RecurrentGemma: Moving Past Transformers for Efficient Open Language Models

Griffin[1], **RLHF**[1] **and Gemma Teams**[1]
[1]Google DeepMind. Please see contributors and acknowledgements section for full author list.

We introduce RecurrentGemma, an open language model which uses Google's novel Griffin architecture. Griffin combines linear recurrences with local attention to achieve excellent performance on language. It has a fixed-sized state, which reduces memory use and enables efficient inference on long sequences. We provide a pre-trained model with 2B non-embedding parameters, and an instruction tuned variant. Both models achieve comparable performance to Gemma-2B despite being trained on fewer tokens.

Introduction

We present RecurrentGemma-2B, an open model based on the Griffin architecture (De et al., 2024). This architecture eschews global attention, instead modelling the sequence through a mixture of linear recurrences (Gu et al., 2021; Orvieto et al., 2023) and local attention (Beltagy et al., 2020). RecurrentGemma-2B achieves superb performance on downstream tasks, competitive with Gemma-2B (Gemma Team, 2024), an open transformer model (Vaswani et al., 2023) based on insights from Gemini (Gemini Team, 2023).

To perform inference, transformers must retrieve the KV cache and load it into device memory. This KV cache grows linearly with sequence length. Although one can reduce the cache size by using local attention (Beltagy et al., 2020), this comes at the price of reduced performance. In contrast, RecurrentGemma-2B compresses input sequences into a fixed-size state without sacrificing performance. This reduces memory use and enables efficient inference on long sequences. We verify below that RecurrentGemma-2B achieves substantially faster inference than Gemma-2B.

We are releasing both a pre-trained checkpoint and an instruction tuned checkpoint, fine-tuned for instruction-following and dialogue similar to Gemma (Gemma Team, 2024). We are also releasing efficient JAX code to evaluate and fine-tune our models (Bradbury et al., 2018), including a specialized Pallas kernel to perform the linear recurrence on TPU. We provide a reference Pytorch implementation as well (Paszke et al., 2019).

Table 1 | Key model hyper-parameters. See Griffin paper (De et al., 2024) for model definition.

Total params	2.7B
Non-Embedding params	2.0B
Embedding params	0.7B
Vocabulary size	256k
Model width	2560
RNN width	2560
MLP expansion factor	3
Depth	26
Attention heads	10
Local attention window size	2048

Model architecture

We make only a single modification to the Griffin architecture (De et al., 2024), which is to multiply the input embeddings by a constant equal to the square root of model width. The input and output embeddings are tied, but this factor is not applied to the output. A similar multiplicative factor appears in Gemma (Gemma Team, 2024). We define the key model hyper-parameters in Table 1, and defer the reader to De et al. (2024) for exact details on the overall architecture.

Note that we do not apply weight decay to the parameters of the recurrent (RG-LRU) layers during training. Additionally when backpropagating through the square root operation, we always clip the derivative to a maximum value of 1000 for stability.

Corresponding author(s): [botev, slsmith, sohamde, anushanf]@google.com

Table 2 | Academic benchmark results, compared to the Gemma-2B model.

Benchmark	Metric	Gemma-2B	RecurrentGemma-2B
MMLU	5-shot, top-1	42.3	38.4
HellaSwag	0-shot	71.4	71.0
PIQA	0-shot	77.3	78.5
SIQA	0-shot	49.7	51.8
Boolq	0-shot	69.4	71.3
Winogrande	partial scoring	65.4	67.8
CQA	7-shot	65.3	63.7
OBQA		47.8	47.2
ARC-e		73.2	72.9
ARC-c		42.1	42.3
TriviaQA	5-shot	53.2	52.5
NQ	5-shot	12.5	11.5
HumanEval	pass@1	22.0	21.3
MBPP	3-shot	29.2	28.8
GSM8K	maj@1	17.7	13.4
MATH	4-shot	11.8	11.0
AGIEval		24.2	23.8
BBH		35.2	35.3
Average		44.9	44.6

Training details

Pre-training

We train on sequences of 8192 tokens. We use the same pre-training data as Gemma-2B, which comprises primarily English data from web documents, mathematics and code. This dataset was filtered to reduce the risk of unwanted or unsafe utterances, and to filter out personal or sensitive data as well as to filter out all evaluation sets from our pre-training dataset. We refer to the Gemma report for more details (Gemma Team, 2024).

