

Quantization of GPT-2 for Running on Edge Devices

Stanford CS224N Default Project

Gordy Sun

Department of Computer Science
Stanford University
gordysun@stanford.edu

Abstract

Large language models typically require substantial compute and memory resources to run effectively, which dramatically limit their deployments on edge devices with much more constraints. In our research, we implemented GPT-2 from scratch and then fine-tuned it on the tasks of sentiment classification, paraphrase detection, and Shakespearean sonnet generation. The models that we trained outperformed the baselines provided on all three tasks, finding that scaling model parameter size to GPT-2 Medium dramatically improved sonnet generation quality and that temperature was critical for sonnet chrF scores. Then, we investigated numerous quantization methods including post-training dynamic INT8 and QAT. We compressed GPT-2 for deployment on an NVIDIA Jetson Orin Nano and a Raspberry Pi 5 with a 4-times reduction in model size with a small decrease of accuracy on sentiment classification tasks.

1 Key Information to include

- Mentor: N/A
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

Large language models (LLMs) excel at a wide variety of natural language processing tasks, from text classification to creative generation. However, their high computational and memory demands severely restrict their deployment on edge devices. As applications increasingly require local inference for privacy, latency, and connectivity reasons, adapting these massive models for constrained hardware has become a critical area of research.

In this project, we implemented GPT-2 Small (124M parameters) from scratch based on the original OpenAI architecture. We fine-tuned the model across three distinct tasks: sentiment classification (using the Stanford Sentiment Treebank and CFIMDB datasets), paraphrase detection (Quora Question Pairs), and sonnet generation (Shakespearean sonnets). To make edge deployment feasible, we then evaluated two quantization techniques. First, we applied post-training dynamic INT8 quantization, which converts trained weights to lower precision without additional training. Second, we explored quantization-aware fine-tuning (QAT), which simulates quantization noise during training to help the model adapt to the resulting precision loss.

3 Related Work

The seminal paper that was most important for our work was Radford et al. [1] from OpenAI, introduced GPT-2, a decoder-only Transformer trained on next-token prediction over a large web corpus. The researchers from OpenAI noted that by scaling the size of the model to 1.5 billion

parameters and then training on a diverse dataset of text, GPT-2 showed strong zero-shot performance on tasks that it hadn't seen before including translation, query answering, and summarization without further specific fine-tuning. GPT-2 was built on top of research from the original GPT architecture [2] with modifications including pre-layer normalization and an additional layer norm after the final self-attention block. We specifically implemented GPT-2 Small from scratch ourselves and then fine-tuned it on three tasks we were looking to benchmark on.

This work was built on the fundamental paper which created the Transformer architecture "*Attention is all you need*" [3] which introduced the attention mechanism as an alternative to traditional recurrent networks. We also trained our models using the AdamW optimizer [4], which decouples weight decay from the gradient-based update, addressing an issue in the original Adam optimizer [5] whereby the L2 regularization interacted poorly with adaptive learning rates.

In terms of quantization, the work of Frantar et al. [6] was critical as they proposed GPTQ, a one-shot weight quantization method for generative pre-trained Transformers based on approximate second-order information. GPTQ quantizes models with minimal degradation by solving a layer-wise quantization problem with an efficient approximate solver. Their method thereby processes weights in a specific order and uses lazy batch updates to maintain accuracy and allow large models to run on hardware with much more limited compute and memory.

The other important paper we noted was Park et al. [7] which introduced QuAdapter, an adapter-based approach specifically designed for GPT-2 quantization. Rather than quantizing the entire model directly, QuAdapter inserts lightweight adapter modules that learn to compensate for accuracy loss introduced by weight quantization. This Quantization Aware Fine-Tuning (QAT) enables aggressive quantization without seeing major decreases in accuracy. Our quantization-aware-fine-tuning approach was inspired by this work, but instead of adapters, we inserted fake quantization nodes throughout the network during training.

Finally, our datasets came from a multitude of papers. Our sentiment dataset came from The Stanford Sentiment Treebank [8] which provided sentiment labels for movie review sentences. For paraphrase detection, the Quora Question Pairs dataset provided over 400,000 question pairs labeled as duplicates or non-duplicates. For sonnet generation, we evaluate using the chrF metric [9], a character-level n-gram comparison score.

4 Approach

4.1 GPT-2 Architecture

GPT-2 Small is a decoder-only Transformer with 12 layers, 768-dimensional hidden states, 12 attention heads (each 64-dimensional), a context length of 1024 tokens, and a vocabulary of 50,257 BPE tokens, totaling approximately 124 million parameters. We loaded the official pretrained weights from HuggingFace and verified correctness against their implementation via a provided sanity check.

