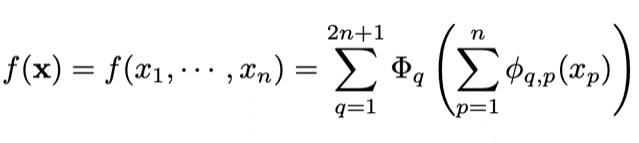
A KAN network is founded on the Kolmogorov-Arnold theorem. The original formula of Kolmogorov-Arnold theorem comes below:

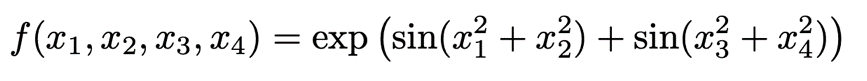


In the Kolmogov-Arnold theorem:

* the inner functions form a KAN layer with nin = n and nout = 2n + 1
* the outer functions form a KAN layer with nin = 2n + 1 and nout = 1.

Thus, the Kolmogorov-Arnold representations are simply compositions of two KAN layers, resulting in only two-layer non-linearities and a small number of terms (2n + 1) in the hidden layer. This 2-layer KAN can be represented with the notation [n, 2n + 1, 1], or simply referred to as a 2-layer width-(2n + 1) KAN.

The original Kolmogorov-Arnold theorem formula opens new possibilities in machine learning applications. However, it is limited in that the non-linearities it represents may be non-smooth. Therefore, **generalizing** this idea to deeper representations seems inevitable to achieve the benefits of smoother activations. For example, the following 4-variable function can be smoothly represented by a [4, 2, 1, 1] KAN which is 3-Layer:



The total number of parameters for a network of depth L, layers of equal width (n0 =n1 =···=nL =N), with each spline of order k (usually k = 3) on G intervals (for G + 1 grid points) is:

O(N2L(G + k)) ∼ O(N2LG)

In contrast, an MLP with depth *L* and width *N* requires O(N2L) parameters, which seems more efficient than KAN. Fortunately, KANs generally need much smaller N than MLPs, which not only reduces the parameter count but also improves generalization. For 1D problems, taking N=L=1 makes the KAN network equivalent to a **spline approximation**.

Given that all functions to be learned are univariate, each 1D function can be parameterized as a B-spline curve with learnable coefficients for the local B-spline basis functions. Generally, spline(x) is represented as a linear combination of B-splines.

KANs properties:

1. KAN relies on a finite set of grid points. Additionally, similar to splines that can be arbitrarily tuned for accuracy, KAN can also be fine-tuned. Initially, KAN can be trained with a few parameters and later extended to include more parameters by refining its spline grids as needed. This feature distinguishes KAN from MLPs, which require independent training for models of different sizes.
2. KAN can be represented by a symbolic formula. For instance, if we know a dataset is generated using a symbolic formula we can deduce that a specific KAN can express the function. However, in practice, we lack this prior information, so it would be beneficial to begin with a sufficiently large KAN, train it with sparsity regularization, and then apply pruning. Pruned KANs are significantly more interpretable than their non-pruned counterparts.
3. KANs have the advantage of being interpretable through mathematical functions (symbolic functions). These compact representations allow for the decomposition of a high-dimensional lookup table into several 1D lookup tables, which can significantly reduce memory usage, with only a minimal overhead of a few additional computations during inference.

Considering the aforementioned general information, the project tasks are defined as follows:

**A time window of 8-16 weeks:**

**When the optimal KAN shape is already constructed**

**Student1:** f(x,y)=exp(sin(πx)+y2) **à** can be exactly represented by a [2,1,1] KAN

**Student2:** f (x, y) = xy **à** can be exactly represented by a [2,2,1] KAN

Grade-3:

1. Train the KAN by increasing grid points.
2. Train MLPs with different depths and widths as base-lines. Both MLPs and KANs should be trained for the same number of steps in total.
3. **LBFGS** is generally used as the standard KAN optimization algorithm. In KAN’s original paper it is mentioned that for a specific G (G=1000) they experience a slow-down in training. Investigate for which grid point G using LBFGS the training phase slows down
4. Find an optimal step size within the already found maximum grid point
5. Study the use of another optimization algorithm and compare its result with LBFGS result
6. Plot test RMSE as a function of the number of parameters for KANs and MLPs
7. Compare the results and explain the differences and similarities.
8. Describe the developed/trained KAN network without any use of already built-in libraries and functions. In other words, in this task you have to develop the same developed KAN network using underlying mathematical grounds/expressions. This task, in itself, serves as a means of evaluating your understanding about how the KAN should be later on implemented in any of the hardware description languages.
9. Write a report elaborating on all steps above.

Grade 4 and 5:

For the steps below, new functions or datasets will be provided by the supervisors.

1. Find and train KANs both with and without pruning:
   * *KANs without pruning*: KANs with width=5 and depths sweeping {2,3,4,5,6}.
   * *KAN with pruning*: use the sparsification (λ = 10−3) and pruning to obtain a smaller KAN pruned from a fixed-shape KAN. Initialize the KAN with G=3, and then increase the grid points every 200 steps to cover G = {3, 5, 10, 20, 50, 100, 1000}.
2. Transferring the formula representation to a lookup table representation.
3. Implement the KAN in any Hardware Description Language (HDL) with which you are comfortable
4. Demonstrate the functionality of hardware implementation with hardware simulations
5. Synthesize the hardware and report area, gate counts, power, clk frequency
6. Optimize the hardware implemented design if there is a room for it
7. Complete your report with the outcomes of the aforementioned follow-up steps. A grade 4 or 5 is dependent on the evaluation of course responsible and examiner on the quality of the results and etc.