

Assignment 2 Handout

Pre-training a Base Model from Scratch

Scalable AI: Bridging Theory, Understanding, and Practice

EE 290 / 194 · Spring 2026

Released	February 11, 2026
Due	February 25, 2026
Where you work	Course-provided node, $8 \times$ H100 per group. Use only this allocation.
Compute policy	No external compute. No extra GPUs. You may use the full two-week window on your assigned node.
Grading split	Part A: learning notebooks, 40% Part B: base model quality, 60%
What you submit	Notebook artifacts for Part A + report, configs, logs, and NeMo Evaluator outputs for Part B.
Checkpoint	Do not upload the checkpoint by default. You must save it and be able to provide it if we ask to verify evals.
How to submit	Submit a single ZIP on Gradescope.

High-level goal

In two weeks, your team will pre-train a **base language model from scratch** and produce **base model evaluations**. This is an end-to-end assignment: data → tokenizer → model → training → evaluation.

Core requirement: your model must use a **Nemotron-Nano-V3-based architecture** as covered in lecture and lab. You choose the scaling and training plan.

Part A: Learning notebooks, 40%

This part is about building practical skills that you will use throughout the course. The notebook work does **not** need to feed into your final pre-training project.

Notebook

NeMo Curator lab: `nemo-curator-lab.ipynb`.

What to do:

- Run the notebook end-to-end.
- Answer the questions in the notebook.
- Save the requested outputs and short written responses.

What to submit:

- The completed notebook file.
- Any generated artifacts referenced by the notebook.

Part B: Pre-training from scratch, 60%

Compute constraints

You must use only your course-provided node with $8 \times \text{H100}$. This is a fixed compute budget. You are expected to make scaling decisions under this constraint.

Your main decision: the scaling setting

Under a fixed compute budget, you must choose a training plan. Your choices can include:

- a larger model and fewer tokens,
- a smaller model and more tokens,
- a different sequence length,
- changes to batch size, optimizer, and schedule,
- data mixture and filtering decisions.

Your objective is to maximize base model quality under the fixed compute budget. In practice, higher quality often comes from consuming more high-quality tokens, so optimize your data pipeline and training efficiency accordingly.

There is no single correct choice. We care about the final base model quality you achieve within the available compute.

Model requirements

1. **Nano-V3-based architecture.** Keep the overall model family consistent with Nemotron-Nano-V3. You must choose the depth and width and report them clearly.
2. **From scratch.** Random initialization. No continuing pre-training from an existing checkpoint.
3. **Tokenizer.** You must select or train a tokenizer and report vocabulary size, training data, and key settings.
4. **Data.** You may use any legally-usable data sources. Document sources and describe any filtering, deduplication, or quality controls.
5. **Training stack.** Use any infrastructure or framework you want, as long as you train on the course node.
6. **Evaluation must use NeMo Evaluator.** All reported benchmark numbers for grading must come from NeMo Evaluator runs.
7. **Checkpoint retention.** Save your final checkpoint and tokenizer. You do not need to submit the checkpoint, but you must be able to provide it to course staff on request for verification.

What we grade

We grade Part B **only on base model quality**, measured by the evaluation results you submit. While we do not assign points for throughput, MFU, or tokens/day directly, better systems performance can help you train on more tokens within the same compute, which can improve your model quality. Please spend as much time as you can on optimizing your system performance prior to launching your training run.

Required evaluations

You must run evaluations with **NeMo Evaluator** and submit the raw outputs.

At minimum:

- **Intrinsic:** validation loss or perplexity on a held-out split of your pre-training mixture.

- **Extrinsic:** a zero-shot base-model benchmark suite using NeMo Evaluator. Use tasks that make sense for a base model.

Submission rule: include the exact NeMo Evaluator command or config used to produce the numbers.

Deliverables and submission format

Submit a single ZIP on Gradescope. Names are flexible, content is not.

```
submission/
• partA/
  - completed notebook, plus any notebook artifacts
• partB/
  - report.pdf
  - train/ training logs, configs, and summaries
  - tokenizer/ tokenizer files and settings
  - eval/ raw NeMo Evaluator outputs and an eval_summary.pdf
  - repro/ one command or script that can reproduce training and evaluation given access to the node and data
  - checkpoint_availability.txt where the checkpoint is saved and how to provide it if requested
```

Required report structure

Keep it concise: **2–6 pages**, appendix allowed.

1. **Architecture and scaling justification.** Report the full Nano-V3-based config, estimate parameter count, and justify why you chose this size under the fixed compute budget. Explain your tradeoff between model size and token budget.
2. **Tokenizer.** Vocabulary size, how it was trained or selected, and why.
3. **Data.** Sources, preprocessing, filtering, deduplication, and final token counts.
4. **Training run.** Hardware, wall-clock time, stability issues, and total tokens consumed.
5. **Evaluation.** NeMo Evaluator setup and results. Include intrinsic validation metrics and the extrinsic benchmark suite.
6. **Reproducibility and checkpoint retention.** Exact commands, configs, and where the checkpoint is stored.

Grading rubric

Component	Weight
Part A: learning notebooks	40%
Part B: base model quality from NeMo Evaluator results	60%

Notes

- This is a base-model assignment.
- We may request your checkpoint and rerun evaluations.

- Be explicit about what you trained, what data you used, and what compute you consumed.