Final Shared Task Submission
11-664/763: Inference Algorithms for Language Modeling
Fall 2025

Your name (Andrew ID): Test Student (tests)

Instructors: Graham Neubig, Amanda Bertsch
Teaching Assistants: Clara Na, Vashisth Tiwari, Xinran Zhao

Due: December 4th, 2025

1 Introduction

Throughout this course, you've explored the fundamental tradeoffs between accuracy and latency in LLM inference. This final project asks you to synthesize everything you've learned– including techniques like best-of-n sampling, minimum Bayes risk decoding, self-refinement, speculative decoding, batching, and continuous batching– to build an optimized inference system.

In real-world serving environments, you must balance heterogeneous requests and carefully consider accuracy-throughput tradeoffs. The final project is to develop an inference system that can process requests from any of the three shared tasks.

There is no single correct solution. This assignment challenges you to make informed design decisions based on your understanding of how different optimizations perform under various system constraints.

Instructions

Please refer to the collaboration and AI use policy as specified in the course syllabus. Additionally, note that **no off-the-shelf inference servers can be used (e.g. vLLM, sglang, etc)**. Assume NVIDIA GPU hardware with CUDA throughout this assignment.

Shared Tasks

Throughout the semester, you have worked with data from three shared tasks. We host the data for each shared task on Hugging Face; you can access them at **[this link]**.

Your final submissions will be evaluated on a hidden test set. You can use all available data for validation, tuning hyperparameters, and selecting what methods you would like to use for your final submission.

One last time, here are the three tasks:

Algorithmic The task that the language model will tackle is N-best Path Prediction (Top-P Shortest Paths). Given a directed graph $G = (V, E)$ with $|V| = N$ nodes labeled $0, \ldots, N-1$ and non-negative

integer edge weights $w : E \to 1, \ldots, W$, the task is to find the top-P distinct simple paths from source $s = 0$ to target $t = N - 1$ minimizing the additive cost

$$c(\pi) = \sum_{(u,v)\in\pi} w(u, v). \tag{1}$$

The output is a pair

$$\texttt{paths} = [\pi_1, \ldots, \pi_P], \quad \texttt{weights} = [c(\pi_1), \ldots, c(\pi_P)], \tag{2}$$

sorted by non-decreasing cost. The language model will be expected to use tool calls[1] to specify its answer.

Evaluation compares predicted pairs $(\pi, c(\pi))$ against the reference set with the score

$$\text{score} = \frac{|(\pi, c(\pi))\text{pred} \cap (\pi, c(\pi))\text{gold}|}{P}. \tag{3}$$

MMLU medicine We will use the two medicine-themed splits of MMLU: college_medicine and professional_medicine. Evaluation is on exact match with the correct multiple-choice answer (e.g. "A").

Infobench Infobench provides open-ended queries with detailed evaluation rubrics. Evaluation **requires calling gpt-5-nano**; we expect that the total cost for evaluation for this homework will be substantially less than $5. See the paper for more information.

[1] https://platform.openai.com/docs/guides/function-calling

2 Deliverables

You will need to turn in **three types of artifacts** for your final shared task: one or more inference systems, a final report, and a poster presentation

3 System submissions [65 points]

The main component of your submission will be the shared task system. You can submit **up to three systems**; we will evaluate all three, and choose the best-performing for each of the grading criteria.[2]

3.0.1 Compute

Your system should use 2 80GB A100 GPUs.

3.0.2 Models

Your system should use model(s) from the Qwen 3 family. You can choose any model or models from this family that you wish, balancing efficiency and performance.

In addition to the Qwen 3 family, you may use *any models you wish* that are 500M parameters or less.

3.0.3 Outside resources

You may include any other data you would like to use in the system submission, but you may **not** use Internet access for any part of the system except for downloading Qwen3 models. For instance, if you want to do RAG, you may include a datastore to retrieve from in your shared task submission, but you may not call an external API for web search.

3.0.4 File format and system setup

Upload your systems to Canvas in a single .zip containing:
- andrewID_1.zip
- andrewID_2.zip
- andrewID_3.zip

If you are submitting three systems, where the `systemID` = 1, 2, or 3.

We will unzip your files and run the following command:

```
./andrewID_systemID/deploy.sh
```

This should call any necessary modal commands to deploy your system. We will allow 5 minutes for setup, and then begin querying your system. Your modal script should:

1. be an app named `andrewID-system-systemID`
2. have a timeout of at least 10 minutes
3. support up to 300 concurrent requests
4. have an entrypoint named `completion` that supports the input and output format of the OpenAI completions API.[3]

[2] You are only *required* to submit one system, though.

[3] However, unlike the OpenAI completions endpoint, you do *not* need to honor any user-specified decoding settings; we will not be passing these at test time.

The starter code shows a minimal example of a deployable modal script that has the correct timeout and concurrency settings, and examples of calling this endpoint with a single example or batch of examples. You should make sure that your final system is callable by passing prompts in the same way. You are not required to pass the exact inputs that the user requested into any models in your system; you can feel free to apply chat templates, additional instructions, or other preprocessing of your choice.

3.0.5 Input size

Your submission folder size should not exceed 10GB. This does *not* include the size of the model checkpoints– you should download these in your system deployment script. This space can be used for your code and for any data you would like the system to have access to (e.g. documents for RAG).