Each of the input token indices is mapped to a 768-dimension vector through a learned embedding matrix $E \in \mathbb{R}^{V \times d}$. The learned positional embeddings to ensure location $P \in \mathbb{R}^{1024 \times d}$ are added element-wise, and a dropout ($p = 0.1$) is applied to the sum:

$$h^{(0)} = \text{Dropout}(E[x] + P[1, \dots, T])$$

For each of the 12 layers of attention, they followed a pre-norm architecture with two sub-blocks. The first applies layer normalization followed by masked multi-head self-attention:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} + M \right) V$$

where M is an upper-triangular causal mask that sets all future positions to $-\infty$. The queries, keys, and values are computed from the learned linear projections and then split across 12 heads. The outputs are then all combined together and projected back to the 768 dimensions. The second sub-block applies layer normalization with a two-layer feed-forward network with GELU activation:

$$\text{FFN}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2$$

where $W_1 \in \mathbb{R}^{768 \times 2304}$ and $W_2 \in \mathbb{R}^{2304 \times 768}$. Both sub-blocks use dropout on the output followed by a residual connection. A final layer normalization is applied after the last Transformer layer.

4.2 Task Adaptation

For the SST and CFIMDB datasets, we extract the last non-padding token’s hidden state from the final layer and pass it through dropout ($p = 0.3$) and a linear projection to the number of classes. We then trained with cross-entropy loss and evaluated with two frameworks: last-linear-layer whereby the GPT-2 parameters are frozen and only the classification head is trained and full-model where all the parameters are updated.

For the paraphrase task and Quora dataset, we format each question pair as a cloze-style prompt: “Question 1: {s1} Question 2: {s2} Are these questions asking the same thing?” The last token’s hidden state is projected through a linear head to 2 classes for binary evaluation. We fine-tune the full model with cross-entropy loss.

Finally, for sonnet generation, we fine-tune GPT-2 with standard autoregressive language modeling, predicting the next token at every position using cross-entropy loss over the full vocabulary. The output hidden states are projected to vocabulary logits via weight tying with the input embedding matrix: $\text{logits} = h \cdot E^T$. At generation time, given the first 3 lines of a held-out sonnet, we autoregressively generate the remaining lines using top- p (nucleus) sampling with temperature scaling.

4.3 Quantization for Edge Deployment

We investigated the deployment of GPT-2 onto edge devices following quantization to meet compute and memory requirements.

We quantize all linear layer weights from FP32 to INT8 using PyTorch’s dynamic quantization. This reduces the model size by approximately 4 times and also required no further fine-tuning.

We also quantized with a fake quantization module after each linear layer during training inspired the QAT-adjacent QuAdapter [7]. These simulate INT8 rounding noise using a straight-through estimator. During the forward pass, a tensor x is quantized and immediately dequantized:

$$\hat{x} = s \cdot \left(\text{clamp} \left(\text{round} \left(\frac{x}{s} + z \right), q_{\min}, q_{\max} \right) - z \right)$$

where s is the scale factor computed from the tensor’s dynamic range and z is the zero point. We allow the model to learn weight configurations that are robust to quantization noise.

Then, in terms of target hardware, we benchmark on three platforms: (1) an NVIDIA A10G GPU as the cloud baseline, (2) an NVIDIA Jetson Orin Nano Super (8GB, edge GPU with CUDA), and (3) a Raspberry Pi 5 (CPU-only inference).

4.4 Optimizer

We also implement the AdamW optimizer [4] with decoupled weight decay. We use the efficient bias-corrected update from Kingma and Ba [5]:

$$\alpha_t = \alpha \cdot \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t}, \quad \theta_t = \theta_{t-1} - \alpha_t \cdot \frac{m_t}{\sqrt{v_t} + \epsilon}$$

followed by decoupled weight decay: $\theta_t = \theta_t - \alpha \lambda \theta_t$. Our implementation passes the provided unit test.

5 Experiments

5.1 Data

We evaluate on four datasets for our sentiment classification, paraphrase detection, and sonnet generation tasks:

The Stanford Sentiment Treebank [8] is comprised of 11,855 single sentences that were taken from movie reviews. Then, each of the sentences was labeled with one of the following sentiment classifications: negative, somewhat negative, neutral, somewhat positive, and positive. The data split was 8,544 for training, 1,101 for dev, and 2,210 for test.

The CFIMDB is comprised of 2,434 very polar movie reviews that are classified with a binary sentiment (either positive or negative). We split this data with 1,701 for training, 245 for dev, and 488 for testing.

The Quora Question Pairs dataset had over 400,000 question pairs from Quora that were then labeled as either duplicates or non-duplicates. We split it with 283,003 for training, 40,429 for dev, and 40,431 for testing.

