IML Final Project Report

SastaGPT 2.0: An Efficient Transformer-Based Language Model Inspired by DeepSeek V3

Abstract

For my final project, I worked on SastaGPT 2.0. This is the direct successor to SastaGPT (v1), an autoregressive character-level language model. This iteration of SastaGPT introduces several architectural changes drawn from the DeepSeek-V3 technical report. Enhancements include the implementation of Mixture-of-Experts (MoE) layers, Key-Value (KV) caching, and gating mechanisms. These modifications resulted in significant improvements in both training efficiency and output quality, even under limited compute constraints. After that, I worked on another minor iteration called SastaGPT 2.1 — Gotham Protocol (Codename "Alfred"). Alfred was born after hours of fine tuning on SastaGPT 2.0. The updated models, especially SastaGPT 2.0, demonstrates how modular transformer architectures can rival much larger models when optimized well.

Introduction

The original SastaGPT model was a basic character-level Transformer trained on literary texts. As we saw last time, it was constrained by tight compute and memory budgets. In this final version, the model evolves with state-of-the-art techniques such as MoE, KV Caching, and simple expert gating. Two variants were introduced — SastaGPT 2.0 and SastaGPT 2.1 — Gotham Protocol (Codename "Alfred"), each building upon modular design ideas to significantly improve training speed and output coherence. SastaGPT 2.1, in particular, utilized 8 experts and routed the top 2 per token using a simple gating mechanism. The results were dramatic: comparable quality to the v1 model was achieved in just 10% of the original training budget.

Literature Review

This project drew inspiration primarily from the DeepSeek V3 Technical Report (2024), which highlighted how expert sparsity and caching allow large models to be trained efficiently on limited compute. Additionally, we studied:
- Vaswani et al. (2017), 'Attention Is All You Need': foundational for Transformer architecture.
- Shazeer et al. (2017), 'Outrageously Large Neural Networks': for Mixture-of-Experts routing.
- SastaGPT (v1): served as the basic framework and baseline comparison for SastaGPT 2.0

Dataset Source and Description

For training, my dataset is the same as the one I used for my midterm project. This time I only used one of the two data corpus I collected during my midterm project:

- Stoic Philosophy Texts: Included Meditations by Marcus Aurelius, Letters from a Stoic by Seneca, and Enchiridion and Discourses by Epictetus. These texts were already processed, requiring minimal cleaning. So I also clubbed them in another text file.

- No complex preprocessing was required, and both corpora provided rich grammatical diversity for sequence modeling.

Data Exploration and Important Features

Since this is a character-level model, the data was processed into sequences of characters. Some key concepts(with my reasoning on using them) I would like to draw your attention to are:

- Contextual Dependencies: The length of context used (window size) played a crucial role in generating coherent text.

- Character Encoding: To make sure the complexity doesn't exceed the computing resources, a simple yet effective mapping of characters to integer indices was used. Each unique character in the corpus is mapped to an integer index to form the vocabulary. This allowed each character to be represented as a unique token within the vocabulary.

New Techniques Implemented

I primarily adopted these three unique ideas to my original SastaGPT:

- KV Cache: During inference, we cache the computed key and value tensors, avoiding redundant calculations across time steps. This improves runtime memory efficiency and latency, and is especially important for generating long sequences.
- Mixture-of-Experts (MoE): We implemented sparse MoE with 8 parallel experts and top-2 expert selection for each token. The router computes scores from hidden states and selects top-k experts. Though expert specialization was not enforced, this already showed major gains in generalization and learning speed.
- Gating Mechanisms: Each expert's output is weighted using a softmax gating vector computed from the input token's hidden representation. This allowed the model to dynamically choose expert pathways for different inputs, albeit with naive routing heuristics.

Model Variants and Observations

SastaGPT 2.0

Built with 2 experts and naive gating. Instead of gating, I took some inspiration from the decision tree ensemble techniques we did in class and implemented it for the experts instead of trees. Without RoPE and expert specialization, the model still outperformed the original v1 model within just 20% of the training steps.

SastaGPT 2.1 — Gotham Protocol (Codename "Alfred")

This is where my tinkering with hyperparameters shined. Alfred leverages 8 experts with top-2 gating, KV Cache, and dropout-based gating. Surprisingly, Alfred achieved SastaGPT v1's quality in just 500 iterations compared to 5000. This 10x improvement underlines the dramatic gain from sparse modular computation. Even with limited context length and batch size, output quality improved visibly.

Final Hyperparameters

```
BATCH_SIZE = 32
CONTEXT_WINDOW = 256
EPOCHS = 2500
CHECKPOINT_INTERVAL = 500
LR = 3e-4
DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'
EVAL_ITERS = 200
EMBEDDING_DIM = 512
HEADS = 8
LAYERS = 8
DROPOUT_RATE = 0.1

# MoE Configuration
NUM_EXPERTS = 8
TOP_K_EXPERTS = 2
```

Note

During inference on the HPC, I encountered that my validation error started increasing after a certain point which meant that my model was starting to overfit the training data. I had two options at that point, either I could add more data or I could reduce the training parameters. Since finding rich data is very difficult, I chose the second option.

Results

The introduction of KV Caching and Mixture-of-Experts drastically improved both convergence speed and final quality. Here is a comparison:

Model Version	Parameters	Training Epochs	Performance vs v1
SastaGPT (v1)	~10M	5000	Baseline
SastaGPT 2.0	~13M	5000	Beats v1 in just 20% training episodes
SastaGPT 2.1	~60M	2500	Beats v1 in just 10% training episodes

Conclusion

This project demonstrates that with clever architectural choices like MoE and KV Cache, small models like SastaGPT can achieve disproportionately good results. The performance gains seen in SastaGPT 2.0 show that modularity and sparsity can drive efficient training even without large compute power. Future improvements could include: RoPE integration, specialized expert routing, and token-level encoding strategies such as Byte-Pair Encoding (BPE).

References

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). "Attention Is All You Need."

- DeepSeek V3 Technical Report, 2024.
- Shazeer et al., 'Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer', ICLR 2017.
- Andrej Karpathy Neural Networks Series
- Internet Archive: Shakespeare's Works and Stoic Philosophy Texts

GitHub Repo:

Please find the respective GitHub Repo here: https://github.com/alwaysafoujdar/SastaGPT2.0