Samuel Jordan

Patrick Woods

Dr. Abolfazl Razi

CPSC4420: Artificial Intelligence

October 27, 2025

Project Report 1: Lightweight / On-Device LLMs & VLMs

Project Description

The goal of this project is to develop and evaluate compact LLM/VLM pipelines optimized for edge deployment on resource-limited devices such as smartphones, drones, and embedded systems. This aligns with the broader push toward on-device AI, where models must maintain high performance while operating under strict constraints of memory, compute power, latency, and energy.

The motivation stems from the increasing need for efficient AI that balances capability and deployability. While large-scale models (like GPT-4 or Gemini) dominate performance benchmarks, their compute, energy, and hardware demands make them impractical for edge applications. This project explores the design principles, compression methods, and quantization strategies that allow smaller models to approach large-model performance efficiently.

Tools and Methods

Our recent contribution to the project leverages a combination of literature analysis and hands-on model experimentation using the following tools and techniques:

- PyTorch for model implementation, quantization, and testing.
- Hugging Face Transformers for running compact pretrained models (e.g., SmolLM-135M).
- Quantization methods (INT8 and 4-bit) to reduce model precision and measure accuracy vs. efficiency trade-offs.
- Performance benchmarking (latency, memory) using time and CUDA profiling tools.

- Jupyter Notebooks for visualization, reproducibility, and documentation.

Our background reading and research also covers key compression strategies including pruning, knowledge distillation, and parameter-efficient fine-tuning (PEFT) to contextualize quantization within the broader landscape of efficient AI models.

Project Activities

Task 1: Background Reading and Review of Compact Models

- Conducted a literature review on the evolution of LLMs from large-scale systems (GPT, PaLM, LLaMA) to compact architectures.
- Summarized key trade-offs between large vs. small models in terms of scalability, accuracy, energy, and hardware needs.
- Studied compression techniques: pruning, quantization, knowledge distillation, and parameter-efficient fine-tuning.
- Reviewed popular small models such as DistilBERT, ALBERT, TinyBERT, MiniLM, and MobileBERT, as well as modern compact LLMs (Mistral, Phi, Gemma, SmolLM).
- Produced a multi-section report comparing design principles and efficiency benchmarks across these architectures

Task 2: Quantization Fundamentals & Baseline Testing

- Implemented a simple INT8 quantizer in PyTorch that converts a fully connected layer's weights to 8-bit integers and measures accuracy degradation before and after quantization.
- Achieved <0.05% accuracy loss, validating that low-bit quantization can preserve model performance with significant memory savings.
- Loaded and executed SmolLM-135M in a Jupyter Notebook to establish baseline metrics:
    - Average latency: ~16-20 seconds per 250-token generation
    - GPU memory usage: negligible (CPU-only run due to system limitations)
- Tested prompts on a few varied topics to observe text coherence and response consistency.

Results and Observations

Our literature review showed that smaller models can retain up to 90–97% of their larger counterparts' accuracy while being up to 60% faster and 40–80% smaller. Efficient architectures like ALBERT and DistilBERT prove that intelligent design choices and distillation can drastically reduce redundancy without major losses in performance. As far as our implementation, the quantization prototype showed that per-tensor INT8 quantization effectively reduces model precision with negligible performance degradation. Our SmolLM-135M tests showed baseline outputs which were coherent and factually reasonable, confirming that compact LLMs can achieve high-quality text generation despite limited parameter counts (~135M). These results establish a strong foundation for future phases of the project.

Challenges and Issues

Our largest current issue comes from hardware limitations and configuration. The environment used for testing our implementation (a Dell Latitude personal laptop) lacked an active CUDA runtime or dedicated GPU drivers, which restricted tests to CPU-only performance. Using other available devices which fully supported FP16, the LLM was fully supported out of the box, so any attempts to optimize and cut down on resource usage such as quantization actually just slowed its processes. Having an edge device or simulated equivalent to test on and optimize for would be valuable for future implementation assignments.

Next Steps and Timeline

While our weekly tasks are assigned and guided by our supervisor, our upcoming work will focus on fulfilling these deliverables through continued implementation and evaluation. We will likely conduct and document performance benchmarks on simulated edge platforms (e.g., Jetson Nano, Raspberry Pi) to compare efficiency trade-offs under realistic hardware constraints. Additionally, we will propose design improvements that enhance deployability across different edge environments and conclude the project with a code submission containing our quantized and adaptive model implementations.