6
HNP	87.9 ± 0.9	<b>88.8 ± 0.9</b>	85.0 ± 1.0
HOE	70.8 ± 0.6	<b>74.4 ± 0.7</b>	68.9 ± 1.0

The best values are in **bold**, and the second best values are in *italics*.

Table 2. KAN performance on classification task for flipons based on MCC-score averaged over five-folds

Data set	Baseline	LKAN	CKAN
ENDO	89.1 ± 0.5	<b>93.5 ± 0.4</b>	89.0 ± 0.8
G4SEQ	83.3 ± 0.6	<b>89.0 ± 0.8</b>	84.5 ± 0.2
G4CHIP	90.2 ± 0.9	<b>92.2 ± 0.7</b>	91.3 ± 1.5
G4CUT	91.1 ± 0.8	<b>94.8 ± 0.7</b>	85.6 ± 1.2
ZKOU	94.5 ± 1.0	<b>96.3 ± 0.8</b>	94.0 ± 0.8
ZSHIN	<b>97.9 ± 0.5</b>	97.0 ± 0.5	95.7 ± 1.1
HKOU	90.5 ± 0.6	<b>95.2 ± 1.1</b>	88.7 ± 0.9

The best values are in **bold**, and the second best values are in *italics*.

of datasets are available in Supplementary Table 1. Similar to Genomic Benchmarks datasets we use the following notations for all flipons datasets: ENDO—Endoquad GQs; G4seq—G4-seq experimental dataset for GQs; G4ChIP—G4 ChIP-seq experimental data set for GQs; G4cut—G4 CUT&Tag experimental dataset for GQs; ZKOU—Kouzine et al. experimental dataset for Z-DNA; ZShin—Shin et al. ChIP-seq experimental dataset for Z-DNA; HKOU—Kouzine experimental dataset for H-DNA.

### Genome understanding evaluation

We follow [19] authors in evaluation on Genome Understanding Evaluation (GUE) and extended version (GUE+) to study models' performance on a wider range of genomic tasks: prediction of promoter, core promoter, splice sites, epigenetic marks, transcription factor (TF) binding sites on human and mouse, and covid variant classification.

## Results

### Benchmarks on DNA classification

#### Genomic benchmarks

KAN performance in comparison with the baseline model for nine datasets from Genomic Benchmarks [41] are presented in Table 1 (MCC score averaged from five-folds). Baseline CNN model is of equal size with LKAN and CKAN. Other classification metrics such as accuracy, ROC-AUC, precision, recall, and F1 are provided in Supplementary Tables 2–14. We observe that both LKAN and CKAN improve the quality of the model, yet CKAN requires higher value of grid size parameter.

#### Flipons, or non-B DNA structures

KAN's performance on seven datasets from Flipon collection [48–53] is presented in Table 2. Baseline CNN model is of equal size with LKAN and CKAN.

Table 3. KAN performance on classification task for GUE based on MCC averaged over five-folds

Data set	Baseline	LKAN	CKAN
H3	70.7 ± 1.5	<b>71.3 ± 1.0</b>	69.4 ± 1.5
H3K14AC	40.2 ± 0.8	<b>43.2 ± 0.7</b>	41.0 ± 1.0
H3K36ME3	44.3 ± 0.9	<b>47.7 ± 0.6</b>	42.9 ± 0.9
H3K4ME1	35.6 ± 0.4	<b>40.0 ± 0.8</b>	35.8 ± 1.1
H3K4ME2	27.2 ± 0.5	<b>29.3 ± 0.5</b>	24.3 ± 1.2
H3K4ME3	28.4 ± 0.5	<b>29.8 ± 0.4</b>	27.2 ± 0.7
H3K79ME3	60.8 ± 0.9	<b>62.1 ± 0.8</b>	61.1 ± 1.5
H3K9AC	50.5 ± 0.6	<b>51.9 ± 0.5</b>	50.0 ± 0.9
H4	75.6 ± 0.9	<b>77.2 ± 0.7</b>	70.9 ± 0.8
H4AC	33.9 ± 0.8	<b>38.1 ± 0.8</b>	36.2 ± 0.9
PD ALL	75.9 ± 1.1	<b>81.2 ± 1.4</b>	77.2 ± 1.2
PD NOTATA	88.2 ± 1.1	<b>89.9 ± 0.7</b>	87.1 ± 1.2
PD TATA	56.1 ± 0.5	<b>58.2 ± 0.8</b>	57.9 ± 0.9
TF HUMAN 1	69.9 ± 1.0	<b>72.4 ± 0.8</b>	66.3 ± 1.3
TF HUMAN 2	49.6 ± 0.4	<b>54.3 ± 0.7</b>	47.2 ± 0.7
TF HUMAN 3	43.2 ± 0.5	<b>46.8 ± 0.5</b>	44.4 ± 0.9
TF HUMAN 4	72.6 ± 0.8	<b>72.9 ± 0.5</b>	71.3 ± 0.9
TF HUMAN 0	65.1 ± 0.7	<b>67.4 ± 0.6</b>	65.5 ± 0.9
CPD ALL	64.8 ± 0.9	<b>68.2 ± 0.6</b>	66.2 ± 1.1
CPD NOTATA	64.5 ± 0.7	<b>67.2 ± 0.8</b>	63.1 ± 1.1
CPD TATA	72.3 ± 0.9	<b>78.2 ± 0.9</b>	74.7 ± 1.2
TF MOUSE 1	81.0 ± 0.7	<b>82.8 ± 0.6</b>	80.1 ± 0.8
TF MOUSE 2	79.8 ± 0.6	<b>85.5 ± 0.8</b>	81.2 ± 0.7
TF MOUSE 3	77.6 ± 1.2	<b>81.4 ± 0.7</b>	79.2 ± 0.9
TF MOUSE 4	42.6 ± 0.7	<b>48.7 ± 0.5</b>	40.1 ± 0.7
TF MOUSE 0	38.2 ± 0.7	<b>40.1 ± 0.6</b>	39.0 ± 0.9
Virus	50.3 ± 0.9	<b>58.2 ± 1.5</b>	50.5 ± 0.6
Splice	78.2 ± 0.8	<b>80.0 ± 0.8</b>	75.2 ± 0.7

The best values are in **bold**, and the second best values are in *italics*.

### Genome understanding evaluation

Results of KAN's performance in comparison with the baseline model for nine datasets from Genome Understanding Evaluation [19] are presented in Table 3 (MCC score averaged from five-folds). Baseline CNN model is of equal size with LKAN and CKAN.

We can see that in almost all classification tasks LKAN outperforms Baseline and CKAN models on average by 7.71% and 9.59%, respectively. The largest difference was detected for virus classification with 70.53% increase over baseline and 59.73% increase over CKAN, but this is because the baseline model does not perform well having only an MCC of 20.7.

