Exploring Language Model Scaling Behavior Using nanoGPT & TinyStories

Course: INF-3600 (Generative Artificial Intelligence)
Deliverable: 2-4 page report + experimental logs/plots

NB: You and your class will be the first official “test candidates” for this assignment! Thus, if you experience any problems, major or minor, or simply want to share feedback/ideas with us, our team (Victor, Helge and Benjamin) will respond as fast as we can. We recommend discord, or ask Victor directly when he’s around in the lessons! Helge is the main person to blame, though, as he is the main author 😎.

1. Overview

In this assignment, you will explore how the architectural hyperparameters of a Transformer language model—specifically **model depth**, **embedding width**, and **number of attention heads**—affect learning behavior, generalization, and output quality.

You will use **nanoGPT** as a lightweight GPT-like workbench and train a sequence of small models on the **TinyStories** dataset. TinyStories is small enough that you can train multiple models in a single day, yet rich enough to reveal scaling patterns and overfitting behavior similar to those seen in larger LLMs.

Your task is to design and conduct controlled experiments that vary the model architecture along one dimension of hyperparameter space at a time, observe how performance changes, and summarize the findings in a short technical report.

2. Setup instructions

1. *Prepare a processing platform:* Create a VM on the Sigma2 (look at the instructions at this link), or use your account on gpusrv1.cs.uit.no or at the cluster at IFI. You may run it on the CPU on your laptop too, but it may take at least 10 times longer to train the models. ***PS1:*** *If you have a macbook with a GPU equipped video card, you should be able to utilize this by changing the parameter* `device` *to* `mps` *in the config file you create (see point 6). If you test this, you may also have to tweak the batch_size to a bit lower value like 3.* ***PS2:*** *If you use gpusrv1, it has two GPU cards. You can use the command* `gpustat` *to check the load on each card. Then, if you want to switch using the Blackwell card instead of the default Ada, you should be able to address this by setting the same* `device` *parameter mentioned above to* `cuda:1`*. You can change this parameter as you train various models to get the optimum load balance.*
2. *Get nanoGPT:* `git clone https://github.com/karpathy/nanoGPT.git`

3. *Get the TinyStories dataset (or skip this step, you can make adjustments in step 4 too):* https://huggingface.co/datasets/roneneldan/TinyStories/resolve/main/TinyStoriesV2-GPT4-train.txt using `wget` or similar and place it under a new directory with the name input.txt: `<nano-home-dir>/data/tinystories/input.txt`
4. *Splitting the data in train/validation parts*: Copy the script `prepare.py` from data/sharespare_char to data/tinystories and make the adjustments to fit the new dataset. **Run it**. PS: For some reason this script doesn't work on Sigma2. We have done this step on another platform and made the two files output from this script on /mnt/aicourse-ns9808k/shared. You may copy the files from there to the data/tinystories directory.
5. *Prepare config file:* Copy the file `train_shakespare_char.py` to `train_tinystories.py` in the directory `<nano-home-dir>/config`. Adjust the parameters `out_dir, wandb_project, dataset`
6. *Ensure the setup works (run it with a few iterations):* `python train.py \ config/train_tinystories.py`

3. Required experiments

You will run **three controlled sweeps**. For all experiments, you will only have to work with adjusting the parameters mentioned below `#baby GPT model` in the config file, except for the dropout parameter which we recommend to change to 0.0 for these experiments.

1. For each experiment, **vary only one architectural dimension at a time**, so the impact of that dimension is observable.
2. Train the model.
3. Make samples from the trained model by running `python sample.py` (after having adjusted the `our_dir` parameter in the top of the file to match where your model was saved. NB: We recommend also changing the max_new_tokens to 1000 or similar)
4. For each experiment record
    a. Generated samples for each model. You may store a few of them in separate files for easier comparison. Keep representative examples showing differences in
        i. Coherence
        ii. Vocabulary richness
        iii. Narrative structure
        iv. Repetition or degeneration
    b. Final validation loss during training
    c. Parameter count (nanoGPT outputs it)
    d. Training loss curves. *PS: You do not need to plot all of them in the report, but if you see one of them deviating a lot from ordinary behaviour, it would be illustrative to add it.*

Experiment 1 - Depth sweep (n_layer)

Goal: Understand how increasing depth (=number of Transformer blocks) affects learning and generalization.

Fixed parameters: n_head = 8, n_embd = 256
Vary: n_layer ∈ {2, 4, 8, 12}
Evaluate in report:

- Observation on coherence, structure and diversity of generated text as you vary number of layers/transformer blocks

Experiment 2 - With sweep (n_embd)

Goal: Study the effect of representation width (model embedding size) on expressiveness and overfitting.
Fixed parameters: n_layer = 6, n_head = 8
Vary: n_embd ∈ {128, 192, 256, 384}
Evaluate in report:

- Consider *diversity* and model overfitting (model becomes "over-ambitious" in predicting sentences compared to the training scope) with increased embedding size.
    - PS: A simple measure on diversity could be n_unique_words/n_words. That is: The fraction of **unique words** in a piece of text versus **total number of words** in that text. The closer this fraction is to 1, the higher the diversity would be.
- Size of the model (this is reported when you start training)

Experiment 3 - Attention head sweep (n_head)

Goal: Evaluate how attention head partitioning affects attention diversity.
Fixed: n_layer = 6, n_embd = 256
Vary: n_head ∈ {4, 8, 16, 32}
Record:

- Brief qualitative differences in story structure or consistency

4. Deliverables

Your must submit a 2-4 page report (pdf or markdown) including

1. Introduction (½ page)
    - Purpose of the experiment
    - Brief description of the TinyStories dataset and nanoGPT setup
2. Methods (½ page)
    - Describe the experimental design
    - State which settings were fixed and which were varied
    - Include a table summarizing your model configurations
3. Results (1-2 pages)
    a. **Quantitative analysis**
        - Plots of training/validation loss
        - Table of parameter counts
        - Diversity metrics
        - Notes on where performance saturates or degrades

   b. **Qualitative analysis**
      - Short excerpts (~2–4 sentences) illustrating differences between models
      - Observations on coherence, creativity, and overfitting
4. Discussion (½ page)
   - Identify the model sizes where performance improves
   - Identify where additional capacity provides diminishing returns
   - Hypothesize why overfitting or degeneration occurs

5. Conclusion
   - What architectural choices matter most?
   - What would you explore next?

5. Optional additional investigations

You may also explore

- Effects of dropout or weight decay
- Context length changes (128 vs 256 vs 512)
- Temperature sampling effects
- Early stopping and learning-rate schedules

Generating Loss Curves

- These can be generated from what nanoGPT reports either from logging through the platform wandb, or from the standard out on the console. You may put some effort into getting the wandb framework up and running (we haven't, but if you do, we would like you to share your expertise with us!). However, you should also be able to log the console output to a file via UNIX commands like this: `python train.py config/train_tinystories.py | tee out_<name>/log.txt`

References

- XArchive paper on TinyStories (recommended read!): https://arxiv.org/pdf/2305.07759.pdf
- Medium article about recreating results in the paper using nanoGPT