Universitat Pompeu Fabra Barcelona

Flexible Quantum Kolmogorov Arnold Networks

Enhancing quantum KAN expressivity via Fractional order Chebyshev basis and trainable QSVT

Javier González Otero

Bachelor's thesis, 2024-2025

Supervised by: Miguel Ángel González Ballester

Presentation structure

1) Quantum computing
2) Kolmogorov-Arnold Networks: Classical and Quantum
3) Objectives
4) Methods
5) Results
6) Conclusion

This project has been sent to the 9th international conference on
Quantum Techniques in Machine Learning (QTML)

Universitat Pompeu Fabra Barcelona

Quantum computing

Qubit definition

Linear superposition of 2^n orthonormal basis states

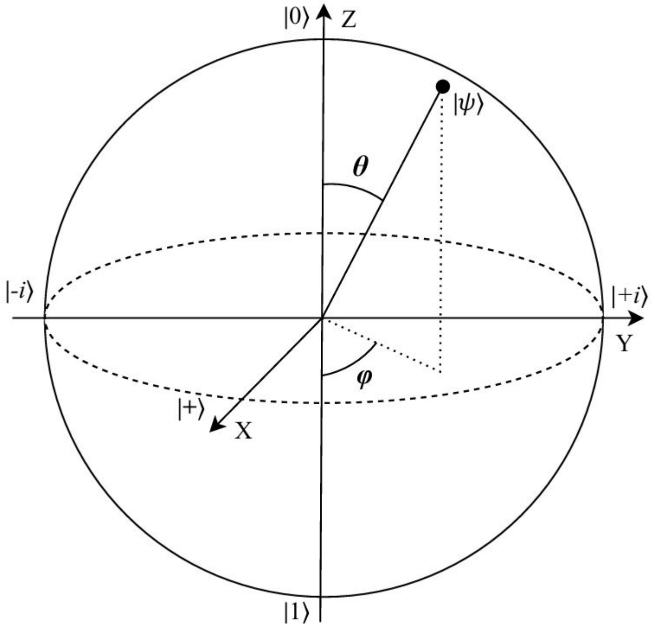
$$|\Psi\rangle = \sum_{j=0}^{2^{n-1}} a_j \, |j\rangle$$

$$a_j \in \mathbb{C}$$

Exponentially large Hilbert space

Parallelism advantage!

1- qubit system
Represented in the surface of the Bloch-Sphere

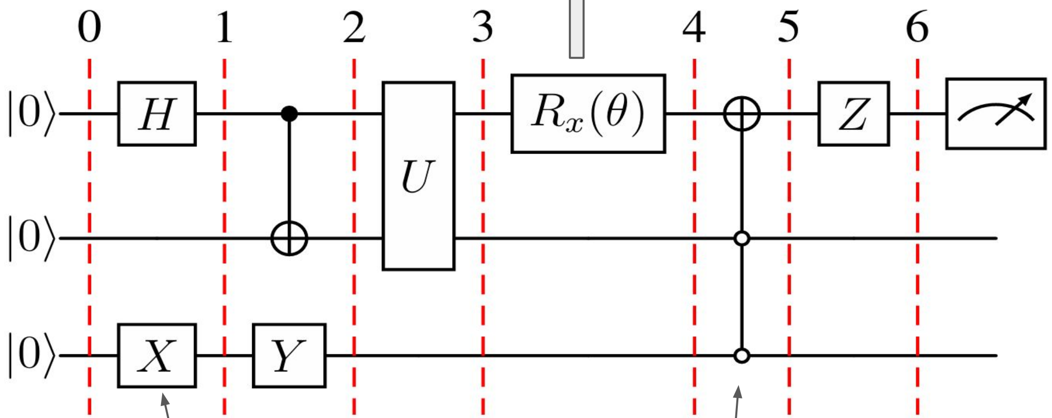


$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\,|1\rangle$$

State transformations - Quantum gates and quantum circuits

Parametrized Quantum Circuit (PQC)

PQCs open the path for Quantum ML (QML)



1-Qubit gates

Rotations around the Bloch-Sphere

n-Qubit gates

May create entanglement

Quantum circuits have a multi-qubit unitary representation

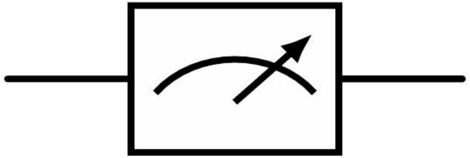
In this case, a 2^3-qubit unitary

Quantum gates are unitary matrices

Universitat Pompeu Fabra Barcelona

Measurements

Probabilistic result



Born rule

Probability amplitudes

$$P(|0\rangle) = |\alpha|^2$$
$$P(|1\rangle) = |\beta|^2$$

QML needs determinism!

1

An observable is defined as a single-qubit unitary matrix $\hat{O}$

2

Prepare $|\Psi\rangle$ through a quantum circuit

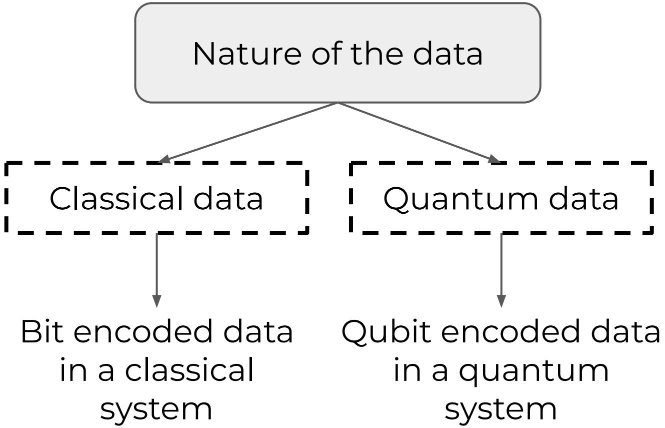
3

Calculate the expected measurement of $\hat{O}$ applied to $|\Psi\rangle$

Universitat Pompeu Fabra Barcelona

Quantum Machine Learning (QML)

Interplay between quantum computing and ML



Types of QML algorithms

	Processing and optimization	
Algorithm type	Classical	Quantum
Classical algorithms	✓	✗
Hybrid algorithms	✓	✓
Quantum algorithms	✗	✓

Universitat Pompeu Fabra Barcelona

Kolmogorov-Arnold Networks: Classical and Quantum

Universitat Pompeu Fabra Barcelona

KAN - Kolmogorov Arnold Networks (classical)

Inspired on the Kolmogorov Arnold representation theorem (KART)

$$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

where $\phi_{q,p} : [0,1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$

φ's can be parametrized! ⇨ High expressive models!

