Efficient AI (CS6013) Project

Autumn 2026–2027

Course Instructor

Prof. Aditya Desai

Teaching Assistants

Lavinia Nongbri
Hasmita Kurre

Contents

1 Project Description

This project will be a course long project where you have to develop a recipe for model compression, i.e., given a model and a target domain, your recipe should be able to compress the model so as to maintain the performance of the model on the target domain. You can experiment and develop your own recipe (including reusing existing research / githubs / papers) as starting points) on different models/target datasets and keep improving the recipe over the course of the project. We will be learning about model compression techniques throughout the course and hopefully the discussion can help you think about how to improve.

1.1 Task Objective

For the Autumn 2026–2027 offering, the project setup is as follows:

- **Task:** Given a model and target domain, compress while preserving performance.
- **Base model:** Qwen3.5-4B
- **Target domain:** Maths

1.2 Tracks

There are three tracks for the project. Each student will be required to submit for the first two tracks. The third track is bonus track and will be optional.

- **Track 1**: Here the models will be evaluated on the "memory" alone, i.e. the size of the model checkpoint in memory. (Specifically, we will not worry about if this memory will lead to any speed benefits on CUDA or not.) This is the pure model compression track. The model compression targets are 10%, 20%, and 40%.
- **Track 2**: Here the models will be evaluated on the "memory" but techniques used should be aligned with those that can be used to speedup the model on CUDA in principle. We will not ask you to implement the speedup part, but you should be able to explain how the techniques you use can be used to speed up the model on CUDA.
- **Track 2+**: This is the bonus track where you can develop and submit kernels required for speedup the model on CUDA. You have to show speedups in specific submodules (ex. linear layers, attention layers, etc.) at some large scale regime. This track will be open for topperformers in mid-term evaluation and will be provided with A100/H100 GPU access. This track will be used to push up the grades.

2 Deliverables

For every model submission, you are required to submit the following:

- **Code:** Submit the complete code for the model compression pipeline through a GitHub repository.
- **Model checkpoint:** Submit the compressed model checkpoint via Hugging Face.
- **README:** A README file must be included in the GitHub repository, documenting the setup, usage, and instructions for reproducing the results.
- **Report:** A technical report describing the compression method experimented with so far and also discussing the results and analysis must be submitted at checkpoints.

2.1 Github

Repository Requirements: The GitHub repository must contain, at minimum, the following files:

- `compress.py` – Implements the model compression pipeline.
- `decompress.py` – Implements the model decompression pipeline.
- `pyproject.toml` – Specifies the Python project configuration and all dependencies required to execute the compression and decompression pipelines. **Please note that the dependencies should be compatible with CUDA 12.6**.

The GitHub repository must follow the hierarchy below:

```
CS6013/
└── <roll_number>/
    ├── Week01/
    │   └── Compression_<compression_target>/
    │       └── Submission01/
    │           ├── compress.py
    │           ├── decompress.py
    │           ├── pyproject.toml
    │           ├── compression/
    │           │   ├── __init__.py
    │           │   ├── calibration.py
    │           │   ├── quantize.py
    │           │   └── ...
    │           ├── decompression/
    │           │   ├── __init__.py
    │           │   ├── dequantize.py
    │           │   └── ...
    │           └── README.md
    ├── ...
    │   └── ...
    │       └── ...
    └── ...
```

2.1.1 Compression Code

The compression pipeline must be executable using the following command.

```
python compress.py \
    --model_name <model_name> \
    --checkpoint_path <checkpoint_path> \
    --output_path <output_path>
```

- `--model_name`: The name of the base model used for compression. In our case, this would be `Qwen-3.5-4B`.
- `--checkpoint_path`: The path to the base model checkpoint.
- `--output_path`: The path where the compressed model checkpoint must be saved.

2.1.2 Decompression Code

The decompression pipeline must take a compressed model (HuggingFace checkpoint) and reconstruct the model back to fp16.

The decompression code must be executable using the following command:

```
python decompress.py \
    --model_name <model_name> \
    --checkpoint_path <checkpoint_path> \
    --output_path <output_path>
```

- `--model_name`: The name of the base model used for compression. In our case, this would be `Qwen-3.5-4B`.
- `--checkpoint_path`: The path to the compressed model checkpoint.
- `--output_path`: The path where the decompressed model checkpoint must be saved.

2.1.3 README

1. The GitHub repository must contain a `README.md` file that documents only instructions for setting up, running the submitted code, and checkpoint. No need to explain the compression techniques.
2. Mention the dtype of the base model and the compressed model (e.g., `torch.float16` →`int4`).

Points to Note

- It is mandatory to add the TAs as collaborators to the GitHub repo.
- No manual modification of the source code should be required for implementing compression and decompression.
- Keep the implementation modular. All compression-related functionality should be kept under `compression/` and decompression-related functionality under `decompression/` except for `decompress.py` and `compress.py`.
- `compress.py` and `decompress.py` must serve as the only entry points for their respective pipelines. **Do not hard code any of the arguments.**
- Follow the folder structure and nomenclature. Failure to do so will lead to a 0 grade.

