EECS 182 Deep Neural Networks
Fall 2026 Gireeja Ranade

Homework 9

1. Coding SSM Forward

The SSM forward pass can be implemented using recurrent and convolution approaches. In this problem, we will implement the two approaches and compare their efficiency. Particularly, you are given the following update equation:

$$\mathbf{h_{t+1}} = (W \cdot \mathbf{h_t} + U \cdot \mathbf{x}_t + \mathbf{b})$$

You are given all the input $\mathbf{x}_t$ for $t = 0, 1, \ldots, T-1$ and the initial hidden state $\mathbf{h}_0 = 0$. You need to implement the forward pass for the SSM computing all the hidden states $\mathbf{h}_t$ for $t = 1, 2, \ldots, T$. Use batch size N, input dimension D, and hidden dimension H.

We will complete the notebooks q_coding_ssm_forward_cpu.ipynb and q_coding_ssm_forward_gpu.ipynb on the CPU and GPU respectively. Setup instructions and starter code are in the HW09 code folder. Use the small default timing grids. For the GPU part, use CUDA or Apple Silicon MPS; if unavailable, use the notebook's CPU correctness fallback and state that no GPU timing was measured. Compare matched dimensions, and use the supplied warmup and device synchronization when timing.

CPU Implementation:

(a) Implement the forward pass by unrolling the SSM in time in a recurrent manner.

(b) Implement the forward pass by using a convolution-based approach. You will create the kernel K and use `nn.Conv1d` (or functional `conv1d`) to apply the kernel to the projected input $s_t = Ux_t + b$. Hint: You can create the kernel using a divide-and-conquer method. Assume T is a positive power of two, include the bias at every step, and use causal padding.

(c) Compare the runtime of the two approaches and explain your findings.

 All Rights Reserved. This may not be publicly shared without explicit permission.

GPU Implementation:

(d) Now implement the forward pass on the GPU. You can copy your existing implementation to the GPU notebook.

(e) Compare the runtime of the two approaches and explain your findings.

(f) To optimize the convolution-based approach, we will constrain W to be diagonal and leverage depthwise convolutions. Implement the new forward pass under this constraint and now compare the runtimes against a diagonal recurrence using elementwise multiplication.

 © Faculty teaching EECS 182, Fall 2026. All Rights Reserved. This may not be publicly shared without explicit permission.

2. Justifying Scaled-Dot Product Attention

Suppose $q, k \in \mathbb{R}^d$ are two random vectors with $q, k \overset{iid}{\sim} N(\mu, \sigma^2 I)$, where $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^+$. In other words, each component q_i of q is drawn from a normal distribution with mean μ_i and standard deviation σ, and the same is true for k.

Attention applies softmax to query–key dot products. When their variance grows with dimension, softmax can become more concentrated and its gradients smaller. This problem motivates scaling the dot products to control their variance.

(a) Define $\mathbb{E}[q^T k]$ in terms of μ, σ and d.

(b) Considering a practical case where $\mu = 0$ and $\sigma = 1$, define $\text{Var}(q^T k)$ in terms of d.

(c) Continue to use the setting in part (b), where $\mu = 0$ and $\sigma = 1$. Let $s > 0$ be a deterministic scaling factor on the dot product. Suppose we want $\mathbb{E}[\frac{q^T k}{s}]$ to be 0, and $\text{Var}(\frac{q^T k}{s})$ to be 1. What should s be in terms of d?

3. Argmax Attention

Recall from lecture that we can think about attention as being *queryable softmax pooling*. In this problem, we ask you to consider a hypothetical argmax version of attention where it returns exactly the value corresponding to the key that is most similar to the query, where similarity is measured using the traditional inner-product. In the notation below, argmax denotes the one-hot encoding of the maximizing index.

 All Rights Reserved. This may not be publicly shared without explicit permission.

(a) Perform **argmax attention** with the following keys and values:

Keys:

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 3 \\ 4 \end{bmatrix}, \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

Corresponding Values:

$$\left\{ \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \right\}$$

using the following query:

$$\mathbf{q} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

What would be the output of the attention layer for this query?

Hint: For example, $\text{argmax}([1, 3, 2]) = [0, 1, 0]$

(b) Note that instead of using *softmax* we used `argmax` to generate outputs from the attention layer. **How does this design choice affect our ability to usefully train models involving attention?**

(Hint: think about how the gradients flow through the network in the backward pass. Can we learn to improve our queries or keys during the training process?)

 All Rights Reserved. This may not be publicly shared without explicit permission.