Empirically generalize KART

KAN layer

$$\Phi^{(l)} = \left(\sum_{p=1}^{N^{(l-1)}} \phi_{p,1}^{(l)}(x_p), \ldots, \sum_{p=1}^{N^{(l-1)}} \phi_{p,N^{(l)}}^{(l)}(x_p) \right)$$

KAN

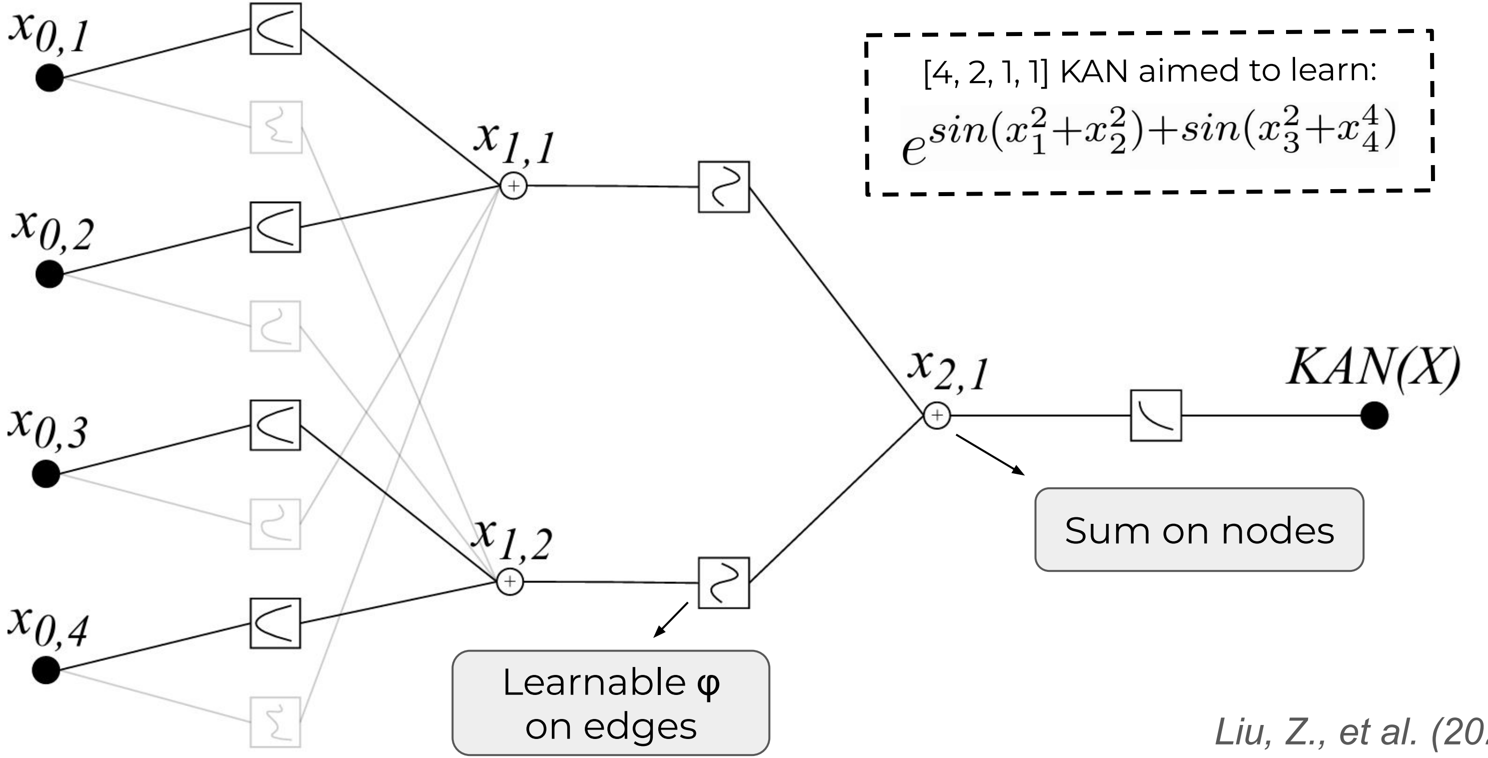
$$\mathrm{KAN}(\mathbf{x}) = \Phi^{(L)} \circ \ldots \circ \Phi^{(1)}(\mathbf{x})$$

Liu, Z., et al. (2024)

KAN - Architecture (classical)

[4, 2, 1, 1] KAN aimed to learn:

$$e^{sin(x_1^2+x_2^2)+sin(x_3^2+x_4^4)}$$



Liu, Z., et al. (2024)

Quantum KANs: State-of-the-art approaches

Name	*Abbreviation*
Quantum Kolmogorov-Arnold Networks	**QKAN**
Variational Quantum Kolmogorov-Arnold Networks	VQKAN
Adaptive Variational Quantum Kolmogorov-Arnold Networks	AVQKAN
Enhanced Variational Quantum Kolmogorov-Arnold Networks	EVQKAN
Quantum Kolmogorov-Arnold Networks by combining Quantum signal processing circuits	-

Universitat Pompeu Fabra Barcelona

QKAN: general idea

Based on Quantum Singular Value Transformation (QSVT) and linear algebra over Block Encodings (BEs)

QSVT forms basis functions which are Linearly combined to form activations

Unitary encoding an N-dimensional input vector

Input BE

$$U_X = \begin{bmatrix} X & * \\ * & * \end{bmatrix}$$

$$X = diag\left(x_1, x_2, \ldots, x_{N=2^n} \right)$$

QSVT + Quantum linear algebra

Output BE

$$U_{QKAN} = \begin{bmatrix} QKAN(X) & * \\ * & * \end{bmatrix}$$

$$QKAN(X) = diag\left(\mathrm{KAN}(\mathbf{x}) = \Phi^{(L)} \circ \ldots \circ \Phi^{(1)}(\mathbf{x})\right)$$

Unitary encoding a K-dimensional output vector

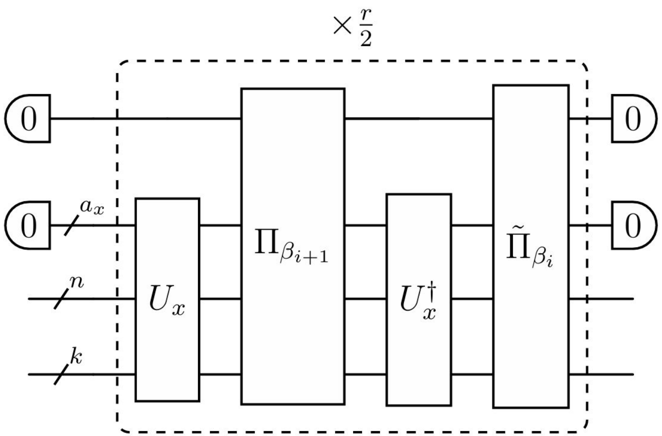
Ivashkov, P., et al. (2024)

Universitat Pompeu Fabra Barcelona

QKAN: QSVT

Applies a polynomial transformation to the singular values of a BE'd matrix

$$\begin{bmatrix} X & * \\ * & * \end{bmatrix} \xrightarrow{\text{QSVT}} \begin{bmatrix} P^{SV}(X) & * \\ * & * \end{bmatrix}$$