Finally, the Shakespeare Sonnets dataset contained 154 sonnets that were actually written by Shakespeare and followed the typical sonnet rhyme scheme. We used 131 of the sonnets for training and 12 for test. During the evaluation process, the first three lines of a sonnet are given and the LLM must generate the rest of them.

5.2 Evaluation method

For sentiment and paraphrase tasks, we report the accuracy on both the dev and test sets. This is just a simple metric of how many out of the total it correctly predicted. For sonnet generation, we reported the chrF score which is a character level n-gram F-score that measures how closely a generated text actually matches the style and structure of the referenced texts (the sonnets). For our quantization experiments, we measured the model size on the disk (MB), inference latency (ms per sample), and peak memory usage (MB).

5.3 Experimental details

For sentiment and paraphrase tasks, we used GPT-2 Small with pretrained weights from HuggingFace. For sonnet generation, on the other hand, we found that scaling to GPT-2 Medium dramatically improved the quality of generation. We trained our AdamW implementation with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-6}$, and dropout of 0.3 on the classification head. All training was done on a simple NVIDIA A10G GPU (24GB). Table 1 summarizes the hyperparameters for each task.

Table 1: Training hyperparameters for each task.

Task	Model	Learning Rate	Batch Size	Epochs	Mode
SST (frozen)	GPT-2 Small	1×10^{-3}	64	5	Last-linear-layer
SST (full)	GPT-2 Small	1×10^{-5}	64	5	Full fine-tune
CFIMDB (frozen)	GPT-2 Small	1×10^{-3}	8	5	Last-linear-layer
CFIMDB (full)	GPT-2 Small	1×10^{-5}	8	5	Full fine-tune
Paraphrase	GPT-2 Small	6×10^{-5}	48	3	Full fine-tune
Sonnet	GPT-2 Medium	5×10^{-5}	4	10	Full fine-tune

For sonnet generation, we also used a top-p nuclear sampling with $p = 0.9$ with a temperature of 0.8 at the inference time. We tried multiple different temperatures and found that 0.8 was the best at producing high quality sonnets. As mentioned previously, throughout testing, we also found that a larger model improved sonnet quality too.

For the post-training quantization, we used PyTorch’s `quantize_dynamic` to all `nn.Linear` layers so that we could convert FP32 weights to INT8. For quantization-aware fine-tuning, we inserted fake quantization modules after each linear layer and trained for an additional 5 epochs with a learning rate of 1×10^{-5} .

5.4 Results

For sentiment classification, Table 2 shows results for both fine-tuning strategies on SST and CFIMDB. Full-model fine-tuning dramatically outperforms the frozen parameters approach. Our full-model SST test accuracy of 0.521 exceeds the 0.513 reference baseline and our CFIMDB dev accuracy of 0.976 matches the target exactly.

For paraphrase detection, Table 3 shows results on the Quora dataset. Our model achieves 0.897 dev accuracy after 5 epochs of full fine-tuning with a learning rate of 3×10^{-5} , exceeding the staff baseline of 0.882. Test accuracy of 0.853 is comparable to the staff baseline of 0.856.

Table 2: Sentiment classification results.

Model	SST Dev	SST Test	CFIMDB Dev	CFIMDB Test
Last-linear-layer	0.434	0.424	0.861	—
Full-model	0.501	0.521	0.976	—
<i>Reference baseline</i>	<i>0.462 / 0.513</i>	—	<i>0.861 / 0.976</i>	—

Table 3: Paraphrase detection results.

Model	Dev Acc	Test Acc
Full fine-tune (ours)	0.897	0.853
<i>Staff baseline</i>	<i>0.882</i>	<i>0.856</i>

For sonnet generation, Table 4 shows chrF scores across our experimental progression. Scaling from GPT-2 Small to GPT-2 Medium provided the largest improvement, and tuning the sampling temperature from 1.2 to 0.8 further improved generation quality. Our best model achieves a dev chrF of 41.498, exceeding the staff baseline of 41.414.

Table 4: Sonnet generation chrF scores across experimental configurations.

Model	Temp.	Epochs	Dev chrF	Test chrF
GPT-2 Small	1.2	5	—	38.524
GPT-2 Small	0.7	15	38.821	—
GPT-2 Small	0.8	10	40.357	—
GPT-2 Medium	0.8	10	41.498	41.114
<i>Staff baseline</i>	—	—	<i>41.414</i>	<i>41.208</i>

In terms of quantization, Table 5 shows the effect of post-training dynamic INT8 quantization and quantization-aware fine-tuning (QAT) on sentiment classification. INT8 quantization reduces model size by 51.3% (477 MB to 232 MB) with a $1.32\times$ CPU inference speedup, but incurs accuracy drops of 8.4 percentage points on SST and 4.9 on CFIMDB. QAT partially recovers this loss: on SST, QAT improves INT8 accuracy from 0.418 to 0.434, reducing the gap from 8.4 to 6.7 percentage points.