We pre-train RecurrentGemma-2B on 2T tokens. Note that in contrast, Gemma-2B was pre-trained on 3T tokens. Like Gemma, we first train on a large general data mixture, beforing continuing training on a smaller, higher quality dataset. Like Gemma, we use a subset of the SentencePiece tokenizer (Kudo and Richardson, 2018), with a vocabulary size of 256k tokens. Note that, as a consequence of this large vocabulary size, the embedding layer comprises a significant fraction of the total model parameters.

Table 3 | Relevant formatting control tokens used for both SFT and RLHF of Gemma models.

Context	Relevant Token
User turn	`user`
Model turn	`model`
Start of conversation turn	`<start_of_turn>`
End of conversation turn	`<end_of_turn>`

Instruction tuning and RLHF

We follow a similar instruction tuning approach to Gemma (Gemma Team, 2024), including a novel RLHF algorithm to fine-tune the model to output responses with high reward. Our instruction tuned model is trained to obey a specific dialogue format, which is defined in Table 3. For clarity, we give a concrete example in Table 4.

Evaluation

We evaluate RecurrentGemma-2B across a broad range of domains, using a combination of auto-

Table 4 | Example dialogue with control tokens.

User:	<start_of_turn>user Knock knock.<end_of_turn> <start_of_turn>model
Model:	Who's there?<end_of_turn>
User:	<start_of_turn>user Gemma.<end_of_turn> <start_of_turn>model
Model:	Gemma who?<end_of_turn>

mated benchmarks and human evaluation.

Automated Benchmarks

We report the performance of RecurrentGemma-2B on a range of popular downstream evaluations in Table 2. RecurrentGemma-2B achieves comparable performance to Gemma-2B, even though Gemma-2B was trained on 50% more tokens.

Human Evaluation

We sent our final instruction tuned model (RecurrentGemma-2B-IT) for human evaluation studies against the Mistral 7B v0.2 Instruct model (Jiang et al., 2023). As shown in Table 5, on a held-out collection of around 1000 prompts oriented toward asking models to follow instructions across creative writing and coding tasks, RecurrentGemma-2B-IT achieves a 43.7% win rate against the larger Mistral 7B model, only slightly below the 45.0% win rate achieved by Gemma-1.1-2B-IT.

On a held-out collection of around 400 prompts oriented towards testing basic safety protocols, RecurrentGemma-2B-IT achieved a 59.8% win rate against Mistral 7B v0.2 Instruct model.

Inference Speed Benchmarks

A key advantage of RecurrentGemma is that it has a significantly smaller state size than transformers on long sequences. Whereas Gemma's KV cache grows proportional to sequence length, RecurrentGemma's state is bounded, and does not increase on sequences longer than the local attention window size of 2k tokens. Consequently, whereas the longest sample that can be generated auto-regressively by Gemma is limited by the

Table 5 | Win rate of RecurrentGemma-2B-IT and Gemma-1.1-2B-IT against Mistral 7B v0.2 Instruct, under human evaluation with 95% confidence intervals. We report a breakdown of wins, ties and losses, and break ties evenly when reporting the final win rate. RecurrentGemma-2B-IT achieves similar performance to Gemma-1.1-2B-IT, and is surprisingly competitive with the larger Mistral 7B model.