QKAN: CHEB-QKAN

Paper proposal

Use Chebyshev polynomials as basis functions

$$T_r(x) = \cos(r\theta)$$

where $x = \cos(\theta)$ and $x \in [-1, 1]$

Easy to implement through the QSVT routine

Chebyshev Approximation is universal

$$f(x) \approx \sum_{r=0}^{d} w_r T_r(x)$$

Ivashkov, P., et al. (2024)

CHEB-QKAN Layer - step 1: Dilate

Objective: Accommodate space for the layer nonlinearities

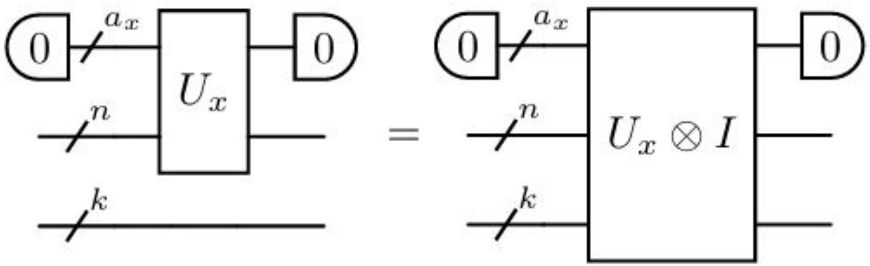
Layer output size

$$\text{DILATE:} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_{BE} \rightarrow \begin{matrix} K \left\{ \begin{matrix} x_1 \\ \vdots \\ x_1 \end{matrix} \right. \\ \\ \\ \\ \end{matrix} \begin{bmatrix} x_1 \\ \vdots \\ x_1 \\ \vdots \\ x_N \\ \vdots \\ x_N \end{bmatrix}_{BE}$$

$\mathbf{x} \in [-1, 1]^N$

Layer input vector

Layer input size



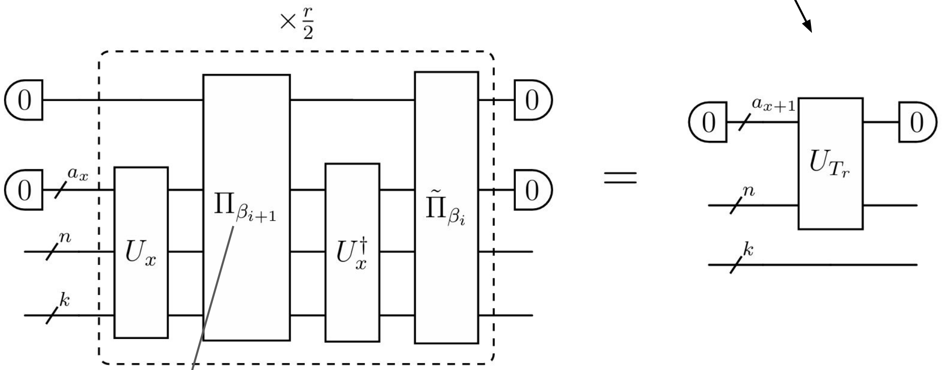
Ivashkov, P., et al. (2024)

CHEB-QKAN Layer - step 2: Cheb

QSVT

One for each degree

$$\text{CHEB:} \begin{bmatrix} x_1 \\ \vdots \\ x_1 \\ \vdots \\ x_N \\ \vdots \\ x_N \end{bmatrix}_{BE} \rightarrow \begin{bmatrix} T_r(x_1) \\ \vdots \\ T_r(x_1) \\ \vdots \\ T_r(x_N) \\ \vdots \\ T_r(x_N) \end{bmatrix}_{BE}$$



Objective: Form basis functions

$$\Pi_\beta = \begin{bmatrix} e^{-i\beta} & 0 & 0 & 0 \\ 0 & e^{i\beta} & 0 & 0 \\ 0 & 0 & e^{i\beta} & 0 \\ 0 & 0 & 0 & e^{i\beta} \end{bmatrix}$$

where $\beta_1 = (1-r)\frac{\pi}{2}$ and $\beta_{i>1} = \frac{\pi}{2}$ with $i \in [1, r]$

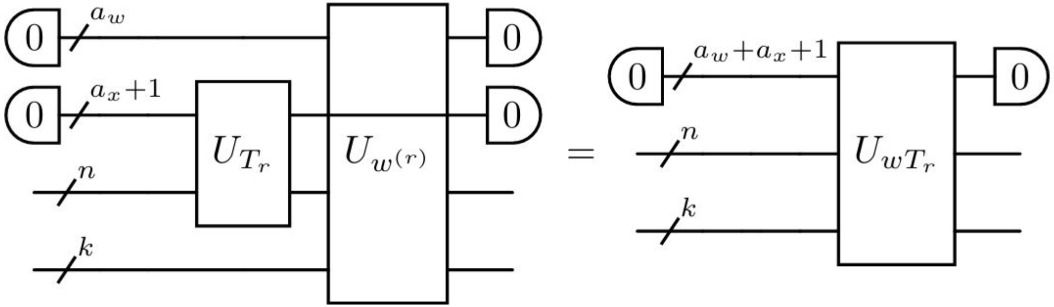
$$U_{T_0} = \begin{bmatrix} T_0(x_1) \\ \vdots \\ T_0(x_1) \\ \vdots \\ T_0(x_N) \\ \vdots \\ T_0(x_N) \end{bmatrix}_{BE} \quad U_{T_1} = \begin{bmatrix} T_1(x_1) \\ \vdots \\ T_1(x_1) \\ \vdots \\ T_1(x_N) \\ \vdots \\ T_1(x_N) \end{bmatrix}_{BE} \quad \dots \quad U_{T_r} = \begin{bmatrix} T_r(x_1) \\ \vdots \\ T_r(x_1) \\ \vdots \\ T_r(x_N) \\ \vdots \\ T_r(x_N) \end{bmatrix}_{BE}$$

Ivashkov, P., et al. (2024)

Universitat Pompeu Fabra Barcelona

CHEB-QKAN Layer - step 3: Mul

$$\text{MUL:}\quad \begin{bmatrix} T_r(x_1) \\ \vdots \\ T_r(x_1) \\ \vdots \\ T_r(x_N) \\ \vdots \\ T_r(x_N) \end{bmatrix}_{BE} \rightarrow \begin{bmatrix} w_{11}^{(r)} T_r(x_1) \\ \vdots \\ w_{1K}^{(r)} T_r(x_1) \\ \vdots \\ w_{N1}^{(r)} T_r(x_N) \\ \vdots \\ w_{NK}^{(r)} T_r(x_N) \end{bmatrix}_{BE}$$