Increase by 16%–22% in LKAN over Baseline and 16%–30% in LKAN over CKAN was reached for enhancer–promoter interaction datasets, and here too the baseline models do not perform well and have an MCC range of 34.3–38.2. For histone marks we observe a larger difference in performance for those genomic elements that are not predicted well by baseline models: these are H3K4ME1 (12.36% for LKAN over Baseline with MCC of 35.6) and to a lesser extent H3K4ME2 (8% for Baseline with MCC of 27.2) and H3K4ME3 (5% for LKAN over Baseline with MCC of 28.4). For human TFs, the biggest difference is for PAX5 (9%) and TRIM28 (8%), which are not well-predicted by baseline (MCC 49.6 and 43.2 correspondingly). The same trend is observed for mouse TFs with the largest increase of 14% for Nelfe, which has a low MCC of 42.6 for baseline. For high MCC baseline models the increase is small such as for human H3 (0.85%) with the Baseline MCC of 70.7 and MXI1 (0.41%) with the Baseline MCC of 72.6. For flipon dataset, an increase in

Table 4. KAN performance on classification task for GUE+ based on MCC averaged over five-folds

Data set	Baseline	LKAN	CKAN
GM12878	36.6 ± 1.1	<b>41.2 ± 0.9</b>	37.0 ± 1.2
HELA-S3	34.3 ± 1.0	<b>40.8 ± 1.2</b>	31.3 ± 0.9
HUVEC	37.2 ± 0.7	<b>39.6 ± 0.9</b>	30.2 ± 0.8
IMR90	34.5 ± 0.9	<b>42.0 ± 0.9</b>	33.3 ± 1.3
K692	38.2 ± 0.9	<b>44.4 ± 0.7</b>	37.9 ± 1.1
NHEK	33.7 ± 1.1	<b>40.1 ± 0.9</b>	34.4 ± 0.6
FUNGI	51.1 ± 0.6	<b>55.5 ± 1.0</b>	50.4 ± 1.2
VIRUS	20.7 ± 0.5	<b>35.3 ± 1.9</b>	22.1 ± 1.8

The best values are in **bold**, and the second best values are in *italics*.

Table 5. KANs for DNA generation with DNA-Diffusion: KAN loss (SP-MSE) comparison on flipon generation

DNA-Diffusion			
Task	Unet (410M)	LKAN (440M)	CKAN (820M)
G4	0.0334	<b>0.0276</b>	0.0340
ZDNA	0.0301	<b>0.0222</b>	0.0299
HDNA	0.0388	<b>0.0281</b>	0.0379

The best values are in **bold**, and the second best values are in *italics*.

performance is also small in 2%–6% for LKAN over Baseline and 1%–11% for LKAN over CKAN but the Baseline model range is also high 83.3–97.9.

As for CKAN performance, on many genomic datasets CKAN showed up the second-best after LKAN, and this was the case for different classes of genomic elements such as histone marks, TFs, promoter–enhancer interactions, flipons. Here we think it is early to make a conclusion because incorporation of CKAN layers is not a trivial task and still a subject of active discussion (see Discussion), but our results point to the promising usage of CKANs along with LKANs too.

In general we can see the larger increase in performance for LKAN for less structured data, i.e. functional genomic elements lacking a clear DNA sequence motif or regular pattern, such as enhancer–promoter interactions, or widely distributed histone marks, or TF binding sites without a well-defined DNA motif. This suggests that LKAN’s architecture is particularly effective for more challenging genomic tasks. In contrast, for data with strong sequence patterns—such as the Z-DNA in the Kouzine dataset, which exhibits an alternating purine-pyrimidine pattern—the baseline MCC of 94.5 is already quite high; consequently, the increase to 96.3 with LKAN is less pronounced.

Considering results for the classification tasks we observe that Linear KAN scales better than convolutional KAN of the same size. We present visualization of averaged model’s performance for models of various size in Fig. 3. Additionally, we provide results of experiments on how grid size parameter affects the performance of LKAN and the number of total parameters along with the time per batch training. We evaluated these results on GUE datasets.

## Benchmarks on DNA generative models

To test KAN’s capability in producing synthetic DNA sequences we incorporated Linear KAN layers into denoising Unet from DNA Diffusion framework [57]. We compared final loss for various models (Table 5) and observed that LKAN is capable of reaching the lowest validation loss given equal amount of parameters with baseline Unet. CKAN requires x2 parameters to achieve the comparable metrics.

Table 6. KANs for DNA generation with Wasserstein GAN: comparison of edit distance and wasserstein cost on flipon generation

Wasserstein GAN			
Task	WGAN (20M)	LGAN (20M)	CGAN (30M)
G4	250	<b>252</b>	249
ZDNA	223	<b>225</b>	220
HDNA	253	<b>255</b>	250

The best values are in **bold**, and the second best values are in *italics*.

Additionally, we observe that KAN affects training stability of Unet, as it is presented in Fig. 5. Both trainings utilize exponential moving average (EMA) for smoother performance. We conclude that LKANs provide better performance than CKANs, yet CKANs require higher values for grid size, which significantly makes the model heavy in terms of parameters. For instance, CKAN requires 820M parameters, while LKAN requires 440M to outperform baseline Unet.

On top of that, we tested KAN by adding it into architecture of Wasserstein GAN [47]. WGAN architecture is composed of a generator and a discriminator (see Fig. 3), and each of these networks consists of five ResNet blocks. Each block consists of a convolutional layer with 5x5 kernel and padding 2, followed by ReLU activation. Training of all GAN models - WGAN, LGAN, CGAN - was accompanied by calculation of two edit distances [58]:  $D_{self}$  is the average edit distance in a set of 1000 generated sequences where distance is measured between the sequence and its nearest neighbor except for itself (Table 6). This metric is also measured in the test and training datasets (Fig. 6). If the generated data are as diverse as the data in the training and test datasets, the values of  $D_{self}$  in the generated sequences will be similar.  $D_{train}$  is the average distance that is used to measure GAN training. The low value of this metric alerts the overfitting of the GAN on the training set.

We compare the performance of KAN in generative design by evaluating Kullback–Leibler (KL) divergence and Wasserstein distance (WD) (Fig. 7 and 8). Since diffusion model requires substantial resources for training and inference (see Supplementary Materials for details), we conduct these experiments only for flipon’s datasets [58]. We compare the diversity calculated on edit distance within a sample of synthetic sequences for various models (Table 7). Higher values ensure more diversity within a sample. Trained generative models have captured the prior distributions of flipons based on calculated distances between real and generated distributions.

We observe that diversity is relatively the same for all types of models. Incorporation of linear and convolutional layers into DNA-Diffusion framework, a large U-net-like model with ResNet convolutional blocks, might not be the core factor influencing generation of diverse synthetic sequences. In all experiments LKAN outperforms both CKAN and Baseline by a very small margin (Table 7). We also note that the absolute values of diversity of real sequences in our datasets are also around 72 depending on different flipons [58]. And here for the generation task CKAN can be more efficient than LKAN if generated sequences have an underlying pattern as in the case with quadruplexes, though the overall difference between LKAN and CKAN in performance is <1%. Here we conclude that current implementation of KAN layers does not significantly affect the diversity of synthetic samples.