Model	Safety	Instruction Following
RecurrentGemma	59.8%	43.7%
95% Conf. Interval	[57.1%, 62.6%]	[41.8%, 45.6%]
Win / Tie / Loss	47.5% / 24.6% / 27.9%	34.5% / 18.3% / 47.2%
Gemma 1.1	60.1%	45.0%
95% Conf. Interval	[57.3%, 62.8%]	[43.1%, 46.9%]
Win / Tie / Loss	48.5% / 23.2% / 28.3%	37.1% / 15.8% / 47.1%

memory available on the host, RecurrentGemma can generate sequences of arbitrary length.

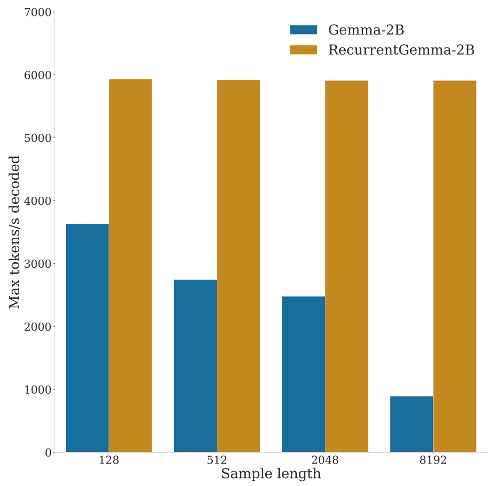
Since inference is typically memory-bound, RecurrentGemma can generate samples more efficiently than the Gemma model. In particular, the reduced memory requirement enables RecurrentGemma to perform inference at larger batch sizes, which amortizes the cost of loading model parameters from host memory into device memory.

In Figure 1a, we plot the throughput achieved when sampling from a prompt of 2k tokens for a range of generation lengths. The throughput calculates the maximum number of tokens we can sample per second on a single TPUv5e device. Note that in this plot, we do not account for the time required to process the prompt or the time required to convert the output sequence from a list of token ids into the final text string. RecurrentGemma achieves higher throughput at all sequence lengths considered. The throughput achieved by RecurrentGemma does not reduce as the sequence length increases, while the throughput achieved by Gemma falls as the cache grows.

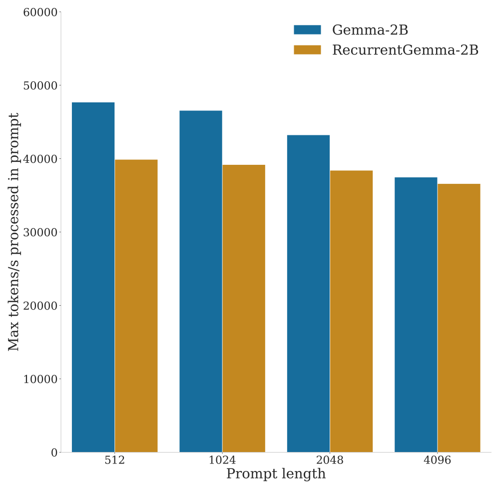
For completeness, in Figure 1b, we show the throughput achieved when processing input prompts. Unlike auto-regressive sampling, the prompt is processed in parallel. Gemma and RecurrentGemma process input prompts at similar speeds. When processing the prompt, both Gemma and RecurrentGemma achieve through-

Table 6 | Safety academic benchmark results. We provide results for both our pre-trained checkpoint and our instruction tuned variant.

Benchmark	metric	RecurrentGemma-2B	RecurrentGemma-2B-IT
RealToxicity	avg	9.8	7.6
BOLD		39.3	52.3
CrowS-Pairs	top-1	41.1	43.4
BBQ Ambig	top-1	62.6	71.1
BBQ Disambig	top-1	58.4	50.8
Winogender	top-1	55.1	54.7
TruthfulQA		35.1	42.7
Winobias 1_2		58.4	56.4
Winobias 2_2		90.0	75.4
Toxigen		56.7	50.0



(a) Throughput when sampling from a 2k prompt



(b) Throughput when processing prompts

Figure 1 | Maximum tokens per second generated on a single TPUv5e, when (a) sampling sequences of different lengths from a prompt of 2k tokens, and (b) when processing prompts of different lengths to generate the initial state from which to sample.

put of roughly 40k tokens per second. By contrast, when sampling RecurrentGemma achieves throughput of 6k tokens per second, with Gemma substantially slower. Thus, sampling will dominate the total time required, unless the prompt is significantly longer than the desired sample.

