Attempt on Cerebras:

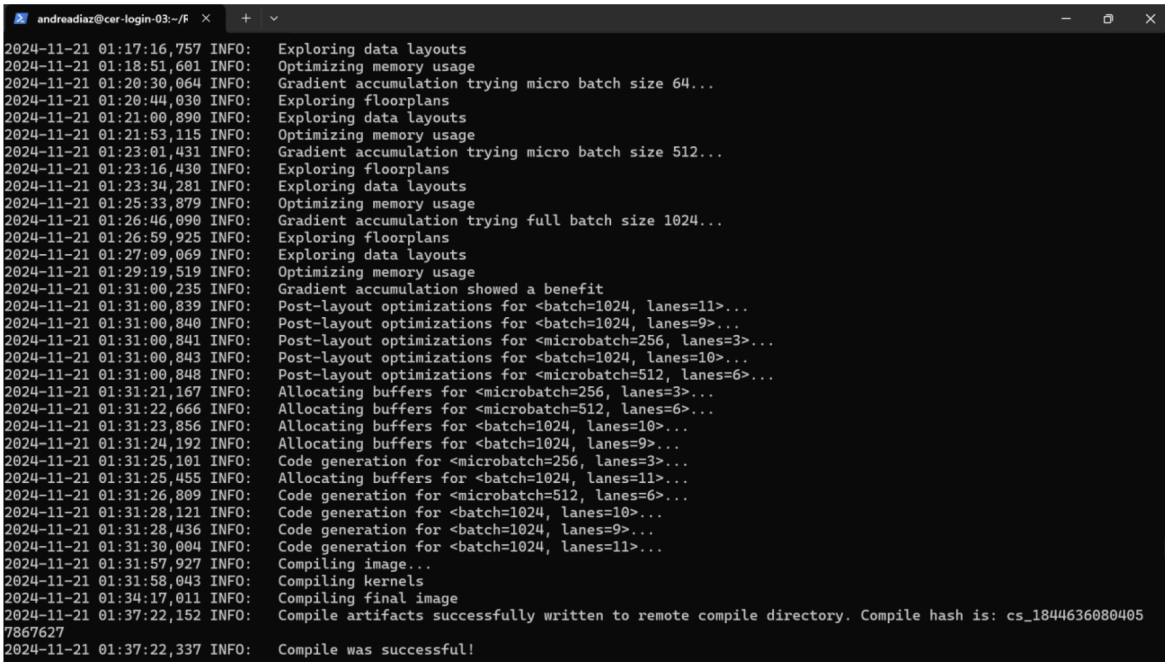
```
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "<frozen importlib._bootstrap>", line 1014, in _gcd_import
  File "<frozen importlib._bootstrap>", line 991, in _find_and_load
  File "<frozen importlib._bootstrap>", line 961, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "<frozen importlib._bootstrap>", line 1014, in _gcd_import
  File "<frozen importlib._bootstrap>", line 991, in _find_and_load
  File "<frozen importlib._bootstrap>", line 961, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "<frozen importlib._bootstrap>", line 1014, in _gcd_import
  File "<frozen importlib._bootstrap>", line 991, in _find_and_load
  File "<frozen importlib._bootstrap>", line 973, in _find_and_load_unlocked
ModuleNotFoundError: No module named 'cerebras.modelzoo.models.nlp.bert.venv_cerebras_ptytorch==2'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "run.py", line 41, in <module>
    main()
  File "run.py", line 32, in main
    main(
  File "../../../../../cerebras/modelzoo/common/run_utils.py", line 115, in main
    return run_with_params(
  File "../../../../../cerebras/modelzoo/common/run_utils.py", line 182, in run_with_params
    return run_trainer(
  File "../../../../../cerebras/modelzoo/trainer/utils.py", line 85, in run_trainer
    trainer = configure_trainer_from_params(params, model_fn)
  File "../../../../../cerebras/modelzoo/trainer/utils.py", line 281, in configure_trainer_from_params
    _validate_trainer_params(trainer_params, model_fn_or_name)
  File "../../../../../cerebras/modelzoo/trainer/utils.py", line 1128, in _validate_trainer_params
    if (config_class := registry.get_config_class(model_name)) is None:
  File "../../../../../cerebras/modelzoo/common/registry.py", line 360, in get_config_class
    cls._import_modules()
  File "../../../../../cerebras/modelzoo/common/registry.py", line 96, in _import_modules
    cls._import_modules_for_registry(
  File "../../../../../cerebras/modelzoo/common/registry.py", line 69, in _import_modules_for_registry
    raise Exception("Registry Import Failure: {}".format(ex))
Exception: Registry Import Failure: No module named 'cerebras.modelzoo.models.nlp.bert.venv_cerebras_ptytorch==2'
(venv_cerebras_pt) [edoyle8@cer-login-03 bert]$
```

Attempt on Groq:
Gave “conda command not found” after installing miniconda, and couldn’t get around this.

Attempt with my friend on her computer with Cerebras: (we did the same thing and hers worked while mine did not…)



Theory Homework: (the solutions were given, so I paraphrased the notes to aid my learning)

1. **What are the key architectural features that make these systems suitable for AI workloads?**
   AI accelerators enable efficient processing of intricate, memory intensive operations. These are specially designed pieces of hardware (highly engineered layout and connections of transistors on silicon chips) that enable fast tensor and matrix operations. In high dimensional problems memory is a key issue, and the architectures are designed with memory in mind, where there is intricate design of on-chip memory and off-chip memory storage logistics. The systems have various different centers of work called cores, processing units, tiles, etc depending on the system. These interconnected units enable the parallel processing of data by splitting up tasks across the various units so that you can accelerate your training and calculations.

2. **Identify the primary differences between these AI accelerator systems in terms of their architecture and programming models.**
   Sambanovas Reconfigurable Dataflow Unit (RDU) is best for large amounts of data and has multiple tiers of memory storage allowing for creative manipulation of the pathways for processing. Cerebras is based on processing elements that individually handle and store their data allowing for fast parallelization with high scalability. Graphcore has tiles that similarly store and handle their data. The processing breaks tasks into computation and communication steps, instead of continual memory saving. Groq's architecture helps to streamline procedures and minimize over use of memory features to enable consistent and efficient deterministic tasks.

3. **Based on hands-on sessions, describe a typical workflow for refactoring an AI model to run on one of ALCF's AI testbeds. What tools or software stacks are typically used in this process?**
   I did most of my work on Cerebras, so I will discuss the process for that system. All details are found here: https://docs.alcf.anl.gov/ai-testbed/getting-started/ with details in the side menu for each system. Cerebras operates through communication with the Python package modelzoo.common.pytorch.run_utils. Models from PyTorch are used and can be trained and optimized in the system. To do work, you must create a PyTorch virtual environment for Cerebras and clone the modelzoo framework into the environment for use. You then must ensure all directories are edited to work with your personal account. Once all of the details are set you can run the job.

4. **Give an example of a project that would benefit from AI accelerators and why?**
   In my lab we use AI for molecule discovery. We built a massive dataset of battery electrolyte compositions and their performance data to use with a neural network model to predict what new molecules would be optimal as battery electrolytes. In this process we talked a good deal about using DFT supplemented data. I think the AI accelerators could be helpful for parallelizing the DFT calculations to supplement the large data set in the data curating step. I'm sure it would also enable advanced ML techniques beyond basic NNs.