CSC490 Assignment A3 - Pre-training Nanochat- 10% of Final Grade

Due: Mar 6th, 10:59pm with a Github PR and on Quercus

It's time to train a LLM! In this assignment you'll make a few tweaks to nanochat [Karpathy, 2025]. You shouldn't need to make too many changes to the repo, but instead make improvemnts and learn how to run ablations. Fork the repository to get started! Note: before running the training on large machines, replicate the code running locally or on a small GPU so you don't waste credits on false starts.

1 Part One: Architecture Review (20 marks)

Review the architecture choices of the original nanochat (Oct 13) and create a diagram of the model architecture. Since the original nanochat, Karpathy has made changes as of a few weeks ago (Feb 2026) (which has led to mixed feelings in the context of this assignment). Now its your turn to continue to update the architecture! Compared to the newest nanochat architecture (Feb), find 3 changes to LLM architecture from the literature referencing meaningful papers. For ideas you can see how ModernBert [Warner et al., 2025] made changes to BERT [Devlin et al., 2018]. Create a table summarizing the changes you found, including the motivation for the change, the technical details of the change and the potential impact on model performance.

2 Part Two: Ablations on a nano nanochat (30 marks)

Pick 2 of the architecture changes you found in Part One and measure their impact.

- Choose a smaller nanochat configuration (picochat), justify your choice and train a baseline model.
- Train a picochat with each change and compare the performance to the baseline picochat
- Create a table summarizing the results and include commentary on the impact of each change.
- Track them using weights and biases or a similar tool to visualize the training process and compare the different models.
- Make sure to use your credits intelligently and include a discussion of the cost of training these models in your writeup.
- Comment how your ablations may translate for a larger run

3 Part Three: Extending the context window (30 marks)

Nanochat runs with a context window of 2048 sequences by default. You'll make a few changes to tweek this:

- Reduce the sequence length to a lower amount (ex 512) and train a picochat on a portion of the dataset. Make sure to justify your choices based on literature.
- Use this reduced sequence checkpoint of your model and increase the sequence length to 2048 and continue training picochat.
- Build a new eval to compare the performance of checkpoint 1 and checkpoint 2 on a task of your choice.

4 Part Four: Training Your Final Nanochat (20 marks)

Train a final nanochat model with the architecture changes.

- Choose a configuration for your final nanochat model, justify your choice and train it on the full pre-training dataset.
- Predict the scaling law for your pico vs nanochat model and compare it to the actual results.
- Create a table summarizing the results and include commentary on the impact of each change.
- What "emergent abilities" do you see in your nano vs picochat model? Create a list of 10 questions that your nanochat can answer,but your picochat cannot and include them in your writeup.

Highlevel marking criteria:

- Understanding of literature and research
- Ablation quality and tracking
- Code quality

5 Submission Instructions

Submit a pull request to the course Github repo with your assignment in a folder named a3 with your a3.pdf on a branch called a3, include your team members names on the first page with student IDs.

References

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., Gallagher, A., Biswas, R., Ladhak, F., Aarsen, T., Adams, G. T., Howard, J., & Poli, I. (2025). Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 2526–2547). Vienna, Austria: Association for Computational Linguistics. `https://aclanthology.org/2025.acl-long.127/`

Karpathy, A. (2025). nanochat: A tiny chatbot arena and training harness. `https://github.com/karpathy/nanochat/discussions/481`