CAMELGPT: A VIABLE SMALL LANGUAGE MODEL

Lehuy Hoang
GPT Research
lehuy.hoang@gpt-research.org

Eric Kimbrell
Undergraduate Research, GPT Research
University of California, Santa Cruz
eric.kimbrell@gpt-research.org

Anshul Kulkarni
Undergraduate Research, GPT Research
San Jose State University
anshulkulkarni@gpt-research.org

ABSTRACT

We present CamelGPT, a novel approach addressing resource limitations in developing high-performing language models. CamelGPT introduces Eager Precached Dynamic Pruning (EPDP), departing from conventional training methods. Unlike post-training pruning in traditional models, EPDP incorporates pruning into both training and inference, dynamically optimizing the model architecture. This enables CamelGPT to create smaller models from the outset, delivering comparable performance to resource-intensive counterparts. EPDP efficiently utilizes computational resources by selectively retaining crucial connections, significantly reducing the model size and memory footprint without compromising performance. Experimental results demonstrate CamelGPT's effectiveness in various NLP tasks, outperforming existing large language models in computational efficiency and storage requirements while maintaining competitive accuracy. CamelGPT offers a sustainable and efficient alternative, bridging the gap for users with limited computational resources. By integrating EPDP, CamelGPT shows promise in creating resource-efficient language models, democratizing access to advanced language processing capabilities, and reducing environmental impact.

Demo

For a demonstration of our research, please visit: https://huggingface.co/spaces/gpt-research/CAMELGPT-DEMO.

1 Background

With the advent of state-of-the-art language models like GPT-3, natural language processing has witnessed a remarkable transformation, enabling applications across various domains, such as language translation, content generation, and more [BMR+20].

Despite their unprecedented achievements, large language models present significant challenges. To attain their results, these models require vast amounts of training data and employ complex architectures with billions or even trillions of parameters [BMR+20]. The training process is computationally intensive and memory-consuming, demanding access to high-performance hardware and substantial computational resources[SPS20]. Additionally, the training data itself is a crucial factor, necessitating large and diverse datasets to achieve optimal performance [SSZ+23].

The resource requirements of large language models present a formidable obstacle for many organizations and individuals. Smaller companies, startups, researchers, and students often face limitations in their access to the necessary computational power and financial resources. This inequality creates a digital divide where only a few with sufficient means can harness the full potential of these advanced language models [BHA+21]. As a result, the transformative benefits of natural language processing are not equally accessible to all, hindering innovation and progress in various domains.

Moreover, the environmental implications of training large language models cannot be ignored. The substantial energy consumption associated with running computationally intensive tasks on powerful hardware raises concerns about the ecological impact [PGL+21]. Discussions around the sustainability of training and deploying large language models have prompted researchers and developers to seek alternative approaches that can mitigate these environmental concerns [TMS+23].

While previous efforts have attempted to address the resource constraints of large language models, they often come with their limitations. Traditional pruning methods applied post-training and have achieved some success in reducing model sizes, but they may not fully optimize the model architecture during the training process [LWS+20]. Thus, a more holistic and efficient approach is necessary to produce high-performing language models with minimal resource requirements.

Through the development and evaluation of CamelGPT, we aim to democratize access to advanced natural language processing capabilities. By achieving comparable performance to large models with fewer resources, CamelGPT opens the door for a broader audience, including smaller organizations and individuals, to harness the power of language models. Moreover, the resource-efficient nature of CamelGPT addresses environmental concerns associated with large-scale model training, making natural language processing more sustainable for the future.

2 Eager Precached Dynamic Pruning

Eager Precached Dynamic Pruning (EPDP) is a revolutionary approach to accelerate transformer-based models, offering promising results in our extensive experiments. The primary motivation behind EPDP is to tackle the computational complexity inherent in transformer models, a challenge that can be cost-prohibitive for many real-world applications.

The key insight that sets EPDP apart is the recognition that not all parameters in a transformer model are equally critical for its performance. A considerable number of parameters can be safely pruned away without significantly compromising the model's effectiveness. EPDP leverages this insight to identify and dynamically remove unnecessary parameters at runtime, resulting in a considerably faster and more efficient inference process.

