CS4681 - Advanced Machine Learning Project Assignment- Progress Evaluation

J. HARISMENAN
210207E

----

1. Project Planning

1.1 Project Objectives

This project aims to conduct a rigorous evaluation of embedding normalization techniques—specifically, the addition of a LayerNorm or RMSNorm layer immediately following the token embedding and positional embedding lookup—in GPT-3–style decoder-only Pre-LayerNorm Transformers. The overarching objectives include:

- **Quantitative Assessment:** Measure the impact of embedding normalization on macro zero- and few-shot accuracy across a comprehensive suite of language tasks under strictly controlled, equal-compute training conditions.
- **Stability Analysis:** Investigate whether embedding normalization improves training stability, particularly in settings showing early divergence or instability.
- **Trade-off Evaluation:** Characterize the trade-offs between potential stability improvements and possible regressions in accuracy or calibration, guiding decisions on whether embedding normalization is justified.
- **Best Practice Guidance:** Produce actionable recommendations regarding embedding normalization use in large Transformer training pipelines supported by reproducible, statistically robust empirical evidence.

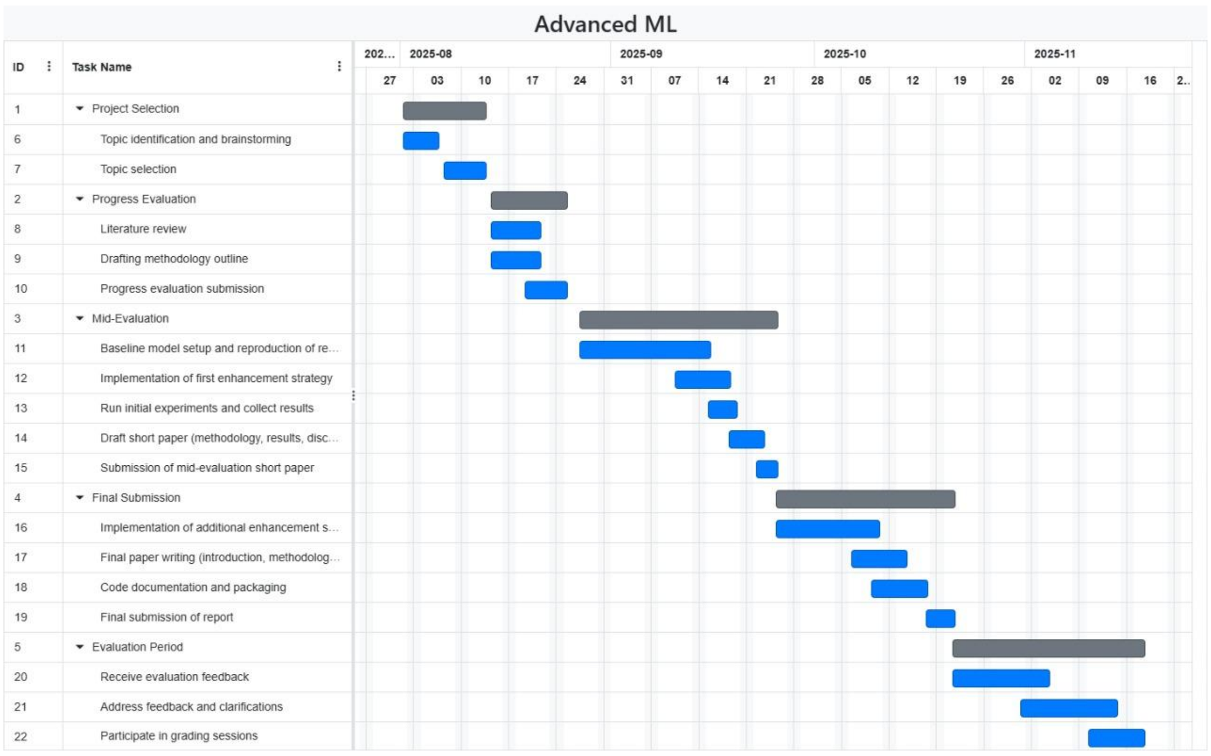
1.2 Scope and Deliverables

The scope of this project is bounded as follows:

- **Baseline Model:** A GPT-3–style, decoder-only Transformer using Pre-LayerNorm, trained on next-token prediction with learned token and absolute positional embeddings; tied input/output embeddings; AdamW optimizer with warmup and cosine decay; fixed tokenizer and data pipeline.
- **Embedding Normalization Variants:** Introduce a single normalization layer (LayerNorm or RMSNorm) applied immediately after the embedding lookup and positional addition, before the first Transformer block, keeping all other architectural and training settings identical.
- **Experimental Controls:** Precisely matched model configurations, hyperparameters, data preprocessing, and training token budgets (e.g., 50 billion tokens).