2.2 Hugging Face Checkpoint

The compressed model checkpoint must be uploaded to Hugging Face and should contain the files required to load and run the compressed model.

The HuggingFace checkpoint URL must be of the format:

```
https://huggingface.co/<huggingface-username>/<EnrollmentNo>-Week<week-number>-
    Compression<compression-target>-Submission<submission-number>
```

A typical checkpoint repository may have the following structure:

```
model-checkpoint/
  ├── config.json
  ├── generation_config.json
  ├── model.safetensors
  ├── tokenizer.json
  ├── tokenizer_config.json
  └── special_tokens_map.json
```

The exact set of files may vary depending on the compression method.

Points to Note

1. Keep the HuggingFace checkpoint **public**.
2. The checkpoint must be self-contained. TAs must not run the compression code to get the submitted model.
3. The checkpoint must be compatible with the submitted `decompress.py` implementation.
4. You must not include a `README` file, source code, notebooks, training logs, experiment outputs, or other files that are not required to load, decompress, or evaluate the model.

2.3 Report Submission

There will be a submission of the project report **twice** during the semester - the first at the **mid-term**, and the second at the **end-term**. TAs will communicate the exact time and details.

The report should document the compression techniques experimented with up to that checkpoint. It should discuss the observations from these experiments, including what failed and what worked.

Points to Note

1. Each report must be at most **3 pages**, excluding references and appendix.
2. The report writing has a strict **NO-AI policy**. Any use of AI in the report will result in a grade of 0 for the entire project. AI should not be used even to check grammar/spell check, or otherwise. If found, the consequences are severe (0 grade for the entire project).

3 Evaluation

3.1 Submission

1. **What to submit for iterative submission and frequency of updates**: You can make a maximum of 1 submission (for each compression target) per week. We will be updating the leaderboard every week with the latest submissions. You need to push your compressed model to your huggingface. And submit the model link to us via Google Form shared with you.

2. The Google Form will remain open throughout the semester. A submission for a given week is considered from **Saturday 12:00 AM** through **Friday 11:59 PM**.

3. If you submit more than one model in a week, the latest submission will be counted.

4. **No duplicate submissions.** If you tried a compression method and the model has been successfully evaluated (the results appear on the leaderboard), the same model must not be submitted again. However, if a submission receives a score of **0** due to a violation of the submission requirements, you may correct the issue and resubmit the model.

5. **Modification of the GitHub repo and HuggingFace is not allowed once submitted.**

3.2 Grading

1. Weekly submissions are graded on the "math" domain. However, the exact dataset for evaluation will be hidden and will only be used for populating the leaderboard.

2. The leaderboard results are kept anonymous. We encourage you not to disclose your identity.

3. **Evaluation Parameters:** The following parameters will be used to evaluate the submitted model:

   - **Temperature:** 1
   - **Token length:** 32K
   - **Thinking mode:** Enabled

4. Submissions that fail to adhere to the folder structure and naming conventions, or use a base model other than the specified, will receive a **zero grade**.

3.3 Weightage

This project constitutes **60% of the overall course grade**. Students who submit for **Track 2 or higher** can earn up to **5% additional points** over the total course grade based on their performance in the bonus track.

The project grade is normalized to 100% and is divided equally between the two evaluations as follows:

<table>
<tr><th colspan="2">Eval 1</th><th colspan="2">Eval 2</th></tr>
<tr><td colspan="2">50%</td><td colspan="2">50%</td></tr>
<tr><th>Position on Leaderboard</th><th>Report</th><th>Position on Leaderboard</th><th>Report</th></tr>
<tr><td>60%</td><td>40%</td><td>60%</td><td>40%</td></tr>
</table>

3.4 AI Policy

The coding, ideation, etc has a Any-AI policy, which implies you can use any AI tools to help you with the project. In fact interesting AI usage is encouraged and will be rewarded when reported in the report. However, the report writing has a strict NO-AI policy. Any use of AI in the report will result in a grade of 0 for the entire project. AI should not be used even to check grammar / spell check or otherwise. If found, the consequences are severe (0 grade for entire project).

3.5 Honor Code

While doing every submission, you will be required to sign the honor code pledge. You can find the pledge at the end of the submission form.

The honor code is as follows:

I hereby declare that this submission is my own work and I have not shared my code or artifacts with any other student. I understand that any violation of the honor code will result in a grade of 0 for the project.

4 Known Issues

At present, vLLM does not provide a registered text-only `Qwen3_5ForCausalLM` class. The Qwen3.5 family uses a hybrid architecture combining Gated DeltaNet and sparse MoE, while vLLM currently exposes the multimodal `Qwen3_5ForConditionalGeneration` class. Loading a text-only checkpoint using the multimodal class can result in a weight-prefix mismatch and subsequent loading failure. Issue Reference

Because of this issue, please use only the `Qwen-3.5-4B` multimodal checkpoint. Qwen3.5-4B