## Ablation study

Additionally, we investigated the impact of the total amount of replaced MLP layers with KANs. For this experiment we used



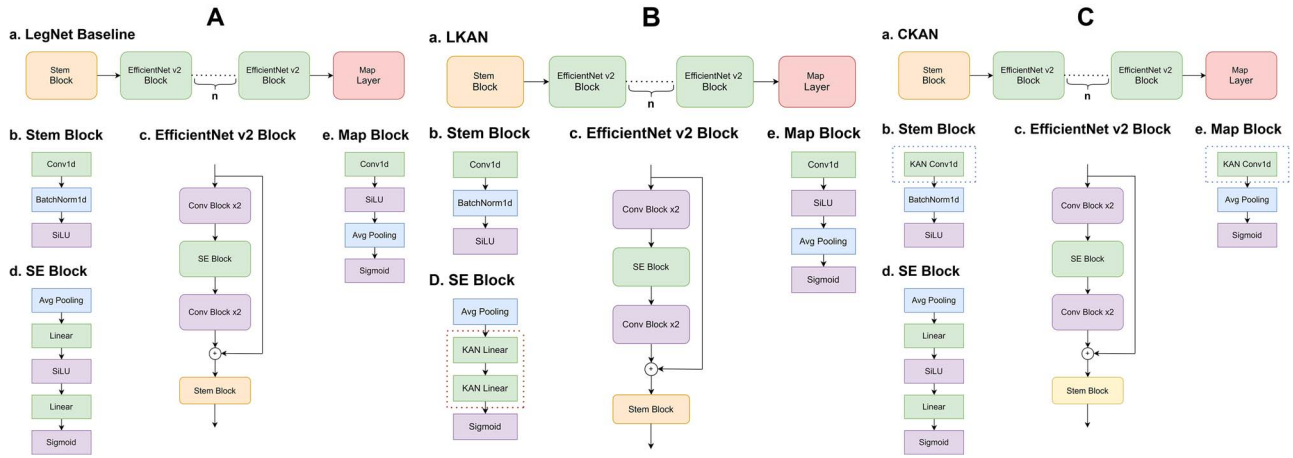


Figure 1. **A. LegNet-based Baseline.** **a.** Architecture of Baseline LegNet-based convolutional neural network.  $m = 6$  **b.** Stem block. **c.** Main convolutional EfficientNet-like [55] block. **d.** Mid SE block with bilinear layer. **e.** Final block. **B. LegNet-based LKAN.** **a.** Architecture of Baseline LegNet-based convolutional neural network with LKAN. **b.** Stem block. **c.** Main convolutional EfficientNet-like [55] block. **d.** Mid SE block with replaced LKAN layers. **e.** Final block. **C. LegNet-based CKAN.** **a.** Architecture of Baseline LegNet-based convolutional neural network with CKAN. **b.** Stem block with replaced CKAN layer. **c.** Main convolutional EfficientNet-like [55] block. **d.** Mid SE block. **e.** Final block with replaced CKAN layer.

Table 7. KANs for DNA generation: metrics of diversity for flippers

DNA-Diffusion			
Task	Unet (410M)	LKAN (440M)	CKAN (820M)
G4	69.2	70.3	<b>73.2</b>
ZDNA	70.4	<b>72.2</b>	71.1
HDNA	64.8	<b>64.9</b>	64.8

Wasserstein GAN			
Task	Unet (20M)	LKAN (20M)	CGAN (30M)
G4	68.9	<b>71.0</b>	70.7
ZDNA	70.5	<b>71.9</b>	71.2
HDNA	63.6	<b>64.7</b>	64.4

The best values are in **bold**, and the second best values are in *italics*.

human enhancers cohn dataset from genomic benchmarks collection. Taking into consideration the total number of EfficientNet-like blocks (Supplementary Table 16) we decided to determine if the same quality can be achieved by replacing only few blocks with LKANs. Model performance based on F1 score with the number of replaced MLP layers and the grid size is presented in Supplementary Table 16. We observe a steady increase in performance with the number of replaced layers. We also see that increasing grid size positively impacts the overall performance. Thus, we found that the best performance is achieved when all blocks are replaced with LKANs; however, with higher grid size values it is possible to achieve fair performance with  $N = 4$  or  $N = 5$  (see Supplementary Tables 16 and 17). Yet, increasing grid size makes model heavier, therefore the training process takes longer. Moreover, higher grid size is prone to certain overfit, when training metrics are close to high values, while testing metrics remain constant (Supplementary Figure 1).

## Discussions and conclusions

In this study, we evaluated performance of the recently developed KANs in the domain of genomics. For that we used a wide range of datasets to cover different types of genomic functional elements

including promoters, enhancers, histone marks, and others collected in two published genomic benchmark datasets and one novel benchmark dataset for flippers, or non-B DNA structures, assembled by us. For our baseline models i.e. models without KAN, we utilized LegNet-based convolutional neural network for DNA classification [54], Wasserstein GAN [47], and DDIM DNA-Diffusion with a core Unet module for generating synthetic sequences [57]. MLP and convolutional layers in these baselines were replaced with linear and convolution KAN layers correspondingly.

Our benchmarking shows that LKANs show promising results in replacing MLPs in a broad range of genomic tasks, while CKANs struggle with scaling over large number of parameters, which require more computational resources. In sequence generation we proved that all models, baseline and KANs, can learn distributions of the input data (Figs 5 and 6); however LKAN outperforms baseline and CKAN in terms of loss, while CKAN requires x2 more parameters to achieve the same performance as LKAN. In diversity of generated samples both LKANs and CKANs outperform baseline but the difference is  $<1\%$ . To enhance diversity in generative tasks, the learning paradigm (i.e diffusion process or adversarial training) plays a major role.

In the study published as preprint [59] the authors utilize the original vanilla version of KAN [24] as the final layer of their models, which are hybrid CNN, RNN, and attention networks named CRA-KAN. CRA-KAN outperforms state-of-the-art models on 50 ChIP-seq datasets, which is in line with the results we presented using another implementation of LKAN [27].

In the original KAN paper [24] the authors demonstrated that KANs improve in accuracy over MLPs on small-scale tasks. We showed that improvement is higher for less structured genomic data. Implementation of B-splines [25] is more technically accurate than vanilla MLP approximation approach. In addition, LKAN utilizes several advanced techniques such as efficient initialization of the spline scaler and weights regularization that can also contribute to LKAN superior performance.