The EPDP process encompasses five crucial stages enumerated below.

2.1 Tokenization

The input sequence undergoes tokenization, breaking it down into subwords or wordpieces. This step is vital as it allows the model to operate on smaller units of meaning rather than individual words or characters. Tokenization enables the model to grasp the nuances of language more effectively.

2.2 Loading Cache Of Parameters

During this stage, only the most relevant parameters are loaded into the cache, based on the input token sequences. By doing so, EPDP significantly reduces the computational requirements of the model, as only a subset of the total parameters needs to be processed. This caching mechanism optimizes inference and makes the model much more resource-efficient.

2.3 Dynamic Pruning

The core of EPDP lies in the dynamic pruning of unrelated parameters from the model, leaving only the essential ones required to generate the output sequence. EPDP employs a simple yet effective pruning strategy, efficiently discarding redundant parameters and streamlining the computation process.

2.4 Calculating Remaining Parameters

The remaining parameters are calculated using a lightweight computation graph, which is constructed on the fly based on the input token sequences. This step ensures that the model generates accurate outputs while avoiding unnecessary computations, leading to a faster and more precise inference process.

2.5 Parsing Tokens

Finally, the generated output tokens are parsed to form a coherent and fluent text sequence. While seemingly trivial compared to the previous steps, this stage is critical in ensuring that the output sequence is grammatically correct and semantically consistent.

3 Model

In this study, we present the development and training of an instruction-tuned model, known as CamelGPT-mini, serving as a proof of concept for the novel architecture proposed by CamelGPT2. Our model is specifically tailored for text generation based on instructions and is trained on a private dataset6.1, comprising numerous instructions, each consisting of a prompt and its corresponding response.

CamelGPT-mini is a variant of the transformer architecture[VSP+17], carefully optimized for instruction-based text generation. It employs a 5-step Eager Precached Dynamic Pruning (EPDP) architecture2, which effectively streamlines the computational complexity of the model and enhances its resource efficiency. This architecture is composed of a total of 56,000 parameters, striking a balance between model size and performance.

To train the model effectively, we employed a masked language modeling objective. During training, a fraction of the input tokens was randomly replaced with a special [MASK] token, indicating that these tokens should be predicted by the model. This objective serves a dual purpose: it enables the model to learn how to generate coherent text given a prompt and, importantly, trains the model to follow specific instructions accurately.

To minimize environmental impact, the training process for CamelGPT-mini was completed in approximately two days, utilizing a single high-performance Intel Core i5-1245U CPU.

We are excited to announce that the model weights for CamelGPT-mini will be made publicly available on GitHub [GRb]. By releasing the model weights to the public, we aim to foster collaboration and facilitate further research and innovation in the field of natural language processing. The availability of the model weights on GitHub will empower researchers, developers, and enthusiasts to experiment with CamelGPT-mini, explore its capabilities, and potentially adapt it to diverse applications.

Moreover, we believe that the release of CamelGPT-mini's model weights will encourage the community to contribute to the development and enhancement of this novel approach to language modeling. As a proof of concept, CamelGPT-mini lays the foundation for future advancements in instruction-based language models and opens up new possibilities for efficient and effective text generation guided by instructions.

4 Performance Analysis

4.1 Evaluation and Comparative Analysis

To assess the effectiveness of CamelGPT-mini, we conducted a comprehensive evaluation of multiple benchmark datasets and compared its performance against several state-of-the-art language models. Our evaluation aimed to demonstrate CamelGPT-mini's ability to achieve high-level performance while demanding significantly fewer computational resources, making it a compelling solution for a wide range of natural language processing applications.

4.2 Benchmark Datasets

We selected a diverse set of benchmark datasets that cover various natural language processing tasks, including NaturalQuestions [KPR+19], TriviaQA [JCWZ17], and ToxiGen [HGP+22].

4.3 Experimental Setup

For each benchmark, we used standard evaluation metrics to measure the performance of CamelGPT-mini and the other comparison models. For non-CamelGPT models, we report the details of evaluation results from Llama 2 and Falcon [TMS+23].

4.4 Comparative Analysis

Our results show that CamelGPT-mini performs remarkably well across the evaluated benchmarks, achieving competitive performance when compared to larger state-of-the-art language models. Specifically, we observed notable reductions in memory usage and training time with CamelGPT-mini, without compromising performance.