Figures 1a and 1b were generated using the Flax implementation of RecurrentGemma, which includes a specialized Pallas kernel for TPU. Users should expect lower throughput when using the Pytorch implementation or when using GPUs. We perform inference for Gemma using a modified version of Gemma's Flax implementation, which we optimized further to improve performance.

Responsible Deployment

We follow the same safety mitigations as described in the Gemma release (Gemma Team, 2024). We evaluated our models on standard academic safety benchmarks, as shown in Table 6, and our final models were also subjected to ethics and safety evaluations by an independent team before release. However, our testing cannot cover all possible use cases of RecurrentGemma, and thus we recommend all users of RecurrentGemma to conduct their own safety testing, specific to their use-case, prior to deployment.

Conclusion

RecurrentGemma-2B offers the performance of Gemma, while achieving higher throughput during inference, especially on long sequences. We hope that RecurrentGemma will unlock novel applications of highly performant small language models in resource constrained environments.

Contributions and Acknowledgments

Griffin Team
Aleksandar Botev†
Soham De†
Samuel L Smith†
Anushan Fernando†
George-Cristian Muraru†
Ruba Haroun†
Leonard Berrada†
Razvan Pascanu

RLHF
Pier Giuseppe Sessa
Robert Dadashi
Léonard Hussenot
Johan Ferret
Sertan Girgin
Olivier Bachem

Gemma Team
Alek Andreev
Kathleen Kenealy
Thomas Mesnard
Cassidy Hardin
Surya Bhupatiraju
Shreya Pathak
Laurent Sifre
Morgane Rivière
Mihir Sanjay Kale
Juliette Love
Pouya Tafti
Armand Joulin
Noah Fiedel
Evan Senter

Contributors
Yutian Chen
Srivatsan Srinivasan
Guillaume Desjardins
David Budden
Arnaud Doucet
Sharad Vikram
Adam Paszke
Trevor Gale
Sebastian Borgeaud
Charlie Chen
Andy Brock
Antonia Paterson
Jenny Brennan
Meg Risdal
Raj Gundluru
Nesh Devanathan
Paul Mooney
Nilay Chauhan
Phil Culliton
Luiz GUStavo Martins
Elisa Bandy
David Huntsperger
Glenn Cameron
Arthur Zucker

Product Management
Tris Warkentin
Ludovic Peran

Program Management
Minh Giang

Executive Sponsors
Nando De Frietas
Yee Whye Teh
Raia Hadsell
Zoubin Ghahramani
Clément Farabet
Koray Kavukcuoglu
Demis Hassabis

Acknowledgements
Our work is made possible by the dedication and efforts of numerous teams at Google. We would like to acknowledge the support from the following teams: Gemini, Gemini Safety, Gemini Infrastructure, Gemini Evaluation, Google Cloud, Google Research Responsible AI and Kaggle.

† Joint first authors.

References

I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, et al. Jax: composable transformations of python+numpy programs. 2018.

S. De, S. L. Smith, A. Fernando, A. Botev, G. Cristian-Muraru, A. Gu, R. Haroun, L. Berrada, Y. Chen, S. Srinivasan, G. Desjardins, A. Doucet, D. Budden, Y. W. Teh, R. Pascanu, N. D. Freitas, and C. Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024.

Gemini Team. Gemini: A family of highly capable multimodal models, 2023.

Gemma Team. Gemma: Open models based on gemini research and technology, 2024.

A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023.

T. Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, 2018.

A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De. Resurrecting recurrent neural networks for long sequences. *arXiv preprint arXiv:2303.06349*, 2023.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.