nanoKimi: A Minimal, Efficient, and Scalable Implementation of the Kimi-K2 Architecture for Next-Generation Language Modeling

Aman Murari Singh

Student

amanmuraris@gmail.com

Abstract

This paper presents nanoKimi, a minimal yet powerful implementation of the Kimi-K2 architecture, developed as part of the Vizuara Research Challenge #1. Inspired by the simplicity of nanoGPT and the performance breakthroughs of Kimi-K2, this work focuses on integrating key innovations — the Muon optimizer, Mixture of Experts (MoE), and Latent Attention — into a lightweight and reproducible training and inference pipeline. We benchmark nanoKimi against nanoGPT on multiple metrics including training speed, inference throughput (TPS), model quality, and parameter efficiency. The results demonstrate nanoKimi's significant gains in both performance and scalability while retaining compactness and simplicity.

1. INTRODUCTION

Recent advancements in large language models (LLMs) demand architectures that are efficient, scalable, and easy to fine-tune. The Kimi-K2 architecture introduces a set of innovations — notably the Muon optimizer, Mixture of Experts, and Latent Attention — that offer substantial improvements in convergence speed, model sparsity, and contextual reasoning.

This paper introduces nanoKimi, a research-grade minimal implementation of Kimi-K2, designed with the goal of replicating its core contributions while benchmarking it directly against nanoGPT — the widely known baseline for GPT-style model simplicity.

2. MODEL ARCHITECTURE OVERVIEW

2.1 nanoKimi Architecture

nanoKimi consists of the following components:

Embedding Layer: Projects tokens into vector space.

Mixture of Experts (MoE) Layers: Sparse routing activates only a subset of expert networks per input

Latent Attention Modules: Activates attention heads selectively based on token utility.

Feedforward Layers: Standard linear projections with activation.

Output Layer: Maps back to vocabulary space.

Each block is implemented using native PyTorch.

2.2 Key Innovations

Component -> Description

Muon Optimizer-> Gradient-aware optimizer for faster convergence with stable updates.

MoE Routing -> Activates top-k experts per input; reduces computation and increases capacity.

Latent Attention -> Selects only useful attention heads based on content importance.

3.Inference Results (vs nanoGPT)

matric	nanoGPT	nanoKimi
Token/sec	30.3	45.2
Perplexity	22.8	18.1
Total params	74 M	75 M
Model size	0.62 GB	0.58 GB

The model produced fluent and coherent generations, demonstrating the utility of latent attention and sparse MoE routing in low-resource settings.

4. Benchmarking Against nanoGPT

- Monu optimizer will not work on CPU so please use GPU.
- Loss convergence of MUON Optimizer is lower than AdamW.
- Through the use of f latent Attention and Mixture of Experts we reduce the number of active parameter of Model.
- In same parameter of both model we see that nanoGPT take more time for infrance.
- With the help of MUON Optimizer we observe that Gradient stability is high as compare to AdamW.
- In nanoKimi take less to train then nanoGPT.

5. Conclusion

nanoKimi shows that the Kimi-K2 architecture can be distilled into a simple, modular, and efficient PyTorch codebase — rivaling and outperforming nanoGPT in both speed and accuracy. Through the integration of Muon optimization, Mixture of Experts, and Latent Attention, we achieve significant gains in inference throughput, convergence, and parameter efficiency.

Code: https://github.com/amanmurari/nanokimi