One of the reasons why CKAN did not perform equally well with LKAN with the same number of parameters is that more experiments are needed to determine the best place for CKAN in the architecture (Fig. 1). In our set of experiments, we followed the best practices presented in [27, 28], suggesting the rationale for putting CKAN into a mapping block. In computer vision,

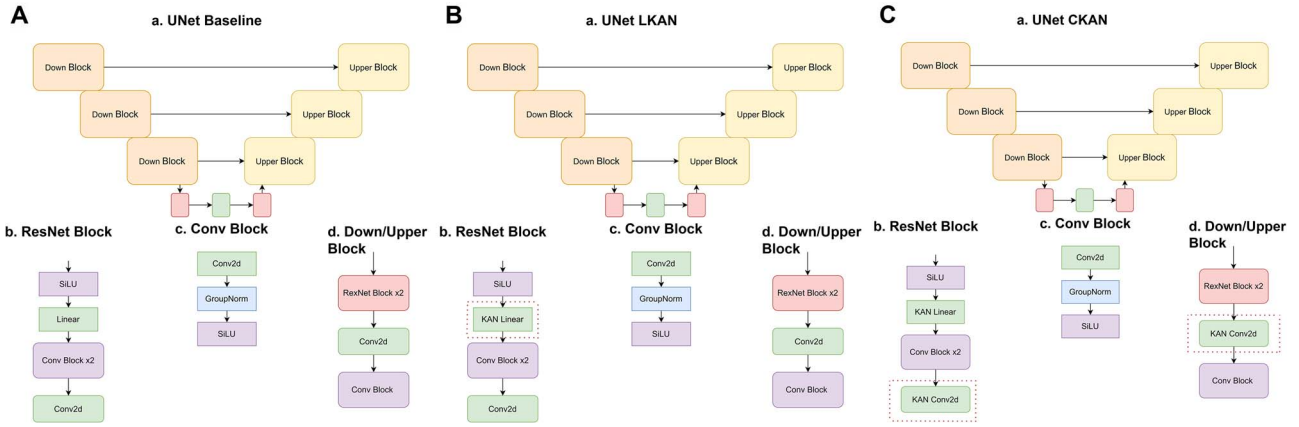


Figure 2. **A. Unet-based Baseline.** a. Architecture of Baseline DNA-Diffusion Unet-based model. b. Base ResNet block. c. Convolutional Block at the end of upper and down blocks. d. Down and upper blocks with ResNet blocks. **B. Unet-based LKAN.** a. Architecture of LKAN modification of DNA-Diffusion Unet-based model. b. Base ResNet block where dense layer is replaced with LKAN. c. Convolutional block at the end of upper and down blocks. d. Down and upper blocks with ResNet blocks. **C. Unet-based CKAN.** a. Architecture of CKAN modification of DNA-Diffusion Unet-based model. b. ResNet block where convolutional blocks are replaced with CKANs. c. Convolutional block at the end of upper and down blocks. d. Down and upper block with ResNet block where convolutional layer is replaced with CKAN.

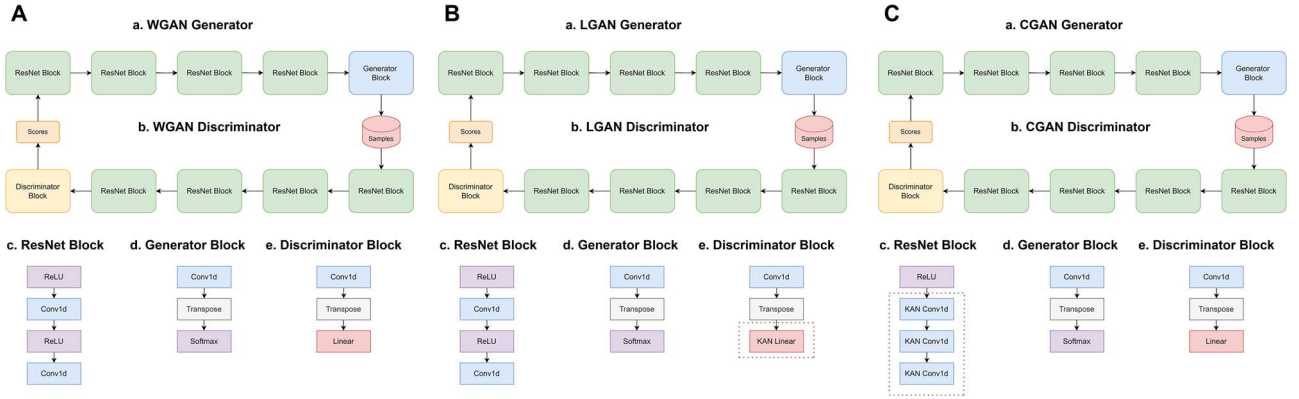


Figure 3. **A. Wasserstein GAN Baseline.** a. ResNet-based WGAN Generator. b. ResNet-based WGAN Discriminator. c. ResNet block. d. Generator block. e. Discriminator block. **B. Wasserstein GAN with LKAN.** a. ResNet-based LGAN Generator. b. LGAN Discriminator where MLP is replaced with LKAN. c. Base ResNet block. d. Generator block. e. Discriminator block with LKAN. **C. Wasserstein GAN with CKAN.** a. ResNet-based CGAN Generator. b. ResNet-based CGAN Discriminator. c. ResNet block with CKAN layers. d. Generator block. e. Discriminator block.

experiments with parameter-lighter models [27] showed that placing CKAN may enhance the performance by 0.8%–1.5%. We did not perform the placement experiments due to the lack of resources but we think there is a positive trend for larger CKAN sizes as indicated in the transition from 8M to 15M (Fig. 4). With more advanced implementation techniques (replacing B-splines with GPU parallelizable functions [36]) and training techniques (parallelizable Batch Ensembling [60]) it could be possible to scale CKANs further. Yet, in our work, we conclude that for this moment, LKAN incorporated into LegNet-like architecture [54] may be used as an improvement for downstream classification.

In different domains, such as computer vision and natural language processing, it is not yet clear if KANs significantly outperform MLPs as it is a topic of active discussions [28, 32, 35, 61]. Our results are preliminary since we have not tested models with large grid size parameters and did not test other types of KANs. Recent advancements in the KAN field include RNN-based and Transformer-based KANs. Temporal Kolmogorov–Arnold Networks using Long Short-Term Memory mechanism to leverage time dependency in KANs were proposed in [33]. Kolmogorov–Arnold Transformer (KAT) architecture is based on

Vision Transformer [62] and utilizes rational polynomial functions to replace splines [29]. KATs were applied to computer vision tasks and, to our knowledge, have not been yet adapted for natural language processing, making it difficult for testing on genomic tasks.

An additional long training (500 epochs) experiment with H3 dataset revealed that training of LKAN is 2.7 times longer than the Baseline. Also we do not observe the trend that LKAN can generalize faster than the Baseline given the same amount of training steps. We consider the different number of epochs for different datasets as they vary in sizes and complexity. In addition, we observe that LKAN can overfit with large grid size parameter (Supplementary Figure 1) after a long training.