Table 1: Benchmark Dataset

	CamelGPT-mini	Falcon	Llama 1	Llama 2
Size (Parameters)	56k	7B	7B	7B
NaturalQuestions (0-shot)	0.05	15.7	16.8	16.4
TriviaQA (0-shot)	0.14	52.6	63.3	65.8
ToxiGen	0.00	14.53	23.00	21.25
Time (GPU Hours)	0.712	N/A (undisclosed)	82,432	184,320

4.5 Discussion of Results

As shown in the table1, CamelGPT-mini achieves comparable performance to larger models represented by Llama 1 7B, Llama 2 7B, and Falcon 7B across various tasks. This demonstrates the efficacy of the EPDP approach, as CamelGPT-mini achieves similar accuracy and robustness with a significantly smaller model size.

Notably, we observed a substantial reduction in memory usage as a result of CamelGPT-mini's minimal parameter count which is 12500000% smaller than comparison models [TMS+23]. This is particularly significant for applications with limited computational resources or for deployment on edge devices.

Furthermore, CamelGPT-mini demonstrated a reduction in training time of up to 25888000% when compared to high-end models like Llama 2 7B[TMS+23]. This improvement in efficiency allows for faster model development and iteration, enabling researchers and developers to explore more ideas and refine their models more rapidly.

Importantly, CamelGPT-mini's strong performance across different domains and tasks underscores its versatility and generalization capabilities. This makes it a powerful tool for a wide array of natural language processing applications, from chatbots and virtual assistants to language translation and information retrieval systems.

5 Inference

We are excited to announce the release of our inference code for CamelGPT models, now available on GitHub [GRa]. This JavaScript-based code allows developers to efficiently make inferences with CamelGPT models, targeting a wide range of platforms, including web, Node.js, Edge, and Deno.

5.1 Inference-Only Code

The inference code we provide is solely dedicated to running CamelGPT models and does not contain any functionality to create new models. By focusing exclusively on inference, we aim to ensure the responsible and safe use of the technology, while promoting the accessibility of language models for diverse applications.

5.2 EPDP Integration with Official Model Releases

To facilitate the runtime phases of Eager Precached Dynamic Pruning (EPDP), we have leveraged binaries bundled alongside official CamelGPT model releases. This approach ensures seamless compatibility with EPDP without exposing the underlying implementation directly in JavaScript. The EPDP implementation bundled with official model releases provides users with efficiency and resource optimization benefits without compromising safety.

5.3 Compatibility Across Platforms

Our JavaScript-based inference code is designed to be compatible with various platforms, including web browsers, Node.js environments, Edge devices, and Deno runtimes. This broad compatibility empowers developers to deploy CamelGPT models in diverse contexts, offering flexibility in utilizing the power of language models where they are needed most.

5.4 Usage Example

To use the "@gpt-research/converters" package, developers can simply install it from npm and import it into their JavaScript project:

```
import { TextGenerationPipeline } from 'https://esm.sh/gh/gpt-research/converters/src/converters.js';

const main = async () => {
  // Initialize the pipeline with the desired model
  const pipeline = await TextGenerationPipeline("@gpt-research/CamelGPT-mini");

  // Generate text using the pipeline
  const generatedText = pipeline("Write a poem about camels.");

  // Log or use the generated text
  console.log(generatedText);
};

main();
```

6 Safety

6.1 Use of Private Dataset to Ensure Safety

To prioritize safety and mitigate potential risks associated with language model training, CamelGPT utilizes a private dataset for training purposes. This private dataset has been carefully curated to adhere to strict safety guidelines, ensuring that the model does not learn harmful or inappropriate information during the training process. By employing a curated dataset, we maintain a high level of control over the information that the model learns, promoting responsible AI development.

6.2 Non-Disclosure of Private Dataset

In the interest of adhering to our effective and altruistic policies, we have decided not to release the private dataset to the public. The decision to keep the dataset private is to prevent any potential misuse or unintended consequences that might arise from unauthorized access to the data. By limiting access to the dataset, we aim to protect the integrity and safety of the information it contains.