- **Evaluation Protocol:** Use LM-Eval or T0-Eval frameworks with fixed prompts for zero- and few-shot evaluation metrics, including macro accuracy, perplexity, and calibration scores.
- **Deliverables:**
    - Well-documented training logs and configurations for reproducibility.
    - Statistical analysis reports including effect sizes, confidence intervals, and multiple-hypothesis corrections.
    - Diagnostic visualizations detailing embedding norm behavior, gradient statistics, and inference calibration.
    - A conference-ready research paper summarizing methodology, results, and practical recommendations.

1.3 Timeline and Resource Allocation



ID	Task Name
1	Project Selection
6	Topic identification and brainstorming
7	Topic selection
2	Progress Evaluation
8	Literature review
9	Drafting methodology outline
10	Progress evaluation submission
3	Mid-Evaluation
11	Baseline model setup and reproduction of re...
12	Implementation of first enhancement strategy
13	Run initial experiments and collect results
14	Draft short paper (methodology, results, disc...
15	Submission of mid-evaluation short paper
4	Final Submission
16	Implementation of additional enhancement s...
17	Final paper writing (introduction, methodolog...
18	Code documentation and packaging
19	Final submission of report
5	Evaluation Period
20	Receive evaluation feedback
21	Address feedback and clarifications
22	Participate in grading sessions

2. Methodology Outline

Objective

The principal objective of the methodology is to rigorously evaluate embedding normalization applied directly on token embeddings before entry into the first Transformer block within GPT-3–style architectures, with a focus on:

- Verifying stability benefits during training at scale.
- Evaluating impact on zero-shot and few-shot downstream task performance.
- Quantifying calibration and inference behavior effects.
- Determining when embedding normalization should be recommended or avoided.

Hypotheses

- **H0 (Null):** Embedding normalization does not affect zero-/few-shot macro accuracy when training is stable without it.
- **H1a (Risk Hypothesis):** Embedding normalization decreases zero-/few-shot accuracy despite smoothing or stabilizing training dynamics [referencing Le Scao et al., 2022].
- **H1b (Conditional Benefit):** If baseline training shows instability or early divergence, embedding normalization may increase stability but at the potential expense of some downstream performance, necessitating a stability-accuracy tradeoff analysis.

Experimental Design

- **Model Variants:**
    - *A (Baseline):* Token embeddings plus learned positional embeddings directly serve as input to the first Transformer block, no extra normalization.
    - *B (+Embedding-Norm):* Add a single normalization layer (LayerNorm or RMSNorm) immediately after summing token and positional embeddings, before the first Transformer block.
- **Normalization Options in Micro-study:**
    - LayerNorm with typical epsilon (1e-5) matching internal Pre-LN layers.
    - RMSNorm with epsilon ~1e-6 and trainable scale γ without bias.
- **Invariants and Controls:**
    - Identical architecture (layer counts, embedding sizes, attention heads).
    - Fixed optimizer parameters (AdamW with $\beta1/\beta2/\varepsilon$), weight decay, warmup and cosine decay schedules.
    - Consistent data sampling, deduplication, sequence length, and batch size.
    - Fixed tokenizer, vocabulary, and special token usage.
    - Same random seeds for parameter initialization, data seed, and dropout.

Training Controls and Budget

- Fixed total tokens processed per variant (e.g. 50B tokens).
- Monitoring to confirm effective tokens matched exactly when using sequence packing.
- No early stopping to prevent selection bias.

Evaluation Protocol

- Use LM-Eval / T0-Eval with pinned version and commit to ensure consistent prompt templates and scoring.
- Evaluate zero-shot and selected few-shot task accuracy, aggregating into macro metrics.
- Metrics:
    - Primary: Macro-averaged zero/few-shot accuracy with 95% bootstrap confidence intervals.
    - Secondary: Validation perplexity, Expected Calibration Error (ECE), Brier score, inference throughput.
- Disable sampling; fixed FP precision in evaluation logits.
- Optional length scaling tests near max input length to evaluate brittleness.