To outline the computational difference between standard convolutions and Kolmogorov–Arnold convolutions we conducted an additional experiment with comparison of one CKAN layer with one convolutional layer (Supplementary Table 18). The results showed that for small batch of shape [1, 4, 500], it takes 103 ms and 6000 FLOPs for conv1d layer from pytorch versus 2 s (x19) and 62 000 FLOPs (x10) for CKAN. Authors in [27] report that in the computer vision domain for ImageNet datasets their implementation of KAN Conv training 10x longer than for vanilla

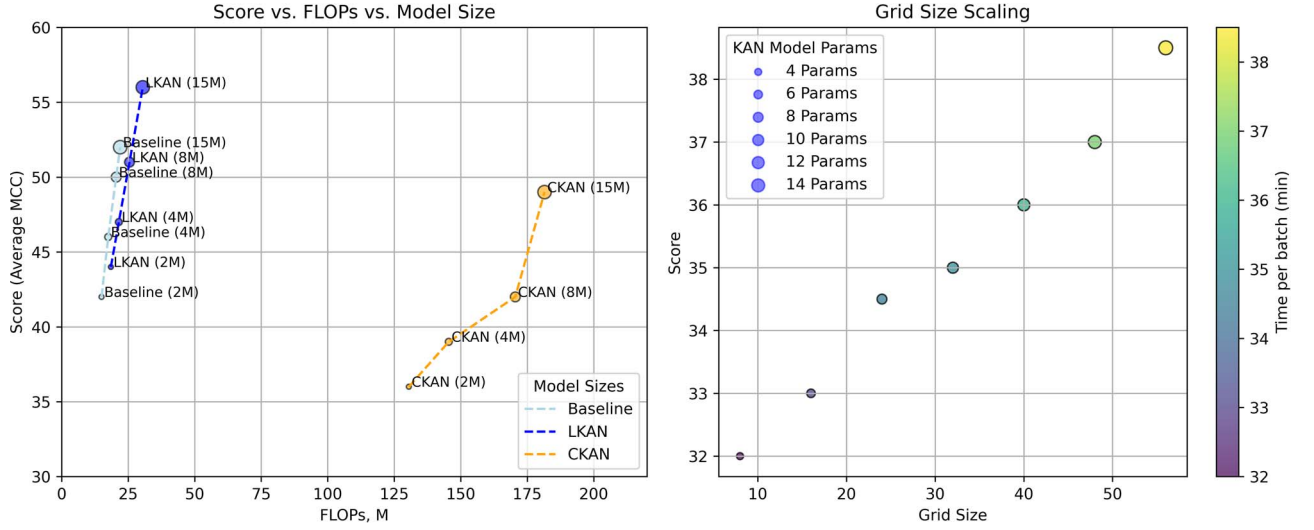


Figure 4. Scaling of models. A. Comparison of the tested models of various sizes, evaluated on different collections of datasets. B. Linear KAN grid size scaling on model's parameters and training time on GUE+ dataset.

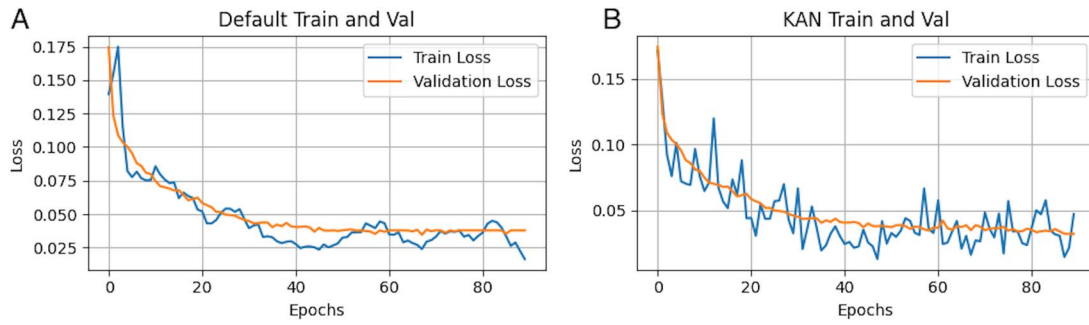


Figure 5. Loss functions of generative models. A. Baseline Unet incorporating EMA. B. Unet with MLP replaced by LKAN and incorporating EMA.

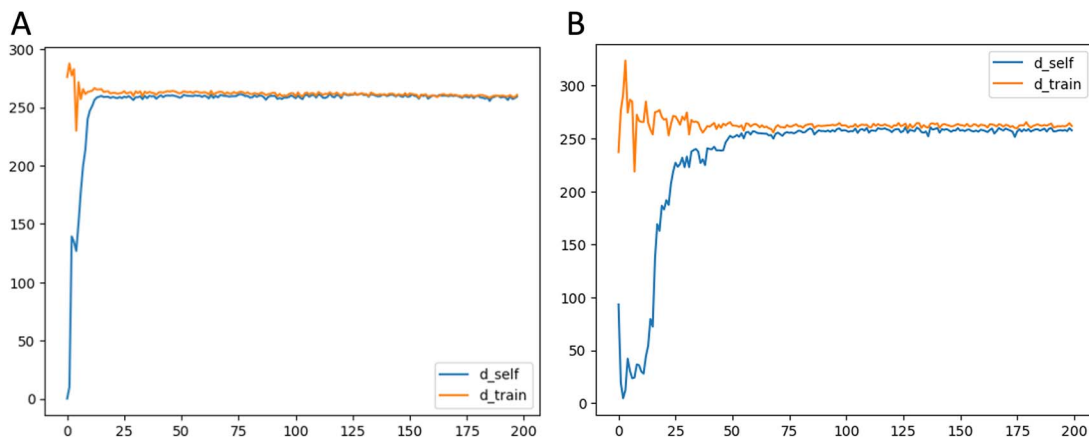


Figure 6. Edit distances for Wasserstein GANs. A. WGAN with KAN linear layer (see Fig. 3B). B. Vanilla GAN (see Fig. 3A).