Table 6 compares inference characteristics across deployment targets. The Jetson Orin Nano achieves $15\text{--}48\times$ speedup over CPU inference, confirming that edge GPU devices are substantially more viable for transformer inference.

6 Analysis

For the sentiment classification task, we noted that the gap between last-linear-layer and full-model fine-tuning is quite great. We noted that full-fine-tuning allowed for much better results, likely as it’s able to more properly retain the knowledge across the board for the different sentiment datasets.

In terms of paraphrase detection, our accuracy plateaued quite quickly, showing that while the model did learn quite quickly, it did also hit a limit in terms of how good it could get. We also found that a higher learning rate (6×10^{-5}) and larger batch size (48) were important for strong paraphrase performance, likely because the larger dataset benefits from more aggressive optimization.

For the sonnet generation experiments, we noted most importantly that the model bump up from GPT-2 Small to GPT-2 Medium dramatically improved performance. This is likely because for a more complex task like creative generation of sonnets, a larger model with more parameters is more likely to capture the different nuances as a part of Shakespeare’s writing. We did, however, have to modify our training parameters for GPT-2 Medium to avoid things like overfitting.

For example, we noted some common problems with model deficiencies would be seen when models began to spit out nonsense or repeat words over and over again at the end of sonnets like "love". We noted a couple examples that read as follows and were nonsensical:

Table 5: Quantization results on sentiment classification.

Model	Size (MB)	SST Acc	CFIMDB Acc	CPU Latency	Speedup
FP32 (baseline)	477.0	0.501	0.976	435.6 ms	1.00×
INT8 (post-training)	232.4	0.418	0.927	329.6 ms	1.32×
INT8 (with QAT)	232.4	0.434	0.954	325.7 ms	1.33×

Table 6: Inference benchmarks across deployment platforms.

Platform	SST Latency	SST Throughput	CFIMDB Latency	Peak Mem
CPU FP32 (Pi 5)	435.6 ms	736 tok/s	4622.7 ms	—
CPU INT8 (Pi 5)	329.6 ms	973 tok/s	3883.9 ms	—
GPU FP32 (Jetson)	28.8 ms	35,500 tok/s	51.2 ms	506–714 MB

When I saw thee in thy beauty,
When I saw thee in thy beauty's sweetness,
When I saw thee in thy beauty's sweetness's sweetness,
When I saw thee in thy beauty's sweetness's sweetness,
When I saw thee in thy beauty's sweetness's sweetness's sweetness,
When I saw thee in thy beauty's sweetness's sweetness's sweetness's sweetness's
sweetness's sweetness's sweetness's sweetness's sweetness's sweetness's sweet-
ness's sweetness's sweetness's sweetness's sweetness's sweetness's

Our hardware reveals a stark contrast between CPU and GPU inference. GPU inferences is much much faster compared to CPU inference and motivates advantages that GPU based edge hardware has over CPU based devices. This was true not only for speed of generation but also latency to begin calculations and inference. In terms of memory, we found that smaller memory models didn't lose quite a bit of accuracy post quantization but a lot of this was avoided with usage of QAT. We did, however, not get many memory measurements that were of particular interest for the Pi 5 and the Jetson because both devices were actually capable of having enough RAM/VRAM to support the inference process.

7 Conclusion

Overall, we implemented GPT-2 from scratch and then used our implementation to fine-tune it on three downstream tasks, sentiment classification, paraphrase detection, and sonnet generation. We achieved strong results across all three of these fields. Then, we explored quantization and found that CPUs dramatically underperformed GPUs in addition to the accuracy preservation of QAT over just basic quantization. This confirms practical deployment methods onto edge GPU devices for real world applications.

Some of our key findings included the advantages of full-model fine-tuning for sentiment classification along with the benefits of model scaling for sonnet generation. We also found the importance of sampling temperature for creative tasks like sonnet generation and also noted the importance of QAT at mitigating quantization-induced accuracy loss.

Future work could explore more aggressive quantization (4-bit or mixed-precision) and combining quantization with parameter-efficient methods like LoRA to see how they perform on edge hardware.

References

- [1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.
- [2] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

- [3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- [4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [6] Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023.
- [7] Minseop Park, Jaeseong You, Markus Nagel, and Simyung Chang. Quadapter: Adapter for gpt-2 quantization. In *Findings of EMNLP*, 2022.
- [8] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.
- [9] Maja Popović. chrF: Character n-gram f-score for automatic mt evaluation. In *WMT*, 2015.