Objective: Form learnable Chebyshev approximation coefficients

Chebyshev approx order

d weight matrices for every NxK activation function

Encoded as BEs

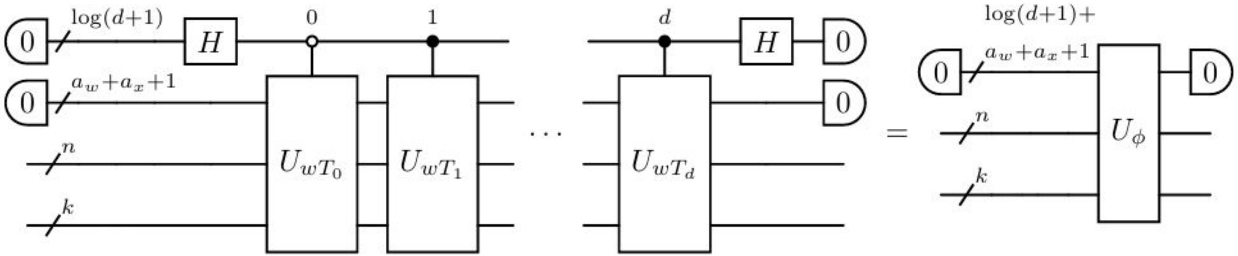
Ivashkov, P., et al. (2024)

Universitat Pompeu Fabra Barcelona

CHEB-QKAN Layer - step 4: LCU

Activation function between p-th input node and q-th output node

$$\text{LCU:}\quad \begin{bmatrix} w_{11}^{(r)}T_r(x_1) \\ \vdots \\ w_{1K}^{(r)}T_r(x_1) \\ \vdots \\ w_{N1}^{(r)}T_r(x_N) \\ \vdots \\ w_{NK}^{(r)}T_r(x_N) \end{bmatrix}_{BE} \rightarrow \begin{bmatrix} \phi_{11}(x_1) \\ \vdots \\ \phi_{1K}(x_1) \\ \vdots \\ \phi_{N1}(x_N) \\ \vdots \\ \phi_{NK}(x_N) \end{bmatrix}_{BE}$$



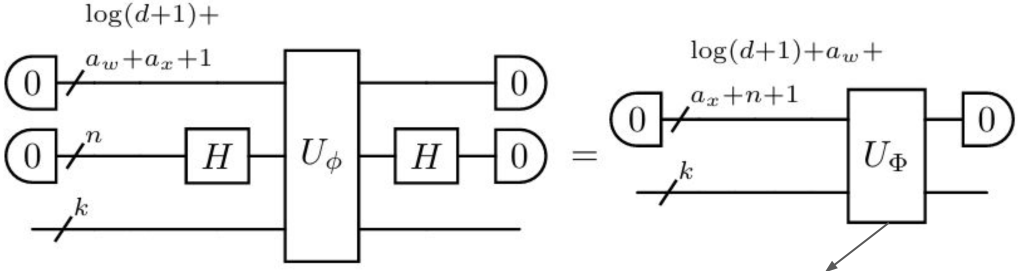
$$\phi_{pq}(x) := \frac{1}{d+1}\sum_{r=0}^{d} w_{pq}^{(r)}T_r(x_p)$$

Objective: Form the NxK activation functions

Ivashkov, P., et al. (2024)

CHEB-QKAN Layer - step 5: SUM (Final step!)

$$\text{SUM:} \begin{bmatrix} \phi_{11}(x_1) \\ \vdots \\ \phi_{1K}(x_1) \\ \vdots \\ \phi_{N1}(x_N) \\ \vdots \\ \phi_{NK}(x_N) \end{bmatrix}_{BE} \rightarrow \begin{bmatrix} \frac{1}{N}\sum_{p=1}^{N} \phi_{p1}(x_p) \\ \vdots \\ \frac{1}{N}\sum_{p=1}^{N} \phi_{pK}(x_p) \end{bmatrix}_{BE}$$



layer output BE!!

Objective: Sum all activation functions that go to the same output neuron

To form **multi-layer** QKAN models:

$$U_\Phi \longrightarrow U_x$$

Use layer output BE as input BE for the next layer

Forms

U_{CHEB} CHEB - QKAN

Ivashkov, P., et al. (2024)

Objectives

Objectives of this work

Current limitations	Objectives
CHEB-QKAN may be limited in expressive power	Increase the expressivity of QKAN-based models
Lack of empirical research on QKAN-based models	Provide empirical research on QKAN-based models for the classical-to-classical setting

Methods

Universitat Pompeu Fabra Barcelona

Generalized Fractional order of the Chebyshev Functions (GFCF) transformation

GFCFs

Fractional degree

$$ {}_{\eta}FT_r^{\alpha}(x) = T_r\left(1 - 2\left(\frac{x}{\eta}\right)^{\alpha}\right) $$

where $\alpha, \eta \in \mathbb{R}_+$ and $x \in [0, \eta]$

Defines a universal basis of functions well suited for representing fractional growths

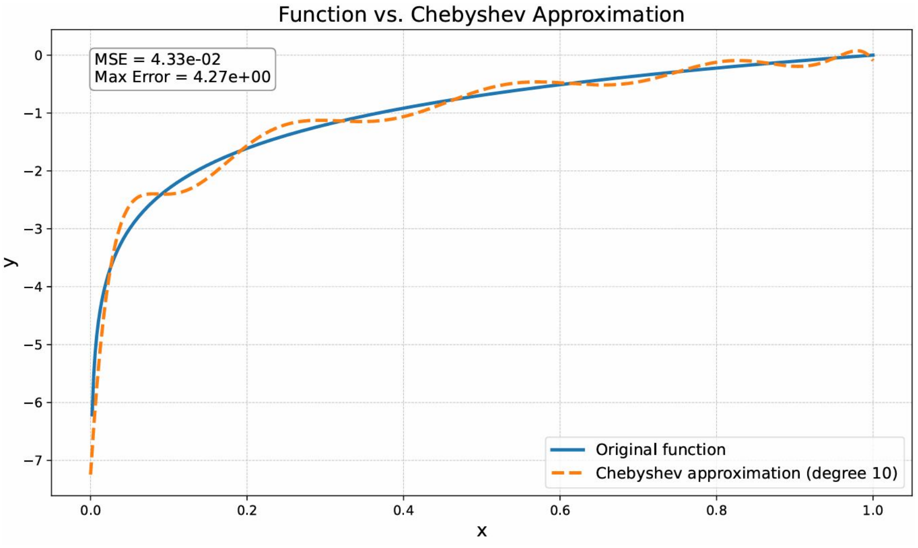
GFCF approximation

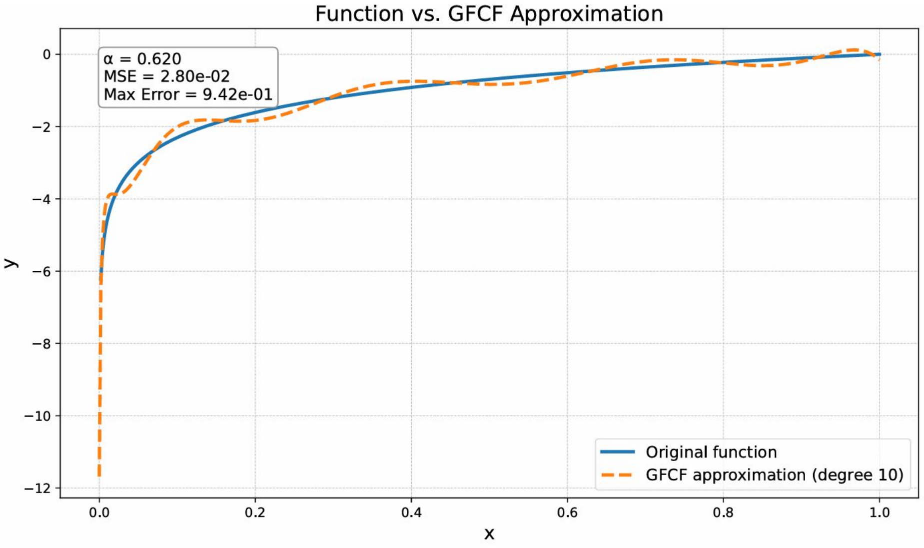
$$ f(x) \approx \sum_{r=0}^{d} {}_{\eta}FT_r^{\alpha}(x) a_r $$

Can converge faster than Chebyshev approximation for some functions

Kourosh, P. and Mehdi, D (2017)

Approximation of log(x) - Chebyshev vs GFCF





Lower MSE!