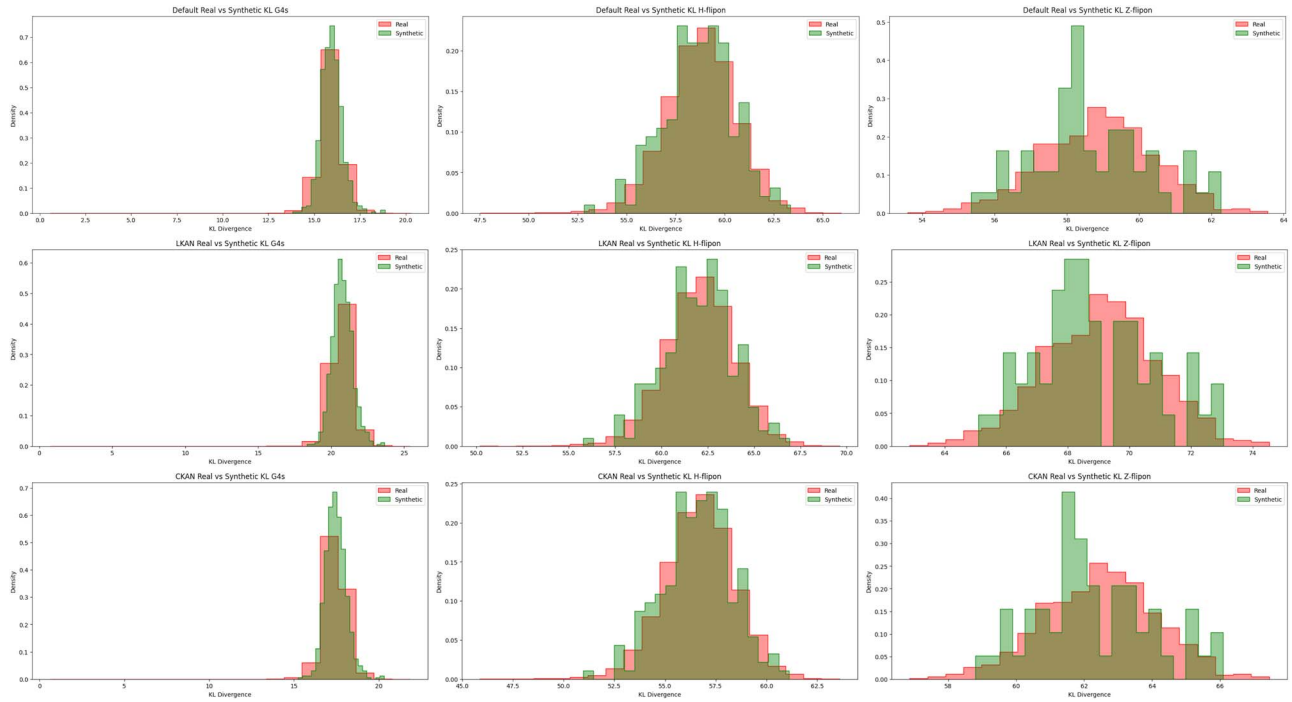


Figure 7. Distributions of Kullback-Leibler divergence for real and synthetic data produced by Baseline Unet (Default) and Unet with LKAN on flipon benchmark datasets.

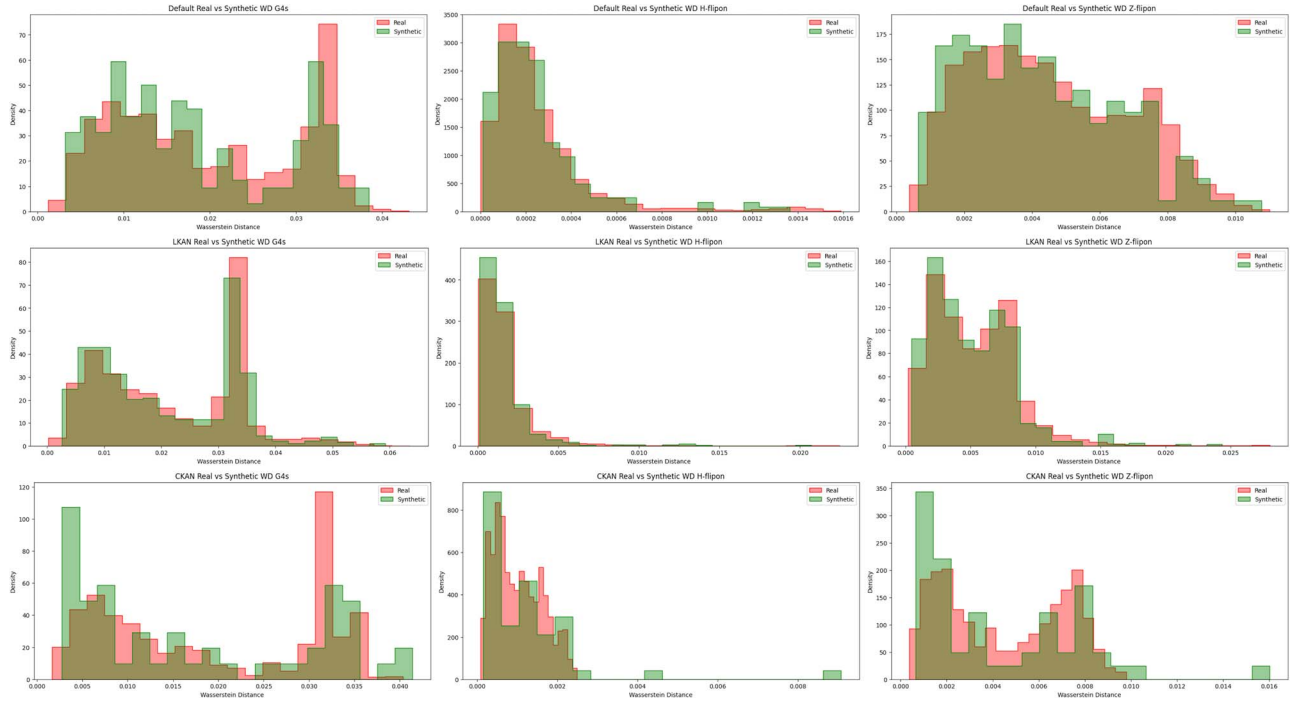


Figure 8. Distributions of Wasserstein Distance for real and synthetic data produced by Baseline Unet (Default) and Unet with LKAN on flipon benchmark datasets.

CNN. MLPs were compared with KANs on tabular data [63] and they observe the same computational behavior: KANs require significantly more (x15) FLOPs than MLPs—see Fig. 2 in [63]. Replacement of B-Splines by Radial Basis Function, which is GPU parallelizable, may significantly enhance the training and tackle CKAN scalability [36].

Current implementations of LKAN lack the same interpretability as original KAN [24] offers: an ability to plot and prune the

latest layers and suggest symbolic expression for outlying dependencies between variables. Potentially, this could bring even more information about inner structure of a biological task of interest.

Research in the KAN field is evolving rapidly as more and more studies emerge and suggest various technical improvements to address challenges related to KAN usage. The future directions may involve training a Transformer-based models with KANs to handle the task of studying latent representations of a language



model's hidden states—embeddings. It was shown [21, 64] that visualization of language model embeddings could enhance biological understanding of model's utility. Of interest is to test more frameworks as state-space models [22], and transformers with GPT-like architecture [65]. In addition, there are multiple advancements in sequence generative modeling like Discrete Diffusion for DNA [66]. Despite our first positive experiments with linear and convolutional KANs, unleashing the potential of KAN for the entire spectrum of genomic tasks requires further extensive research.

## Acknowledgments

The work was supported by the Basic Research Program of the National Research University Higher School of Economics.

## Author contributions

Oleksandr Cherednichenko (Conceptualization, Formal Analysis, Writing—review & editing) and Maria Poptsova (Conceptualization, Formal Analysis, Funding acquisition, Writing—review & editing)

## Supplementary data

Supplementary data is available at *Briefings in Bioinformatics* online.

## Funding

The work has been done within the framework of the Basic Research Program at HSE University, RF.

## Competing interests

No competing interest is declared.