4. Hardware Check: Get Your Machine Ready for the Project

You must understand your hardware to do a good job on the project. Set up your own computer and run the supplied stress test to find the largest model sizes that finish the fixed inference and training workloads below. **Submit only the automatically generated `hardware-results.csv`.** No written answers, screenshots, environment dump, or edited code are required.

(a) **Set up the environment and verify your device.** Download `hardware_check.py` from the HW09 code folder into a new working folder. Use Python 3.11 or 3.12 and create an environment:

```
python -m venv .venv
```

Use `python3` on macOS/Linux if needed. In all commands below, replace `PY` with the environment's Python executable:

```
Windows PowerShell: .venv\Scripts\python.exe
macOS/Linux: .venv/bin/python
```

Install the PyTorch build selected by the official installation selector for your OS and hardware: replace the selector's leading `pip` or `pip3` with `PY -m pip`, keeping its package names and index option. Then run:

```
PY -m pip install "transformers==5.21.0" "accelerate==1.15.0"
PY -m pip install "peft==0.21.0" psutil
PY hardware_check.py --mode setup --device auto
```

The setup must complete its matrix-product check. The runner chooses CUDA on a supported Windows/Linux installation, MPS on Apple Silicon, or CPU otherwise. It records your OS, device, RAM, and accelerator memory; Apple memory is labeled as unified. Platform-specific instructions are in the hardware setup guide.

(b) **Launch the model-size stress test.** Run this command in the same folder:

```
PY hardware_check.py --suite --device auto --download
```

It downloads checkpoints as needed and tests the text backbone of the Qwen3.5 models at 0.8B, then 2B, then 4B, then 9B parameters. It also independently tests Ling-3.0-tiny and Mellum2.1-12B-A2.5B-Thinking, subject to the same budgets. These mixture-of-experts models have 7.9B/1.3B and 12B/2.5B total/active parameters, respectively; memory must hold the total weights. Each configuration runs in a fresh process with seed 0, batch size 1, and the same 128 input tokens. The suite launches:

- **FP32 inference:** one warm-up token, then 16 greedily generated tokens.
- **Quantized inference:** the same workload with packed INT8 linear weights on CPU, plus an FP32 CPU control when your main device is an accelerator. The runner checks that quantization actually occurred.
- **FP32 full fine-tuning and LoRA:** two complete AdamW updates at learning rate 10^{-4}, first on all text-model parameters, then with a frozen base and rank-8 adapters. The runner supplies the causal-language-model labels and performs forward, backward, and optimizer steps.
- **Offloaded inference:** the same inference workload with weights staged from disk to CPU. Additional CUDA offloading and 4-bit options are listed in the setup guide.

The CPU quantization and disk-staging runs also work as the specified fallback on Macs; their results are labeled CPU. The runner stops increasing a Qwen configuration after its first resource or backend limit; this does not skip the independent Ling and Mellum checks. If all attempted sizes finish, it records completion; do not deliberately cause a failure.

 All Rights Reserved. This may not be publicly shared without explicit permission.

(c) **Submit the measured limits.** Submit the single CSV produced by the runner. It records the machine resources, model/revision, configuration, completion or stopping status, elapsed time, and memory use. These are the evidence needed to choose a feasible project model. If the smallest model cannot finish, submit the recorded failure and seek setup help from course staff. Credit is for performing the check, not for owning hardware that fits a larger model. Exit code 2 indicates a recorded resource/backend limit and is a valid result; exit code 1 indicates a setup or runner error to fix and rerun before submission.

Keep the test bounded. The suite allows at most 30 minutes of benchmark runs and 3 minutes per configuration, excluding downloads, and enforces RAM and storage limits. The host-RAM cap is at most 95% of initially available RAM, with a 1 GiB reserve. It pauses 5–30 seconds between runs for memory recovery, outside benchmark timing. Keep the defaults unless you understand the available capacity; do not run unrelated heavy jobs during the test.

5. Read About Attention, S4, and Mamba-2

Read the following three required readings:

- Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin. "Attention Is All You Need." *NeurIPS*, 2017.
- Ré, Fu, Goel, and Saab. "Simplifying S4." *Hazy Research blog*, June 21, 2022.
- Gu and Dao. "State Space Duality (Mamba-2) Part I – The Model." *Research blog post*, May 31, 2024.

Write out three notable things about each required reading. You can comment on the figures, the different new techniques used, any implications of the reading etc.

6. Homework Process and Study Group

Citing sources and collaborators is an important part of academic work.
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student IDs. (For a homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework?**

Contributors: Matteo Guarrera, Mert Cemri, Sizhe Chen, Gireeja Ranade, Anant Sahai, Naman Jain, Qiyang Li, Sheng Shen, David M. Chan, Saagar Sanghavi.

 All Rights Reserved. This may not be publicly shared without explicit permission.