GFCFs for QKAN - classical processing step

Assuming a normalized classical input in [0, η]

Apply GFCF transformation prior to QKAN ingestion

$$\mathbf{x}_{\text{norm}} \in [0, \eta]^N \quad \Longrightarrow \quad \mathbf{x}_{gfcf} = 1 - 2\left(\frac{\mathbf{x}_{\text{norm}}}{\eta}\right)^{\alpha}$$

where $\alpha, \eta \in \mathbb{R}_{>0}$

α

Can be made trainable to increase expressivity → Basis functions adapt themselves to fractional growths

Flexible QKAN (Flex-QKAN)

Idea

Parametrize QSVT angles

$$\cancel{\text{where } \beta_1 = (1-r)\frac{\pi}{2} \text{ and } \beta_{i>1} = \frac{\pi}{2} \text{ with } i \in [1, r]}$$

$$\beta_1, \ldots, \beta_r \longrightarrow \boxed{\text{Trainable!}}$$

Why?

Learnable polynomial basis functions

$$F_1^{\beta_1}(x) \ldots F_d^{\beta_1 \ldots \beta_d}(x)$$

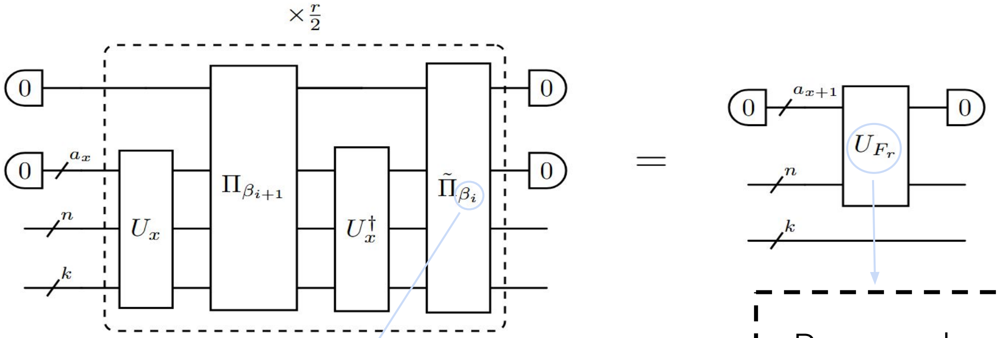
Implications

Augmented expressivity

Choose the best polynomial basis for each specific problem

Flex-Step and U_{FLEX}

Flex-Step



Trainable

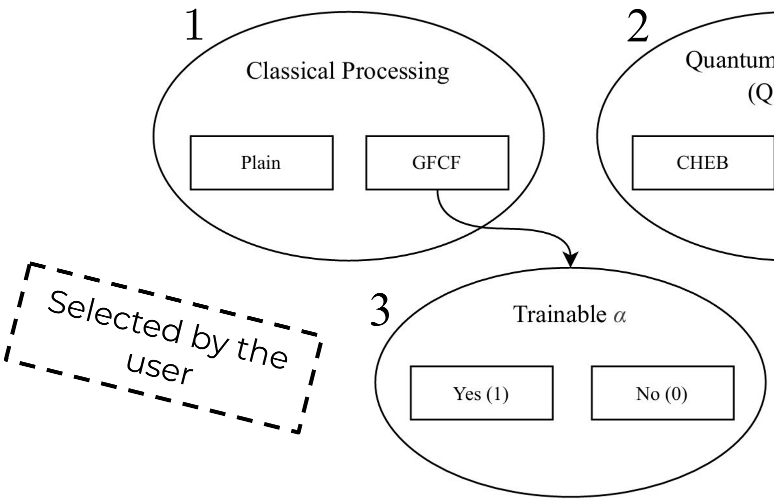
Renamed as U_{Fr}

Multi-layer Flex-QKAN

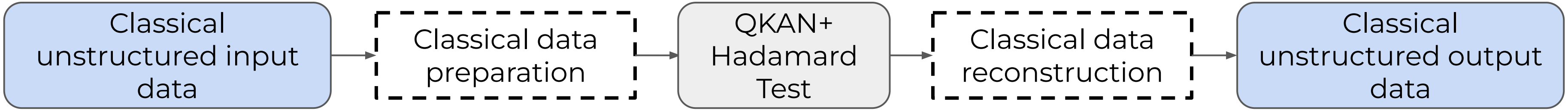
U_{FLEX}

GFCF + U_{FLEX} to increase even more the expressivity!

Classical-to-Classical QKAN framework: CCQKAN



Model Name
Plain-CHEB
Plain-Flex
GFCF-CHEB-0
GFCF-CHEB-1
GFCF-Flex-0
GFCF-Flex-1



$\mathbf{x}_{in} \in \mathbb{R}^N$

Models pipeline

$\mathbf{x}_{out} \in \mathbb{R}^K$

Universitat Pompeu Fabra Barcelona

CCQKAN models initialization

Implemented as a Python class:

1) Initialization parameters
Network structure
Degree of approximation
GFCF flag
α and η
Trainable α flag
Flex flag
Input + Output domains

2) Trainable Parameters initialization:

Parameter	**Initialization**
Weights	Random or fixed 0.5
α	User's choice
QSVT angles	Chebyshev
Output domain	User's choice

CCQKAN models end-to-end pipeline I

0

Classical unstructured input data ingestion

$\mathbf{x}_{in} \in \mathbb{R}^N$

1

Classical data preparation

Plain models

$$\underbrace{\mathbf{x}_{in} \longrightarrow \mathbf{x}_{plain} \in [-1,1]^N}_{\text{min-max norm}} \longrightarrow \mathbf{X}_{plain} = \begin{bmatrix} x_{\text{plain}_1} & 0 & \cdots & 0 \\ 0 & x_{\text{plain}_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_{\text{plain}_N} \end{bmatrix}$$