## References

1. Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. *Nat Methods* 2015;**12**: 931–4. <https://doi.org/10.1038/nmeth.3547>
2. Avsec Ž, Agarwal V, Visentin D. et al. Effective gene expression prediction from sequence by integrating long-range interactions. *Nat Methods* 2021;**18**:1196–203. <https://doi.org/10.1038/s41592-021-01252-x>
3. Oubounyt M, Louadi Z, Tayara H. et al. Deepromoter: robust promoter predictor using deep learning. *Front Genet* 2019;**10**:1–9. <https://doi.org/10.3389/fgene.2019.00286>
4. Le NQK, Ho QT, Nguyen TTD. et al. A transformer architecture based on bert and 2d convolutional neural network to identify dna enhancers from sequence information. *Brief Bioinform* 2021;**22**:1–7. <https://doi.org/10.1093/bib/bbab005>
5. Quang D, Xie X. Factornet: A deep learning framework for predicting cell type specific transcription factor binding from nucleotide-resolution sequential data. *Methods* 2019;**160**: 40–7.
6. Yin Q, Wu M, Lv H. et al. Deephistone: a deep learning approach to predicting histone modifications. *BMC Genomics* 2021;**20**: 11–23.
7. Wen W, Zhong J, Zhang Z. et al. Dhica: a deep transformer-based model enables accurate histone imputation from chromatin accessibility. *Brief Bioinform* 2024;**25**:1–11. <https://doi.org/10.1093/bib/bbae459>
8. Beknazarov N, Jin S, Poptsova M. Deep learning approach for predicting functional z-dna regions using omics data. *Sci Rep* 2020;**10**:1–15. <https://doi.org/10.1038/s41598-020-76203-1>
9. Umerenkov D, Herbert A, Konovalov D. et al. Z-flipon variants reveal the many roles of z-dna and z-rna in health and disease. *Life Sci Alliance* 2023;**6**:e202301962. <https://doi.org/10.26508/lsa.202301962>
10. de Sainte Agathe JM, Filser M, Isidor B. et al. Spliceai-visual: a free online tool to improve spliceai splicing variant interpretation. *Hum Genomics* 2023;**17**:7. <https://doi.org/10.1186/s40246-023-00451-1>
11. Sato K, Hamada M. Recent trends in rna informatics: a review of machine learning and deep learning for rna secondary structure prediction and rna drug discovery. *Brief Bioinform* 2023;**24**:1–13. <https://doi.org/10.1093/bib/bbad186>
12. Hwang H, Jeon H, Yeo N. et al. Big data and deep learning for rna biology. *Exp Mol Med* 2024;**56**:1293–321. <https://doi.org/10.1038/s12276-024-01243-w>
13. Shen Z, Zhang Q, Han K. et al. A deep learning model for mapro-tein binding preference prediction based on hierarchical lstm and attention network. *IEEE/ACM Trans Comput Biol Bioinforma* 2020;**19**:753–62. <https://doi.org/10.1109/TCBB.2020.3007544>
14. Franke JKH, Runge F, Köksal R. et al. Rnaformer: a simple yet effective model for homology-aware rna secondary structure prediction. *bioRxiv* 2024:2024.2002.2012.579881. pp 1–24.
15. Georgakilas GK, Grioni A, Liakos KG. et al. Multi-branch convolutional neural network for identification of small non-coding rna genomic loci. *Sci Rep* 2020;**10**:1–10. <https://doi.org/10.1038/s41598-020-66454-3>
16. Chen KM, Wong AK, Troyanskaya OG. A sequence-based global map of regulatory activity for deciphering human genetics. *Nat Genet* 2022;**54**:940949. <https://doi.org/10.1038/s41588-022-01102-2>
17. Shen Z, Bao W, Huang DS. Recurrent neural network for predicting transcription factor binding sites. *Sci Rep* 2018;**8**:15270. <https://doi.org/10.1038/s41598-018-33321-1>
18. Ji Y, Zhou Z, Liu H. et al. Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics* 2021;**37**:2112–20. <https://doi.org/10.1093/bioinformatics/btab083>
19. Zhou Z, Ji Y, Li W. et al. Dnabert-2: Efficient foundation model and benchmark for multi-species genome. In *International Conference on Learning Representations (ICLR, 2024)*, pp 1–23. Vienna, Austria: OpenReview.net, 2024.
20. Dalla-Torre H, Gonzalez L, Revilla JM. et al. Nucleotide Transformer: building and evaluating robust foundation models for human genomics. *Nat Methods* 2025;**22**:287–97. <https://doi.org/10.1038/s41592-024-02523-z>
21. Nguyen E, Poli M, Faizi M. et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in Neural Information Processing Systems* 2023;**36**:43177–201. <https://doi.org/10.48550/ARXIV.2306.15794>
22. Gu A, Dao T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*. 2023. <https://doi.org/10.48550/arXiv.2312.00752>
23. Schiff Y, Kao C-H, Gokaslan A. et al. Caduceus: Bi-directional equivariant long-range dna sequence modeling. In: *Proceedings of the 41nd International Conference on Machine Learning (ICML, 2024)*. 2024, pp 43632–48.

