Neil Kachappilly, Greg Ou, Abhishek Chaudhary

1. **Project Title**

   Optimizing Kolmogorov-Arnold Networks: Accelerating Training and Inference in KANs

2. **Goal/Objective**

   Our primary objective is to advance Kolmogorov-Arnold Networks (KANs) as a viable alternative to traditional MLP based neural network implementations by improving its efficiency. Research has shown that KANs provide greater transparency and interpretability due to their learned activation functions. It has also been shown that KANs can converge faster than MLPs with greater accuracy in certain tasks despite using fewer parameters. The primary drawback of KANs today is their slower inference and training times when compared to MLPs. Our contribution is to optimize KAN training and inference, pushing performance closer towards the SoTA for MLPs so that KANs become more practical for deep learning.

3. **Challenges**

   KANs are relatively new in the space and very new to us, resulting in a major challenge up front in learning the novel architecture before we can seek to improve it. Furthermore, we expect lack of documentation and general online experience regarding potential debugging. Modifying the underlying framework will require a deep theoretical understanding along with extensive experimentation, and as a result, time will also be a challenging constraint as we seek to achieve our goals within 8 weeks.

4. **Approach and Performance Optimization Techniques to be used**

   a. Quantization

      i. Post-Training Quantization (PTQ): This would allow us to track the performance of the model and level of data-precision where the effects of

      ii. Quantization-Aware Training (QAT): This will allow us to compare the training times for MLPs and quantized KANs

   b. Model Pruning

      i. Implement sparse inference kernels, enforcing sparse matrix multiplication with specialized libraries.

      ii. Use regularization to zero out individual weights to make computations sparse.

   c. Mixed Precision training - Use FP16/BF16 instead of FP32 to reduce memory usage and increase speed.

   d. Parallelize B-Spline Calculations on GPU - Build on work from Coffman and Chen's MatrixKAN paper to parallelize computation during training and inference. Implement a custom CUDA kernel to further accelerate these computations.

5. **Implementation details: hardware (compute GPU/TPU, etc., cloud-based, edge devices), software (framework, existing code to reuse), dataset**

We will use the GCP VM with 1 A100 NVIDIA GPU to train our models and run inference. The existing code from the original paper's authors provides Pytorch implementations of the KAN layer architecture and B-spline functionality to define the activations (https://github.com/KindXiaoming/pykan/blob/master/kan). Following, they also provide the MLP model they used to compare to the KAN. The original implementation of KANs focuses on synthetic datasets from math and physics domains with one such simple example being $f(x, y) = exp(sin(\pi x + y^2)$ and RMSE for model performance. We will focus on the continued usage of symbolic formula-based datasets for KANs while benchmarking the aforementioned optimization techniques' impact on training/inference times and model RMSE.
Optional extension - Extending to image-based datasets either using MLPs' benchmarking metrics as a baseline or switching over to CNNs and ConvKANS (https://arxiv.org/abs/2406.13155) would give us access to a larger breadth of accuracy metrics for these well benchmarked datasets. Datasets that are well defined and heavily studied would be ImageNet, CIFAR-10 & -100, and Fashion-MNIST & MNIST which all have sufficient benchmarking for our purposes.

6. Demo planned

Our demo will feature a set of presentation slides and a live demo during the presentation. The structure will depend heavily on our final implementation, findings, and dataset used.

a. **Introduce KANs:** dive into the KAN architecture, specifically the activation function implementation, modified training process, and paradigm shift from conventional MLPs.
b. **Optimizations/Benchmarks:** explain the optimizations we perform, the mechanisms used, and the speedup observed. Compare inference and training times on the same dataset to a traditional MLP model and unoptimized KAN.
c. **Live Demo:** real time demonstration of inference using KANs after presenting metrics collected from the training process.
d. **Findings/Conclusions:** summarize our findings, contributions, and their implications on future development and research on KANs in machine learning.

7. References

1. [KAN: Kolmogorov-Arnold Networks](https://arxiv.org/abs/2404.19756)
2. [KAN 2.0: Kolmogorov-Arnold Networks Meet Science](https://arxiv.org/abs/2408.10205)
3. [UKAN: Unbound Kolmogorov-Arnold Network Accompanied with Accelerated Library](https://arxiv.org/abs/2408.11200)
4. [MatrixKAN: Parallelized Kolmogorov-Arnold Network](https://arxiv.org/abs/2502.07176)
5. [A Comprehensive Survey on Kolmogorov Arnold Networks (KAN)](https://arxiv.org/pdf/2407.11075v3)
6. [ReLU-KAN: New Kolmogorov-Arnold Networks that Only Need Matrix Addition, Dot Multiplication, and ReLU](https://arxiv.org/pdf/2406.02075v1)