GFCF models

$$\underbrace{\mathbf{x}_{in} \longrightarrow \mathbf{x}_{\text{norm}}}_{\text{min-max norm}} \in [0,\eta]^N \longrightarrow \mathbf{x}_{gfcf} = 1 - 2\left(\frac{\mathbf{x}_{\text{norm}}}{\eta}\right)^{\alpha} \longrightarrow \mathbf{X}_{gfcf} = \begin{bmatrix} x_{\text{gfcf}_1} & 0 & \cdots & 0 \\ 0 & x_{\text{gfcf}_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_{\text{gfcf}_N} \end{bmatrix}$$

where $\alpha, \eta \in \mathbb{R}_{>0}$

CCQKAN models end-to-end pipeline II

2

Quantum processing

if

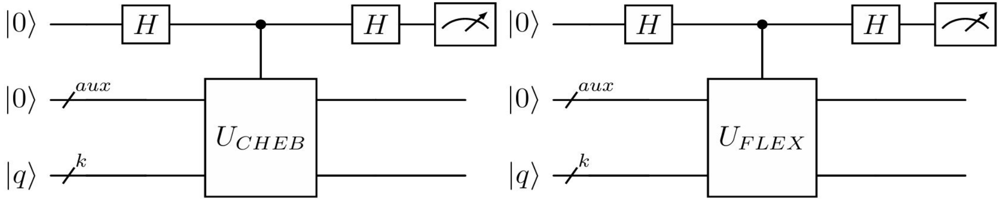
|q> = |5> = |101>

then

X_{h5}

For q in 1...K

PennyLane qnode receiving diagonal input matrix



Extract

X_{hq}

$$\mathbf{x}_h \in [-1, 1]^{K_{out}}$$

Universitat Pompeu Fabra Barcelona

CCQKAN models end-to-end pipeline III

3

Classical data reconstruction

$\mathbf{x}_h \in [-1, 1]^{K_{out}} \rightarrow$ Inverse step 1 normalization $\rightarrow \mathbf{x}_{out} \in \mathbb{R}^K$