3.0.6 API endpoint

Since this simulates a production setting, your code must support online inference. You should create a RESTful API that can handle online, concurrent requests according to the OpenAI completions API. Your system should handle heterogeneous request types (i.e. inputs from all three shared tasks). Inputs will *not* be labeled with which task they come from. Inputs will be passed in batches of 1 or more examples per request call, throughout the serving period.

3.0.7 Packages/library code

You **cannot use** inference serving packages– this includes:

1. SGLang
2. TensorRT
3. MLC LLM
4. FlashInfer
5. vLLM
6. Megatron-LM
7. Other inference-serving-focused libraries

You **may use** the following packages if you wish:

1. Hugging Face `transformers` and/or `accelerate`
2. Torch's native distributed inference support (e.g. `torch.nn.parallel.DistributedDataParallel`)
3. Quantization libraries
4. Flash attention
5. Libraries for retrieval, including vector datastores
6. Other off-the-shelf inference optimizations, as long as these do not call SGLang/FlashInfer/vLLM

If you have questions about using a particular library or method, post on Piazza.

3.1 Grading

For grading the three metrics to consider are accuracy, throughput, and the distance from the pareto-optimal. The three systems you submit will each be graded separately for these three metrics with the following grading criterion.

Your system performance will be divided into three components. For each component, your grade will be **scaled relative to these three baselines**:

1. **Baseline (80% performance):** To earn at least a score of 80%, your system should match the performance of a very simple baseline system that the instructors will implement.

2. **Improved Baseline (90% performance) :** To earn at least 90%, your system should be able to match at least the performance of a more intelligent but still simplistic baseline design that we will implement.

3. **Ceiling (100% performance):** The best performance any student submission achieves for this task.

For instance, if your best task accuracy performance was between the accuracy scores of the 90% and 100% systems, your task accuracy grade would be assigned relative to your distance between the two.

Task accuracy [25 points]: Your submitted system that gets the highest *average score* across the hidden test set, averaged over the three tasks, will be scored for task accuracy. You can assume that there will be equal numbers of examples from each of the three tasks in the test set.

Efficiency performance [25 points]: Your submitted system that gets the highest *total number of requests served* in 20 minutes will be scored for efficiency performance. This system *must match at least the task accuracy of the 80% baseline* in order to receive credit for efficiency improvements.[4]

Accuracy-efficiency tradeoff [15 points]: We will graph all systems' tradeoff between the task performance and efficiency performance scores, and use this to estimate the empirical Pareto frontier. You will be scored by your distance to this frontier (so multiple submissions may receive 100%, if they are all on the frontier). Again, we will use your highest scoring submission on this criteria to assign your score for this component, and systems should achieve at least the task accuracy of the 80% baseline.

In general, you may submit systems that are expected to score well on only one or some of the criteria – for example, you may wish to submit one system that is expected to have high task performance, one that is expected to be very efficient, and one that is more balanced across efficiency and accuracy.[5] On the other hand, a single system may be scored for multiple criteria (e.g. including if only one or two systems are submitted) – we will run all of up to three submitted systems for each student and assign scores automatically.

[4] i.e. a submission that is highly efficient because it immediately "completes" all requests without attempting to perform inference with them does not receive any credit for high efficiency. Our 80% baseline system will be very simple, and some of its task accuracy will be capped by its inefficiency (i.e. it will return only a subset of all requests in time), so it should not be difficult to outperform it on both accuracy and efficiency.

[5] The caveats are that 1) a system that is so inefficient that it cannot serve more than a few requests in the allotted time will not be able to score well on task performance either, and 2) as mentioned, an efficient system must still meet the task accuracy performance of the 80% baseline.

4 Final report [20 points]

4.1 Required components

Your report should use the COLM conference LaTeX, with 4-8 pages of main text including a figure. You can use appendices for additional content.

Minimally, it should include:

1. Introduction to tasks (in your own words, not copying the homework!).
2. Related work discussion: what ideas inspired your system?
3. System description: this should be at a similar level of detail to a research paper; complete enough that someone well-versed in the field could recreate your design.
4. Original diagram/visualization of system: this should be a figure you made that shows how your system works. If you have more than one system, you can either choose one to display in the main text and add appendices describing the other systems or try to combine all systems into a set of figures/system descriptions.

4.2 Grading

Your report will be graded on its clarity, quality, and thoughtfulness. If there are methods you chose not to include, you should discuss these as well and explain your final design choices. Your system does *not* need to invent a new inference technique or use the fanciest possible method for each task, but if you use a simple technique, it should still be described clearly.

5 Poster presentation [15 points]

You must create a poster for your project (in the style of a research paper at a conference). These will be presented at our final poster presentation on December 1st.

5.1 Required components

The poster should include your system diagram/visualization and explain your design choices.

You must **print a physical poster** for the presentation. If you are in SCS, you can use SCS printing services for free; if you are not in SCS and would like us to print your poster, you can email it to `abertsch@cs.cmu.edu` **no later than midnight EST on Friday, November 28th** and we will submit a print request on your behalf. If you are unable to make the poster presentation time, please post on Piazza as soon as possible.

5.2 Grading

Your poster will be graded on poster quality and clarity [10 points] and your presentation of the work during the poster session [5].