Diagnostics and Metrics Instrumentation

- Loss, gradient norms, and second moment statistics monitored stepwise.
- Embedding norms distribution dissected by token class (alphabetic, numeric, punctuation, EOD).
- Cosine similarity analyses over embeddings during training to detect oversmoothing by norm layers.
- Residual norm distribution at first Transformer block input compared across variants.
- Logit magnitude distribution and softmax temperature sensitivity to reveal calibration drift.
- Confirm stable data sampling proportion and deduplication rates.

Statistical Analysis

- Use ≥3 random seeds per condition; 5 preferred for robustness.
- Paired bootstrap testing comparing macro accuracy across each seed pair (baseline versus embedding-norm).
- Correct per-task multiple tests with Holm-Bonferroni procedure.
- Report means, standard deviations, and confidence intervals for performance deltas.
- Pre-register minimum detectable effect (e.g., 0.3–0.5 macro points).

Decision Criteria

- Choose baseline if embedding normalization shows statistically significant accuracy regression with stable baseline training.
- Adopt embedding normalization only if it solves instability or divergence and stability benefits outweigh accuracy/calibration cost per pre-registered criteria.
- Use calibration and throughput as tie-breakers if accuracy is statistically indistinguishable.

3. Literature Review

Direct Evidence on Embedding Normalization

A definitive controlled ablation study at the ~1.3 billion parameter scale by Le Scao et al. [1] demonstrated that adding embedding normalization (a single normalization layer immediately post embedding lookup) significantly reduces zero-shot generalization accuracy under matched computation and evaluation conditions. Their recommendation is to avoid embedding normalization by default unless required to stabilize an unstable baseline training run.

Cross-Implementation Transferability

Narang et al. [5] show that changes to normalization and similar architectural tweaks often fail to transfer across different codebases and tasks when all other factors are held constant. This underscores the importance of reassessing embedding normalization effects explicitly within each distinct implementation and training setup rather than assuming prior positive or negative findings generalize.

Related Normalization Studies Within Transformer Blocks

The BLOOM project [2] performed ablations on LayerNorm versus RMSNorm inside Transformer blocks at smaller scales (1.3–6.7B parameters) and cautioned about scale dependencies; these studies do not directly address embedding normalization immediately post embeddings but suggest normalization choice influences stability and throughput.

Similarly, Teuken7B [4] explored RMSNorm vs LayerNorm variants and other stabilizers in a multilingual 7B setting. While it offers insights about stability and throughput trade-offs, it does not isolate embedding normalization layers.

Methodological and Evaluation Discipline

The LM-Eval and T0-Eval standardized evaluation frameworks established by Wang et al. [3] emphasize meticulous prompt template control, fixed evaluation commits, and matched compute baselines to adequately detect subtle architecture or training objective effects. These practices are critical given the small effect sizes typical for normalization-related micro-changes.

Large-Scale Training Best Practices

Gopher [6] and similar large-scale training efforts detail reproducible baseline setups including data curation, optimizer recipes, and evaluation spread that serve as exemplars for conducting ablation studies such as embedding normalization. Chinchilla’s [7] compute-optimal scaling laws highlight the dominant importance of precise token budget matching in attributing performance differences to architectural changes rather than data scale or training duration variations.

Synthesis

Taken together, the literature presents moderate but compelling evidence that embedding normalization is more likely detrimental to zero-/few-shot performance for stable GPT-3–style training at large scale, supporting a conservative stance to use embedding normalization only when a demonstrable stability benefit is warranted and rigorously validated by matched experiments.

References

[1] Le Scao et al., “What Language Model to Train if You Have One Million GPU Hours?” (controlled ablations; embedding-norm hurt zero-shot; rigorous matched protocol).

[2] BLOOM (176B paper; ablations on normalization variants, scale-dependent effects).

[3] Wang et al., “What Architecture and Pretraining Objective Work Best for Zero-Shot Generalization?” (compute-matched comparisons; LM-Eval/T0-Eval discipline).

[4] Teuken7B, multilingual study on normalization micro-changes including RMSNorm.

[5] Narang et al., “Do Transformer Modifications Transfer Across Implementations and Applications?” (effects of normalization rarely transfer without retuning).

[6] Gopher training and evaluation protocol, reproducibility focus.

[7] Chinchilla, compute-optimal scaling laws; necessity of token-budget matching for fair comparison.