Building a KAN-Coder Model in Three Months on a 100k+ Budget: A Distributed GPU and Bare-Metal Approach

Global KAN Team
Magneton Labs

January 28, 2025

Abstract

This paper outlines a three-month roadmap for developing a large-scale KAN-Coder (KalmaGrove-Arnold-Networks) model on a budget exceeding $100k. The project leverages distributed cloud GPUs, dynamically optimized for cost-effectiveness and performance, and custom bare-metal trainers that exploit a specialized Linux kernel configured for CUDA-level acceleration. We propose an end-to-end approach combining the unique strengths of KAN-Coder's architecture with practical methodologies for scaling, cost management, and rapid deployment. Our aim is to demonstrate that a small, globally distributed team of 3–5 researchers and engineers can produce a competitive large-language-model system, comparable to GPT-o1/Deepseek-R1, within a constrained timeline and budget.

1 Introduction

Large Language Models (LLMs) such as GPT-o1 and Deepseek-R1 have demonstrated remarkable capabilities in natural language understanding and generation. However, the high computational and financial barriers often deter smaller teams from replicating or innovating on these breakthroughs. This paper addresses these challenges by presenting a practical strategy to build a KAN-Coder model in under three months with a $100k+ budget. We underscore the importance of:

- **Global collaboration:** Engaging a distributed team of 3–5 individuals, erasing nation-state boundaries.
- **Cost optimization:** Dynamically routing training workloads to different cloud providers based on real-time GPU pricing.
- **Bare-metal trainers:** Using custom Linux kernels for enhanced CUDA-level performance.
- **Scalable architecture:** Leveraging the KalmaGrove-Arnold-Networks (KAN) framework and KAN-Coder math models.

2 Background

2.1 KalmaGrove-Arnold-Networks (KAN)

KalmaGrove-Arnold-Networks are a class of architectures focusing on efficient parameterization and advanced sequence modeling. They incorporate Kalman-style updates (for robust state estimation) with Grove-based iterative expansions and Arnold transformations for non-linear embedding spaces. When combined, these approaches exhibit strong capabilities in learning linguistic representations and multi-modal patterns efficiently.

2.2 KAN-Coder

KAN-Coder is a specialized variant of KAN models tailored for code generation, text-to-text transformations, and symbolic reasoning. Its architecture seeks to integrate domain-specific embeddings with advanced sequence-to-sequence transformations, achieving high fidelity in tasks that require structured output or domain-constrained language generation.

3 Overall Project Strategy

3.1 Timeline: Three Months

The proposed approach divides the three-month window into several phases:

1. **Planning & Data Preparation (Weeks 1–2):** Finalizing objectives, collecting datasets, and performing initial data preprocessing.
2. **Prototype & Infrastructure Setup (Weeks 3–4):** Installing and testing the KAN-Coder codebase, setting up cloud accounts, configuring bare-metal trainers, and building a custom Linux kernel with CUDA optimizations.
3. **Model Development & Training (Weeks 5–10):** Implementing the KAN architecture, training prototypes, tuning hyperparameters, and refining cost-optimization strategies across multiple cloud providers.
4. **Validation & Iteration (Weeks 11–12):** Evaluating model performance, adjusting training strategies, and preparing the final deliverables.

3.2 Budget: $100k+

A budget of $100k+ will be allocated primarily to:

- **Cloud GPU resources:** Dynamically acquiring compute hours from multiple cloud vendors.
- **Bare-metal trainers:** Procuring high-performance servers where latency and overhead can be minimized.
- **Data storage and bandwidth:** Maintaining large datasets and handling high-volume I/O efficiently.
- **Engineering & Research efforts:** Supporting a globally distributed team of 3–5 researchers.

4 Technical Implementation

4.1 Distributed Cloud GPU Strategy

The training workloads will be orchestrated using a scheduling system (e.g., Kubernetes or Docker Swarm) that polls GPU pricing data from major cloud providers. Real-time bidding or subscription pricing will be leveraged to minimize training costs. When a GPU spot instance or discounted GPU resource becomes available, the scheduler will instantly migrate workloads to these resources.

4.2 Bare-Metal Optimization

To fully exploit the performance potential of CUDA, a custom Linux kernel with GPU-specific patches will be compiled. This kernel will include:

- Low-latency scheduling options optimized for HPC tasks.
- Stripped-down module set to minimize OS overhead.

- Tailored CUDA driver configurations for direct memory access.

Bare-metal servers will ensure consistent performance, reduced virtualization overhead, and finer control over GPU resource allocation.

4.3 KAN-Coder Math Model

KAN-Coder integrates:

- **Attention Mechanisms:** Extended multi-head attention, adapted from the KAN blueprint to capture both local and global dependencies.
- **Kalman Filters:** Integrations for robust hidden-state updates, reducing noise in the training signal.
- **Arnold Layers:** Non-linear transformations for feature disentanglement.
- **Grove Expansions:** Hierarchical expansions to capture multi-resolution patterns.

5 Evaluation & Iteration

Quantitative benchmarks will include perplexity on language modeling tasks, code generation accuracy (for selected coding benchmarks), and performance on general reasoning tasks. Throughout the final weeks, results will inform iterative changes in hyperparameters, data processing, and model architecture.

6 Conclusion

This paper presents a roadmap for building a large-scale KAN-Coder model within three months on a $100k+ budget. By combining globally distributed expertise, dynamic GPU allocation, bare-metal performance optimization, and the flexible KAN architecture, we propose that a small, committed team can achieve competitive results comparable to GPT-o1/Deepseek-R1. This endeavor underscores the democratization of AI research and paves the way for more agile, cost-efficient large-model development.

References

[1] OpenAI. *GPT Models*, 2023. Link.

[2] DeepSeek Group. *DeepSeek-R1: A Multi-Task Transformer*, 2024.

[3] KAN Researchers. *KalmaGrove-Arnold-Networks: The Next Generation of Sequence Modeling*, 2025.