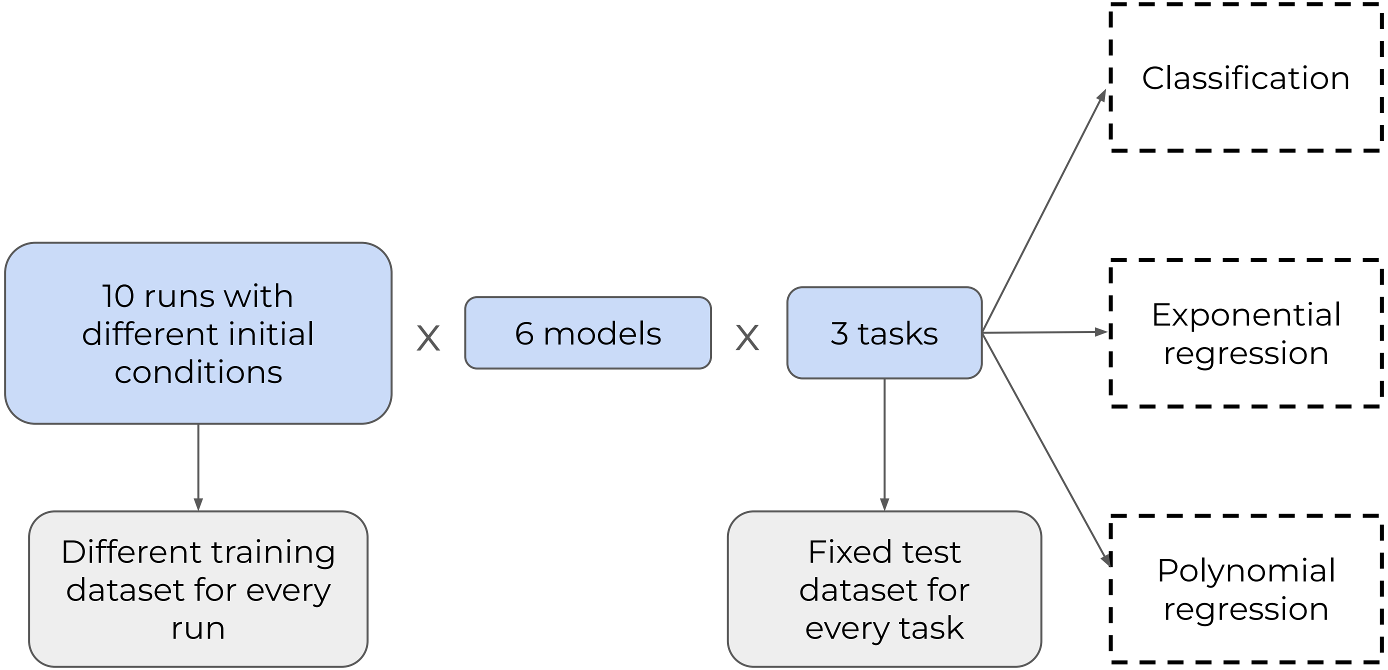
Learnable output bounds

Model classical unstructured output

Results

Universitat Pompeu Fabra Barcelona

Experimental setup

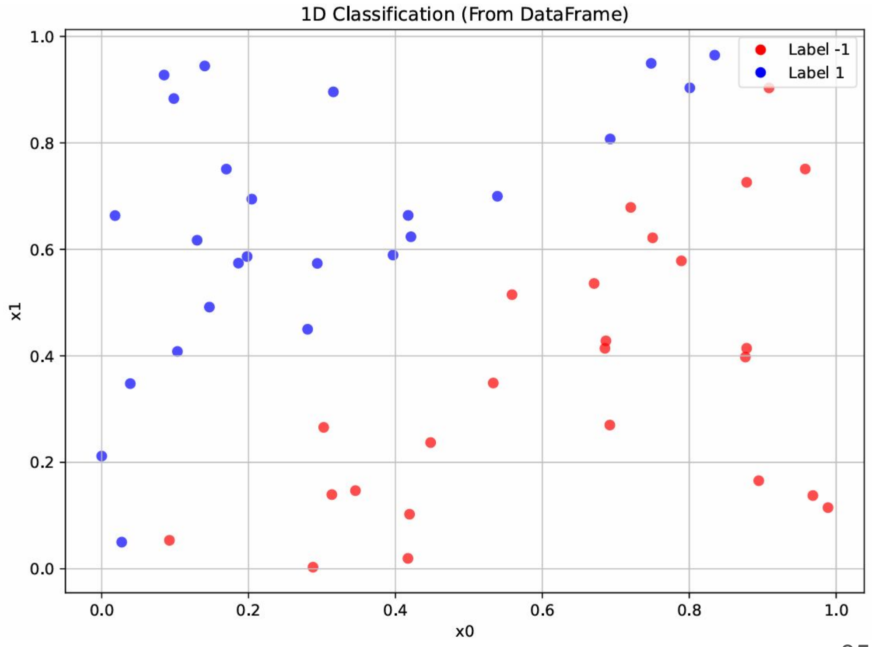


Classification task - description

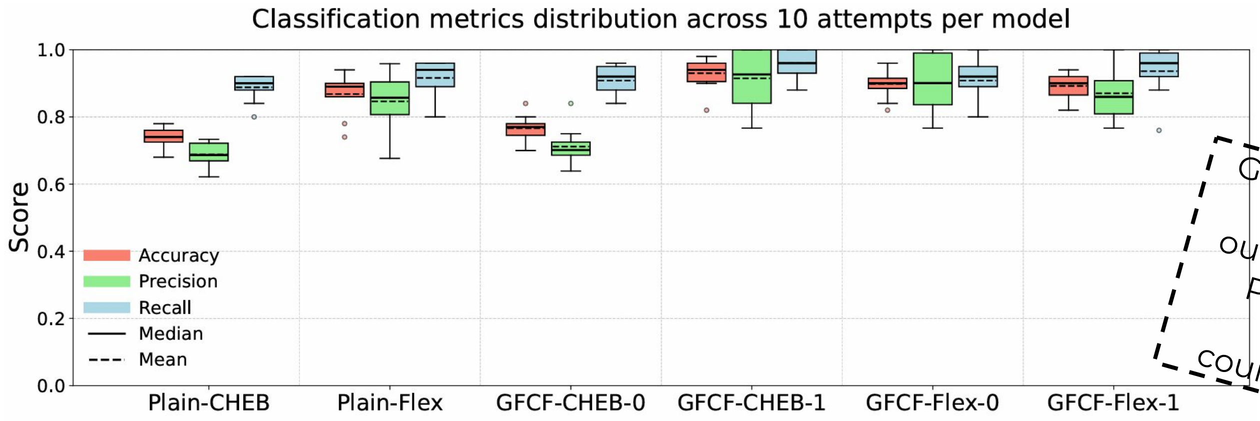
Trained as 1-d regression

Training datasets size	45
Test dataset size	50
QKAN	[1,2,1] d=2
Optimizer	Adam, 0.3
Epochs	20
Training loss	MSE
Evaluation	Accuracy, Precision, Recall

Test dataset

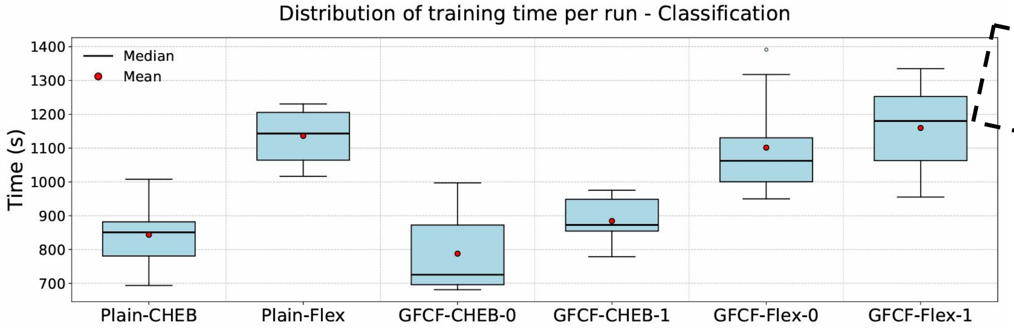


Classification task - results



GFCF and Flex methods outperformed Plain and CHEB counterparts

GFCF fixed α has the same training time as Plain!



+ expressivity
- trainability

Exponential regression task - description

Training datasets size	100
Test dataset size	50
QKAN	[4,1] d=2
Optimizer	Adam, 0.3
Epochs	55
Training loss	MSE
Evaluation	Sum Abs. Distances

$$SAD = \sum_{i=1}^{i=50} |f^{aim}(\mathbf{x})_i - \hat{f}^{aim}(\mathbf{x})_i|$$

Datasets generator

$$f_{exp}^{aim}(\mathbf{x}) = \exp\left(\sin(x_0^2 + x_1^2) + \sin(x_2^2 + x_3^2)\right)$$

Input domain: $[-1,1]^4$

Test dataset: 5 samples

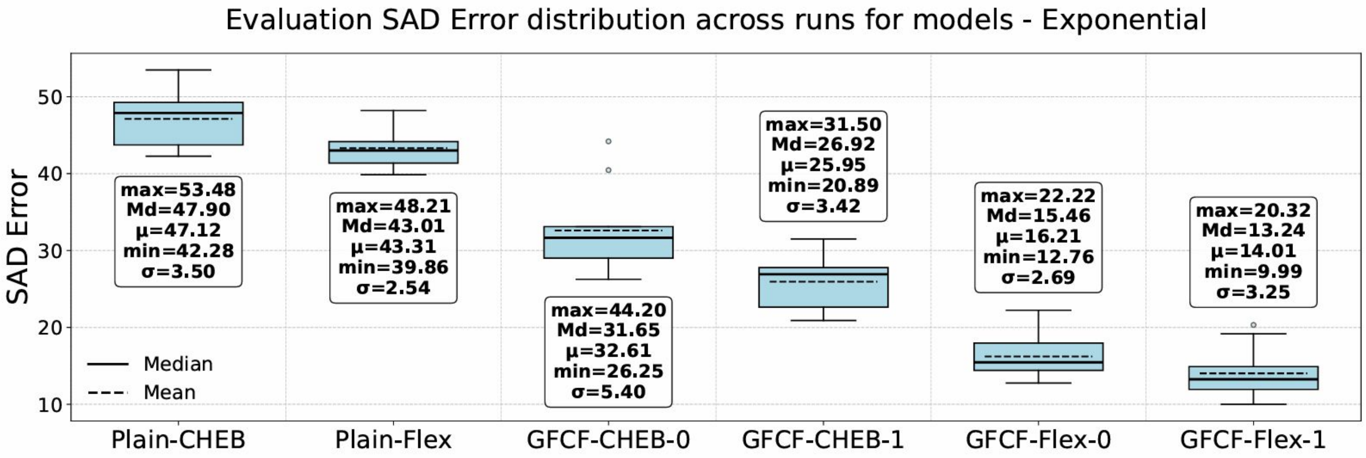
x_0	x_1	x_2	x_3	$target$
0.10	0.14	0.36	-0.70	1.84
0.43	-0.12	-0.46	0.74	2.42
0.21	0.98	0.47	-0.68	4.33
0.09	-0.80	0.92	0.23	4.00
-0.15	-0.58	-0.50	-0.75	2.96

Target domain: [1.08, ,7.18]

Contextualize distance based eval.

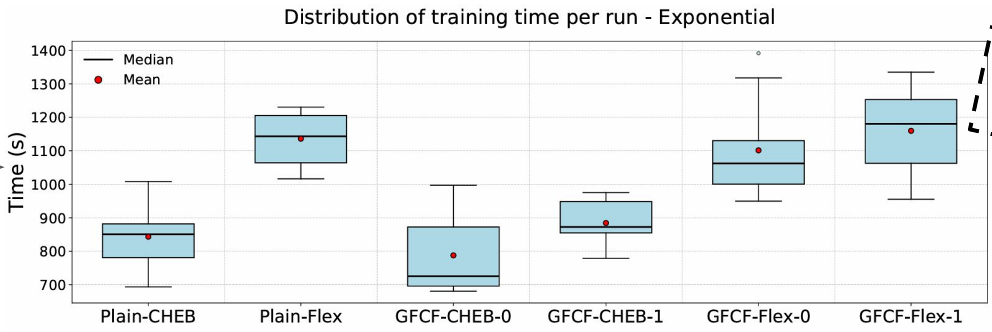
Universitat Pompeu Fabra Barcelona

Exponential regression task - results



GFCF fixed α clearly outperformed Plain models

With no trainability cost!!



