# Project Proposal

High-Performance Machine Learning (HPML)

## 1. Project Title

**Accelerating BERT Fine-Tuning for Question Answering on SQuAD v2 via CUDA-Level Optimization and Profiling**

## 2. Team members (2)


- Nikhil Arora (na4063)
- Devanshi Bhavsar (dnb7638)


## 3. Goal / Objective

**North-star:** How much *hardware-level efficiency* can we unlock in **BERT-base** fine-tuning on **SQuAD v2** using modern compiler/attention kernels, and do **PEFT** methods (LoRA/BitFit) still help once kernels are fused—*without* losing accuracy?

### 3.1 Reference-guided evaluation goals:

- **Throughput:** Improve over a baseline FP16 run via SDPA+`torch.compile`+dynamic padding, targeting *1.3–2.0*× ranges reported for transformer workloads.
- **Memory:** Reduce peak VRAM via 8-bit AdamW and/or checkpointing; quantify VRAM vs. step-time trade-off.
- **Quality:** Keep F1/EM within *about 1 point* of the baseline; analyze any accuracy/efficiency trade-off.

## 4. Challenges

- Pinpointing true CUDA bottlenecks (attention vs. FFN, kernel launch overhead).
- Preserving accuracy with fused attention, mixed precision, and compiler fusion.
- Fairly comparing PyTorch compile backends (Inductor/NVFuser) under identical training conditions.
- Staying within T4 (16 GB) memory while collecting useful Nsight traces.

## 5. Approach / Techniques

**Experiment Matrix (named configs).** Each variant toggles one factor to isolate its effect.

| | |
|---|---|
| **A1** | Baseline full fine-tune (FP16). |
| **A2** | A1 + AMP/TF32 + SDPA (fused attention path). |
| **A3** | A2 + `torch.compile` (Inductor). |
| **A4** | A3 + dynamic padding/bucketing (length-based). |
| **B1** | A3 + **LoRA** (r=8) on attention projections. |
| **B2** | A3 + **BitFit** (bias-only). |
| **B3** | A3 + **freeze last 4 encoder layers**. |
| **C1** | A3 + **8-bit AdamW** (bitsandbytes). |
| **C2** | A3 + **gradient checkpointing**. |
| **D1** | A3 with **backend swap**: Inductor vs. NVFuser (`aot_eager` if stable). |

### 5.1 What's new (our contributions).

- **Stepwise systems gains:** quantify SDPA vs. `torch.compile` and their combination (A1→A4).
- **PEFT × compiler interaction:** do LoRA/BitFit still help once kernels are fused (A3 vs. B1/B2/B3)?
- **Practical T4 memory recipes:** VRAM savings and runtime cost of 8-bit AdamW & checkpointing (C1/C2 curves).
- **Backend insights:** Inductor vs. NVFuser for BERT QA fine-tuning (kernel count, step time, SM occupancy; D1).

## 6. Implementation details

| | |
|---|---|
| **Hardware:** | **T4 (16 GB)** primary; **A100 (40 GB)** for stretch/polish runs. |
| **Software:** | PyTorch **2.x** (`torch.compile` Inductor/NVFuser, SDPA); HF **Transformers + PEFT**; `bitsandbytes`; *reuse* HF `run_qa.py`/Trainer pipeline + official `torch.profiler`/Nsight examples. |
| **Dataset:** | SQuAD v2 (150k Q&A pairs incl. unanswerable). |
| **Training:** | seq_len=384 (512 for stress), batch=8–16 (accum. to eff. 32–64), AdamW, 3 epochs, FP16 (TF32 on A100). |
| **Metrics (every run):** | **Quality:** F1/EM (dev).<br>**Perf:** tokens/s, steps/s, step time.<br>**Memory:** peak VRAM, optimizer-state size. |
| **Profiling:** | `torch.profiler` (kernel counts); Nsight Systems (timeline); Nsight Compute (SM occupancy, achieved memory BW in GB/s). |
| **Assumptions/Protocol:** | Warmup 100 steps; measure next 500; fixed dataloader workers & cuDNN benchmarking; A1 is baseline and all deltas reported w.r.t. A1; **fixed random seed and average over 3 runs for throughput/latency**. |

## 7. Demo planned

- **Gradio QA demo:** user enters paragraph + question, model highlights answer and shows latency.
- **Performance dashboard:** stepwise A1→A4 table (tokens/s, VRAM, F1/EM), PEFT vs. Full-FT table, VRAM vs. step-time plot (C1/C2), two Nsight visuals.

## 8. References (papers + how we build over them)

| Paper | Contribution (what they did) | Our extension (what we add) |
|---|---|---|
| Hu et al. (ICLR 2022), LoRA | Low-rank adapters enable parameter-efficient fine-tuning. | Apply LoRA to QA and test *with* compiler fusion; measure GPU effects (kernel counts, occupancy) vs. Full-FT. |
| Ben-Zaken et al. (2022), BitFit | Bias-only tuning with small accuracy drop. | Compare BitFit on the compiled stack; quantify accuracy/memory trade-offs for QA. |
| Dao et al. (2023), FlashAttention-2 | I/O-aware attention with better parallelism; strong A100 training throughput. | Use fused attention path (SDPA; FA on A100 if feasible); profile bandwidth/utilization improvements. |
| Dettmers et al. (NeurIPS 2022), 8-bit AdamW | Optimizer-state quantization reduces VRAM with similar accuracy. | Integrate 8-bit AdamW; report VRAM savings and runtime impact (C1 curve) on T4/A100. |
| He et al. (2021), DeBERTa-v3 | Strong SQuAD v2 dev baseline (F1∼91.5/EM∼89). | Use as accuracy context; our focus is efficiency on BERT-base with modern kernels. |