24. ZIMING Liu, YIXUAN Wang, SACHIN Vaidya. et al. Kan: Kolmogorov-Arnold Networks. arXiv preprint arXiv:2408.10205, 2024. <https://doi.org/10.48550/arXiv.2404.19756>.
25. Liu Z, Ma P, Wang Y. et al. Kan 2.0: Kolmogorov-Arnold networks meet science. arXiv preprint arXiv:2404.19756, 2024. <https://doi.org/10.48550/arXiv.2408.10205>
26. Abueidda DW, Pantidis P, Mobasher ME. Deepokan: Deep Operator Network Based on Kolmogorov Arnold Networks for Mechanics Problems. arXiv preprint arXiv:2405.19143, 2024. <https://doi.org/10.48550/arXiv.2405.19143>.
27. Bodner AD, Tepsich AS, Spolski JN, Pourteau S. Convolutional Kolmogorov-Arnold Networks. arXiv preprint arXiv:2406.13155, 2024. <https://doi.org/10.48550/arXiv.2406.13155>
28. Drokin I. Kolmogorov-Arnold convolutions: design principles and empirical studies. arXiv preprint arXiv:2407.01092, 2024. <https://doi.org/10.48550/arXiv.2407.01092>
29. Yang X, Wang X. Kolmogorov-Arnold transformer. arXiv preprint arXiv:2409.10594, 2024. <https://doi.org/10.48550/arXiv.2409.10594>
30. Aghaei AA. Fkan: fractional Kolmogorov-Arnold networks with trainable jacobi basis functions. arXiv preprint arXiv:2406.07456, 2024. <https://doi.org/10.48550/arXiv.2406.07456>
31. Cheon M. Demonstrating the efficacy of Kolmogorov-Arnold networks in vision tasks. arXiv preprint arXiv:2406.14916, 2024. <https://doi.org/10.48550/arXiv.2406.14916>
32. Yu R, Yu W, Wang X. Kan or mlp: a fairer comparison. arXiv preprint arXiv:2407.16674, 2024. <https://doi.org/10.48550/arXiv.2407.16674>
33. Genet R, Inzirillo H. Tkan: temporal Kolmogorov-Arnold networks. arXiv preprint arXiv:2405.07344, 2024. <https://doi.org/10.48550/arXiv.2405.07344>
34. Genet R, Inzirillo H. A temporal Kolmogorov-Arnold transformer for time series forecasting. arXiv preprint arXiv:2406.02486, 2024. <https://doi.org/10.48550/arXiv.2406.02486>
35. Zhou Q, Pei C, Sun F. et al. Kan-ad: time series anomaly detection with kolmogorov-Arnold networks. arXiv preprint arXiv:2411.00278, 2024. <https://doi.org/10.48550/arXiv.2411.00278>
36. Hoang-Thang T. Bsrbf-kan: a combination of b-splines and radial basis functions in Kolmogorov-Arnold networks. arXiv preprint arXiv:2406.11173, 2024. <https://doi.org/10.48550/arXiv.2406.11173>
37. Xu A, Zhang B, Kong S. et al. Effective integration of kan for keyword spotting. arXiv preprint arXiv:2409.08605, 2024. <https://doi.org/10.48550/arXiv.2409.08605>
38. Li R, Li M, Liu W. et al. Gnn-skan: harnessing the power of swallowkan to advance molecular representation learning with gnns. arXiv preprint arXiv:2408.01018, 2024. <https://doi.org/10.48550/arXiv.2408.01018>
39. Nagai Y, Okumura M. Kolmogorov-Arnold networks in molecular dynamics. arXiv preprint arXiv:2407.17774, 2024. <https://doi.org/10.48550/arXiv.2407.17774>
40. Park J-D, Kim K-M, Shin W-Y. Cf-kan: Kolmogorov-Arnold network-based collaborative filtering to mitigate catastrophic forgetting in recommender systems. arXiv preprint arXiv:2409.05878, 2024. <https://doi.org/10.48550/arXiv.2409.05878>
41. Gresova K, Martinek V, Cechak D. et al. Genomic benchmarks: a collection of datasets for genomic sequence classification. BMC Genomic Data 2023;**24**:1–9. <https://doi.org/10.1186/s12863-023-01123-8>
42. Herbert A. Flipons: The Discovery of z-Dna and Soft-Wired Genomes. Boca Raton: CRC Press, 2024. <https://doi.org/10.1201/9781003463535>
43. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Netw 1989;**2**:359–66. [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8)
44. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014. <https://doi.org/10.48550/arXiv.1409.0473>
45. Kolmogorov AN. On the representation of continuous functions of several variables as superpositions of continuous functions of a smaller number of variables. Dokl Akad Nauk 1956;**114**:953–6.
46. Song J, Meng C, Ermon S. Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502, 2020.
47. Arjovsky M, Chintala S, Bottou L. Wasserstein Gan. arXiv preprint arXiv:1701.07875, 2017. <https://doi.org/10.48550/arXiv.1701.07875>.
48. Kouzine F, Wojtowicz D, Baranello L. et al. Permanganate/s1 nuclease footprinting reveals non-b dna structures with regulatory potential across a mammalian genome. Cell Syst 2017;**4**:344–356.e7. <https://doi.org/10.1016/j.cels.2017.01.013>
49. Shin SI, Ham S, Park J. et al. Zdna-forming sites identified by chip-seq are associated with actively transcribed regions in the human genome. DNA Res 2016;**23**:477–86. <https://doi.org/10.1093/dnares/dsw031>
50. Qian SH, Shi MW, Xiong YL. et al. Endoquad: a comprehensive genome-wide experimentally validated endogenous g-quadruplex database. Nucleic Acids Res 2024;**52**:D72–80. <https://doi.org/10.1093/nar/gkad966>
51. Zhang Y, Yang L, Kucheralapati M. et al. Global impact of somatic structural variation on the dna methylome of human cancers. Genome Biol 2019;**20**:209. <https://doi.org/10.1186/s13059-019-1818-9>
52. Chambers VS, Marsico G, Boutell JM. et al. High-throughput sequencing of dna g-quadruplex structures in the human genome. Nat Biotechnol 2015;**33**:877–81. <https://doi.org/10.1038/nbt.3295>
53. Lyu J, Shao R, Kwong Y. et al. Genome-wide mapping of g-quadruplex structures with cut&tag. Nucleic Acids Res 2022;**50**:e13.
54. Penzar D, Nogina D, Noskova E. et al. Legnet: a best-in-class deep learning model for short dna regulatory regions. Bioinformatics 2023;**39**:1–9. <https://doi.org/10.1093/bioinformatics/btad457>
55. Tan M, Le QV. EfficientNetV2: Smaller Models and Faster Training. arXiv preprint arXiv:2104.00298, 2024. <https://doi.org/10.48550/arXiv.2104.00298>
56. Goodfellow IJ, Pouget-Abadie J, Mirza M. et al. Generative adversarial networks. arXiv preprint arXiv:1406.2661, 2014. <https://doi.org/10.48550/arXiv.1406.2661>
57. LF DS, Senan S, Patel ZM. et al. Dna-diffusion: leveraging generative models for controlling chromatin accessibility and gene expression via synthetic regulatory elements. bioRxiv preprint bioRxiv 2024.02.01.5783522024, 2024. <https://doi.org/10.1101/2024.02.01.578352>
58. Cherednichenko O, Poptsova M. Data augmentation with generative models improves detection of non-b dna structures. Comput Biol 2025;**184**:109440. <https://doi.org/10.1016/j.combiomed.2024.109440>
59. He G, Ye J, Hao H. et al. A kan-based hybrid deep neural networks for accurate identification of transcription factor binding sites PREPRINT (Version 1) available at Research Square. 2024, 1–23. <https://doi.org/10.21203/rs.3.rs-4664531/v1>
60. Wen Y, Tran D, Ba J. Batchensemble: An alternative approach to efficient ensemble and lifelong learning. In International Conference on Learning Representations (ICLR 2020), pp 1–19. Ethiopia, 2020. <https://openreview.net/pdf?id=Sk1f1yrYDr>

61. Xu J, Chen Z, Li J. et al. Fourierkan-gcf: Fourier Kolmogorov-Arnold network—an effective and efficient feature transformation for graph collaborative filtering. arXiv preprint arXiv: 2406.01034, 2024. <https://doi.org/10.48550/arXiv.2406.01034>
62. Dosovitskiy A, Beyer L, Kolesnikov A. et al. An image is worth 16x16 words: transformers for image recognition at scale. In: *Proceedings of the 9th International Conference on Learning Representation (ICLR 2021)*. Austria, p: OpenReview.net, 2021.
63. Poeta E, Giobergia F, Pastor E. et al. A benchmarking study of Kolmogorov-Arnold networks on tabular data. In: *2024 IEEE 18th International Conference on Application of Information and Communication Technologies (AICT)*. pp. 1–6. IEEE, 2024.
64. Zhou Z, Wu W, Ho H. et al. Dnabert-s: learning species-aware dna embedding with genome foundation models. arXiv preprint arXiv:2402.08777, 2024. <https://doi.org/10.48550/arXiv.2402.08777>
65. Cui H, Wang C, Maan H. et al. Scgpt: towards building a foundation model for single-cell multi-omics using generative ai. bioRxiv 2023.04.30.5384392023. <https://doi.org/10.1101/2023.04.30.538439>
66. Sarkar A, Tang Z, Zhao C. et al. Designing dna with tunable regulatory activity using discrete diffusion. bioRxiv 2024:2024.2005. 2023.595630. <https://doi.org/10.1101/2024.05.23.595630>