+ expressivity
- trainability

Polynomial regression task - description

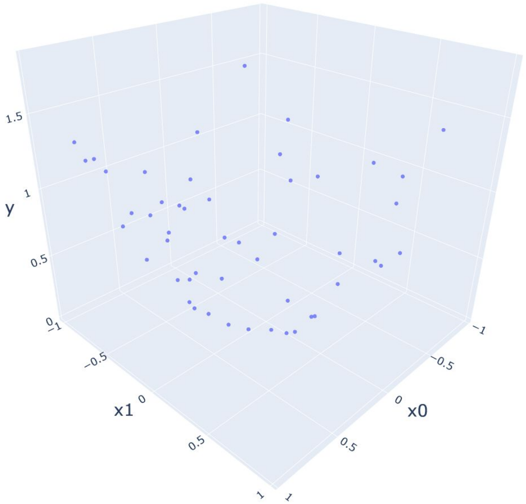
Training datasets size	100
Test dataset size	50
QKAN	[2,1] d=2
Optimizer	Adam, 0.3
Epochs	55
Training loss	MSE
Evaluation	Sum Abs. Distances

$$SAD = \sum_{i=1}^{i=50} |f^{aim}(\mathbf{x})_i - \hat{f}^{aim}(\mathbf{x})_i|$$

Datasets generator

$$f^{aim}(\mathbf{x}) = x_0^2 + x_1^2$$

Test dataset



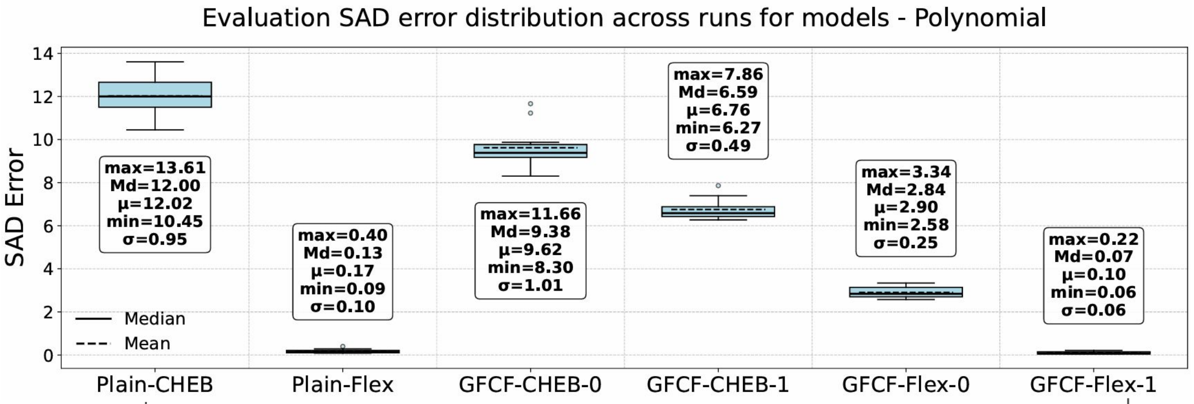
Input domain: $[-1,1]^2$

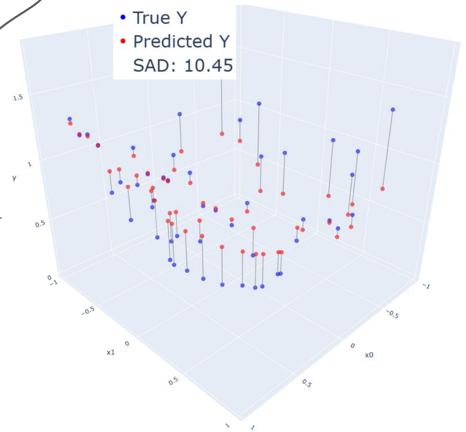
Target domain: [0.29, 1.82]

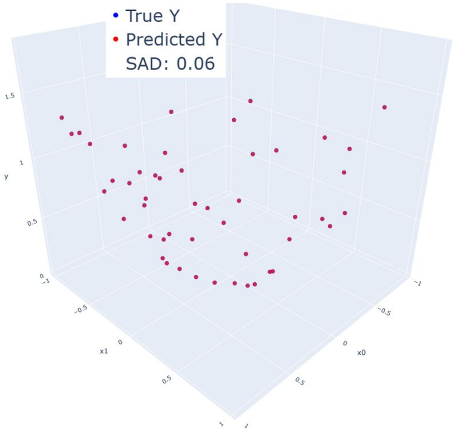
Contextualize distance based eval.

Polynomial regression task - results

Flex models performed surprisingly good here







Discussion

Expressivity is augmented

GFCF with fixed α has no trainability cost

GFCF and Flex outperformed Plain and CHEB

GFCF learnable α + Flex QKAN

Only 1 extra parameter

$\mathcal{O}(d^2)$ extra parameters

Use it if fractional growth is suspected

Once trained, forward pass is equally fast as other models. Use them!!

Conclusion

Universitat Pompeu Fabra Barcelona

Contributions

Augmented expressivity for QKAN-based methods

GFCF transformation

$$\mathbf{x}_{gfcf} = 1 - 2\left(\frac{\mathbf{x}_{\text{norm}}}{\eta}\right)^{\alpha}$$

Prior to BE

α learnable -> + expressivity - trainability

Improved Plain results

Flex-QKAN

Learnable polynomial basis functions

$$F_1^{\beta_1}(x) \ldots F_d^{\beta_1 \ldots \beta_d}(x)$$

+ expressivity - trainability

Improved CHEB results

Empirical research on QKAN-based models

CCQKAN framework

6 models
Plain-CHEB
Plain-Flex
GFCF-CHEB-0
GFCF-CHEB-1
GFCF-Flex-0
GFCF-Flex-1

Open source classical-to-classical QKAN based framework

Baseline established for future research

Future work

Polynomial basis functions

B-Splines

+ Locally adaptive

Implementable through QSVT

Options:

Make CCQKAN hardware compatible

1) Use FABLE to BE

2) Encode classical input as state amplitudes

Transform to BEs

Apply GFCFs on every layer

Solve the following problem:

Given a BE Ux that encodes a diagonal matrix X of a vector x ∊ RN, find a BE Uz of the diagonal matrix Z such that the diagonal values of Z are the diagonal elements of X transformed by $z = 1-2(x/\eta)^{\alpha}$

Arbitrary power to *Ux*

LCU step

- Increase expressivity on every layer

Universitat Pompeu Fabra Barcelona

Thank you

Contact details:

javier.gonzalez10@estudiant.upf.edu