6.3 Restricted Release of Training Code

Similarly, we have chosen not to release the training code at this time due to safety reasons. The training code plays a crucial role in the model development process, and its unrestricted release could lead to misuse or unethical practices. We take the responsibility of ensuring the responsible use of technology seriously and, therefore, have decided to keep the training code restricted to maintain control over its usage.

6.4 Limited Release of EPDP Implementation

The Eager Precached Dynamic Pruning (EPDP) implementation, a critical component of CamelGPT's resource-efficient architecture, will not be released at this time due to safety reasons. EPDP empowers users to develop high-quality models at scale with minimal computational resources. However, providing access to the EPDP implementation without proper oversight might lead to unintended consequences or unethical use. Hence, for safety and responsible AI development, we are currently refraining from releasing the EPDP implementation to the public.

6.5 Safety Concerns with Code Accessibility

The primary safety concern behind restricting access to certain components of CamelGPT lies in the potential misuse of the technology. Allowing open access to code that empowers the creation of powerful language models with minimal resources could lead to the generation of malicious content, misinformation, or other harmful applications. By controlling access to sensitive components, we aim to prevent such misuse and promote responsible AI development.

6.5.1 Responsible Usage through Official Model Releases

To ensure the responsible and ethical application of CamelGPT, we strongly encourage users to utilize official model releases provided by our team. These official model releases are thoroughly evaluated and adhere to safety guidelines, enabling users to harness the power of CamelGPT responsibly and without compromising safety.

7 Conclusion

In this paper, we introduced CamelGPT, a groundbreaking resource-efficient approach to large language model development, addressing the challenges posed by the resource-intensive nature of modern models[PGL$^+$21][SPS20]. Using Eager Precached Dynamic Pruning (EPDP), CamelGPT achieves high-level performance with minimal computational resources and storage capacity. Our experiments 4 on benchmark datasets demonstrate CamelGPT's effectiveness in attaining comparable performance to larger models.

The core innovation of CamelGPT lies in its ability to train and run smaller models that maintain comparable accuracy. By dynamically pruning unnecessary parameters during training and inferencing, CamelGPT optimizes its architecture from the outset, leading to resource-efficient models with impressive language processing capabilities. This approach not only reduces the environmental impact of large-scale model training but also dismantles barriers for organizations and individuals with limited resources.

CamelGPT holds the promise of democratizing access to large language models, and revolutionizing natural language processing. Our JavaScript-based inference code on GitHub [GRa] enhances accessibility, enabling developers to leverage CamelGPT models effortlessly.

Because we prioritize safety and ethics, we have chosen not to release the private dataset used for training CamelGPT and related code implementations. This decision aligns with our commitment to responsible AI development. We encourage the responsible use of CamelGPT through official model releases to ensure safety and ethical standards.

CamelGPT represents a significant advancement, offering an efficient and accessible solution to natural language processing challenges. By combining the benefits of EPDP with thoughtful implementation choices, CamelGPT paves the way for a more sustainable and inclusive AI future, unlocking new possibilities for language processing applications and empowering a wider range of users. This research sets a strong foundation for future developments in resource-efficient language modeling, steering the field toward more equitable and responsible AI practices.

References

[BHA+21] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[BMR+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[GRa] GPT-Research. GitHub - gpt-research/converters: Converters: State-of-the-art Machine Learning for Javascript, Typescript, Node, Deno, Bun — github.com. `https://github.com/gpt-research/converters`. [Accessed 25-11-2023].

[GRb] GPT-Research. GitHub - gpt-research/hub: The gpt research Hub is a treasure trove of open-source model weights from pioneering research projects. Our mission is to democratize access to advanced language models while prioritizing responsible AI development. Explore, innovate, and contribute to shape the future of AI in a safe and inclusive environment! — github.com. `https://github.com/gpt-research/hub`. [Accessed 25-11-2023].

[HGP+22] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *arXiv preprint arXiv:2203.09509*, 2022.

[JCWZ17] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[KPR+19] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[LWS+20] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on machine learning*, pages 5958–5968. PMLR, 2020.

[PGL+21] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[SPS20] Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*, 2020.

[SSZ+23] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arxiv:2305.17493*, 2023.

[TMS+23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.